
بسم الله الرحمن الرحیم

دانشکده : ریاضی

گروه : ریاضی کاربردی

عنوان پایان نامه ارشد

ارایه یک مدل شبکه عصبی برای حل مساله رگرسیون

دانشجو : عبدالمجید دولتی

استاد راهنما :
جناب آقای دکتر علیرضا ناظمی

پایان نامه ارشد جهت اخذ درجه کارشناسی ارشد

بهمن ۹۲

تقدیم به

پسرم عزیزم مهدی دولتی

خدایا...

به من زیستنی عطا کن که در لحظه مرگ، بر بی‌ثمری لحظه‌ای که برای زیستن گذشته است، حسرت نخورم و مُردنی عطا کن که بر بیهودگیش، سوگوار نباشم. بگذار تا آن را، خود انتخاب کنم، اما آنچنان که تو دوست می‌داری.

تو می‌دانی و همه می‌دانند که شکنجه دیدن بخاطر تو، زندانی کشیدن بخاطر تو و رنج بردن به پای تو تنها لذت بزرگ زندگی من است، از شادی توست که من در دل می‌خندم، از امید رهایی توست که برق امید در چشمان خسته‌ام می‌درخشد و از خوشبختی توست که هوای پاک سعادت را در ریه‌هایم احساس می‌کنم. نمی‌توانم خوب حرف بزنم. نیروی شگفتی را که در زیر کلمات ساده و جمله‌های ضعیف و افتاده، پنهان کرده‌ام، دریاب.

تو می‌دانی و همه می‌دانند که زندگی از تحمیل لبخندی بر لبان من، از آوردن برق امیدی در نگاه من، از برانگیختن موج شعفی در دل من، عاجز است.

تو، چگونه زیستن را به من بیاموز، چگونه مردن را خود خواهم آموخت.

به من توفیق تلاش در شکست، صبر در نومیدی، رفتن بی‌همراه، جهاد بی‌سلاح، کار بی‌پاداش، فداکاری در سکوت، دین بی‌دنیا، مذهب بی‌عوام، عظمت بی‌نام، خدمت بی‌نان، ایمان بی‌ریا، خوبی بی‌نمود، گستاخی بی‌خامی، قناعت بی‌غرور، تنهایی در انبوه جمعیت، و دوست داشتن بی‌آنکه دوست بداند، روزی کن.

تشکر و قدردانی

سپاس خداوندگار حکیم را که با لطف بی‌کران خود، آدمی را زیور عقل آراست و تقدیم سزاوار هر آنچه و هر آنکه لطف و محبت اوست. خدا را شاکرم که به من این توان را داد که این مرحله از زندگی را با سلامتی و سر بلندی پشت سر بگذارم و سر بر آستان پربرکت او می سایم که هر چه عزت و سرافرازی هست، از اوست.

در آغاز وظیفه خود می دانم از زحمات بی دریغ استاد راهنمای خود، جناب آقای دکتر علیرضا ناظمی، صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی های ارزنده ایشان، این مجموعه به انجام نمی رسید.

در پایان، بوسه می زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می کنم وجود مقدس شان را و تشکر می کنم از برادران عزیزم به پاس عاطفه سرشار و گرمای امید بخش وجودشان، که بهترین پشتیبان من بودند.

عبدالمجید دولتی
بهمن ۱۳۹۲

نام خانوادگی: دولتی نام: عبدالمجید ث

عنوان پایان‌نامه: ارایه یک مدل شبکه عصبی برای حل مساله رگرسیون

استاد راهنما: دکتر علیرضا ناظمی

استاد مشاور:

مقطع تحصیلی: کارشناسی ارشد رشته: ریاضی کاربردی گرایش: تحقیق در عملیات

دانشگاه: دانشگاه صنعتی شاهرود دانشکده: علوم ریاضی

تاریخ فارغ‌التحصیلی: زمستان ۱۳۹۲ تعداد صفحه: ۸۷

کلیدواژه‌ها: ماشین بردار پشتیبان، شبکه‌های عصبی، رگرسیون، قاعده یادگیری دلتا، پرسپترون،

چکیده

ماشین های بردار پشتیبان SVMs ، با انگیزه نظریه یادگیری آماری ، در سال های اخیر مورد توجه بسیاری از پژوهشگران قرار گرفته است. اصول نظریه ماشین‌های بردار پشتیبان توسط واپنیک [1] توسعه داده شده است. ویژگی اصلی SVMs ها این است که آنها با به حداقل رساندن ریسک ساختار به جای به حداقل رساندن ریسک تجربی به کار می روند. بر خلاف پرسپترون چند لایه MLP شناخته شده و شبکه های تابع RBF، آموزشSVM معادل با حل مساله برنامه ریزی محدب خطی مقید درجه دوم است. از آنجا که شبکه های MLP و RBF در به حداقل رساند نتایج خطا غیر خطی تکیه می کنند که ممکن است نامحدب باشند، مساله مینیمم موضعی در آموزش با استفاده از روش SVM می تواند نادیده گرفته شود. SVMs برای اولین بار برای حل مساله طبقه بندی توسعه پیدا کردند. از آنجا که SVMs دارای خواص قوی در برابر نویز بودند، SVMs در دامنه گسترده‌ای از مسایل رگرسیون نیز وارد شده است. بر اساس روش ها بهینه سازی عددی، چندین الگوریتم و اصلاحاتشان را برای طبقه بندی بردار پشتیبانی SVC و رگرسیون بردار پشتیبانی SVR پیشنهاد شده است. در بسیاری ازکاربرد های مهندسی و نظامی، خواسته ها در پردازش داده ها در زمان واقعی است که اغلب مورد نیاز است، مانند طبقه بندی در محیط الکترومغناطیس مختلط، در تشخیص هویت پزشکی و در تشخیص شیء رادار در درهم و برهمی پس زمینه قوی، و غیره. بنابراین، به موازات و رویکردهای توزیع نیاز به آموزش SVMs ضروری و مطلوب هستند. به خوبی شناخته شده است که یکی از مهم ترین مزایای آنها در توانایی پردازش در زمان واقعی از شبکه های عصبی است. در سال های اخیر، رویکرد شبکه های عصبی کارایی‌های زیاد خود را برای بهینه سازی نشان داده‌اند. نتایج گزارش شده بسیاری از تحقیقات مزایای استفاده بیش از الگوریتم های بهینه سازی سنتی را نشان داده اند، به خصوص در کاربردهای زمان واقعی. یک شبکه عصبی دو لایه برای SVC استفاده شده است. شبکه های عصبی برای پیاده سازی سخت افزاری

[1] Vapnik

آنالوگ مناسب است و به یک راه حل خوب SVC منجر می شود. در این پایان نامه یک شبکه عصبی تک لایه برای SVC و یادگیری SVR پیشنهاد شده است. مزیت شبکه عصبی پیشنهادی دو قسم است. بر خلاف شبکه عصبی دولایه موجود، شبکه عصبی پیشنهادی پیچیدگی کمی برای پیاده سازی دارد. تجزیه و تحلیل نظری نشان می دهدکه شبکه عصبی پیشنهادی می تواند همگرا نمایی به جواب بهینه از آموزش SVM باشد. علاوه بر این، نرخ همگرایی نمایی را می توان به طور دلخواه با یک پارامتر شبکه ساده زیاد کرد. سه نمونه گویا نشان دهنده عملکرد برترعصبی پیشنهادی شبکه ای برای یادگیری SVM هستند.

فهرست مطالب

[۲] Artificial neural network

[۳] Support vector regression

لیست تصاویر

فصل ۱

مقدمه

در این پایان نامه یک شبکه عصبی[۱] تک لایه بازگشتی برای ماشین بردار پشتیبانی[۲] (SVM) در الگوی یادگیری طبقه بندی و رگرسیون را ارائه می کنیم. اولین مساله یادگیری SVM تبدیل به فرمول معادل آن، و پس از آن یک لایه شبکه های عصبی بازگشتی برای یادگیری SVM پیشنهاد شده است. شبکه عصبی پیشنهادی برای به دست آوردن راه حل بهینه از طبقه بندی بردار حامی و رگرسیون تضمین شده است. شبکه های عصبی موجود دو لایه برای طبقه بندی ،SVM در مقایسه با شبکه عصبی پیشنهادی پیچیدگی کمتری برای پیاده سازی دارد. علاوه بر این، شبکه عصبی پیشنهادی می تواند همگرای نمایی به جواب بهینه از یادگیری SVM باشد. نرخ همگرایی نمایی می تواند به سادگی با تبدیل کردن یک پارامتر مقیاس به صورت دلخواهی بالا رود. نمونه های شبیه سازی بر اساس مشکلات پایه مورد بحث قرار گرفته اند تا عملکرد درست شبکه عصبی پیشنهادی برای یادگیری SVM نشان دهند.

۱.۱ کاربرد مساله رگرسیون

مساله رگرسیون در زمینه های مختلف علوم و مهندسی کاربرد فراوانی دارد. بالاخص در مهندسی و بسیاری از کاربردهای نظامی که برنامه های کاربردی، نیازمند پردازش در زمان واقعی داده ها

[۱] neural network

[۲] Support Vector Machine

است، اغلب نیازمند استفاده از شبکه های عصبی به عنوان یک ابزار کارا در یافتن جواب های زمان واقعی هستیم. چون زمان مورد نیاز برای حل مسأله رگرسیون داده ها وابستگی زیادی به بعد و ساختار مسأله دارد لذا معمولاً روش های عددی مرسوم در کاربردهای واقعی چندان مؤثر نیستند. یک رهیافت امیدوار کننده عبارت است از استفاده از شبکه های عصبی بازگشتی مبنی بر پیاده سازی دوری با مدارهای الکترونیکی. در این پایان نامه یک شبکه عصبی[۳] تک لایه بازگشتی برای ماشین بردار پشتیبانی [۴] در الگوی یادگیری طبقه بندی و رگرسیون را ارائه می کنیم. اولین مساله یادگیری SVM تبدیل به فرمول معادل آن ، و پس از آن یک لایه شبکه های عصبی بازگشتی برای یادگیری SVM پیشنهاد شده است. شبکه عصبی پیشنهادی برای به دست آوردن راه حل بهینه از طبقه بندی بردار حامی و رگرسیون تضمین شده است. شبکه های عصبی موجود دو لایه برای طبقه بندی SVM، در مقایسه با شبکه عصبی پیشنهادی پیچیدگی کمتری برای پیاده سازی دارد. علاوه بر این، شبکه عصبی پیشنهادی می تواند همگرای نمایی به جواب بهینه از یادگیری SVM باشد. نرخ همگرایی نمایی می تواند به سادگی با تبدیل کردن یک پارامتر مقیاس به صورت دلخواهی بالا رود. نمونه های شبیه سازی بر اساس مشکلات پایه مورد بحث قرار گرفته اند تا عملکرد درست شبکه عصبی پیشنهادی برای یادگیری SVM نشان دهند.

۲.۱ تاریخچه تحقیق

قدمت شبکه‌های عصبی مصنوعی به ۵۰ سال پیش می‌رسد. در سال ۱۹۵۶ بنیاد راکفلر برگزاری کنفرانسی را بر عهده داشت که چشم‌اندازش به صورت زیر بود:

امکان استفاده از کامپیوترها و شبیه سازی در هر زمینه از یادگیری و سایر حوزه‌های آموزش.

در همین کنفرانس بود که اصطلاح هوش مصنوعی مورد استفاده عمومی قرار گرفت. بطور کلی هوش مصنوعی را به صورت زیر می‌توان تعریف نمود.

فرایندهای کامپیوتری که سعی دارند تفکر انسان را تقلید نمایند، این فرایندها با فعالیت‌هایی که نیاز به استفاده از هوش دارند در ارتباطند.

عموماً این تعریف شامل زمینه‌های یادگیری خودکار، فهم زبان طبیعی، بینایی و تشخیص صدا، بازی کردن، حل مسائل ریاضی، رباتیک و سیستم‌های خبره است. در سال‌های اخیر برای محققان،

[۳] Neural network

[۴] SVM

شبکه‌های عصبی و فناوری‌های مرتبط با آن را به عنوان بخشی از هوش مصنوعی در نظر می‌گیرند. در حالی که بعضی دیگر با توجه به سررشته خود در علوم پزشکی، شبکه‌های عصبی را زیرمجموعه هوش مصنوعی نمی‌دانند.

۳.۱ اهداف و ضرورت انجام پایان‌نامه

در این پایان نامه یک شبکه عصبی بازگشتی برای مساله رگرسیون را ارائه می کنیم. ابتدا مساله رگرسیون یک خانواده از داده ها را بصورت یک مساله برنامه ریزی درجه دوم مدل بندی می‌کنیم. سپس با ارائه یک مدل شبکه عصبی کارا برای مساله برنامه ریزی درجه دوم بدست آمده، تابع رگرسیون داده ها را بدست خواهیم آورد. ما ثابت خواهیم کرد که مدل بدست آمده پایداری لیاپانوف و همگرای سراسری به جواب مساله خواهد بود. قابل توجه است که در مساله رگرسیون می‌توان از توابع هسته مختلفی از جمله هسته‌های نمایی، چندجمله‌ای و سیگموید استفاده نمود.

مساله رگرسیون داده ها تابحال توسط افراد مختلف و با روشها و الگوریتمهای مختلف مورد بررسی قرار گرفته است. برای اطلاع بیشتر منابع [۶۷]- [۷۸] را ببینید.

فصل ۲

مفاهیم اولیه و پیش‌نیازها

۱.۲ تعاریف

در این فصل تعاریف مورد نیاز در فصل‌های بعدی به‌طور مختصر آورده شده است. ابتدا تعاریف مرتبط با تابع محدب [۱] را بیان نموده و در خصوص مسائل بهینه‌سازی مقید و نامقید به ذکر شرایط لازم و کافی بهینگی [۲] می‌پردازیم. در انتها مفاهیم پایداری [۳] و تابع انرژی در سیستم‌های دینامیکی را ارائه می‌دهیم.

۲.۲ توابع آفین و محدب

تعریف ۱.۲. *تابع $f : \mathbb{R}^n \to \mathbb{R}$ را تابع خطی می‌گوییم، هرگاه حافظ عملیات در فضای برداری $\mathbb{R}^n$ باشد، یعنی:*

$$\begin{aligned} f(x+y) =& f(x) + f(y), \\ f(\alpha x) =& \alpha f(x). \end{aligned}$$

برای هر $x, y \in \mathbb{R}^n$ و هر $\alpha \in \mathbb{R}$.

[۱] Convex

[۲] Optimality

[۳] Stability

تعریف ۲.۲. *تابع $f : \mathbb{R}^n \to \mathbb{R}^l$ را تابع آفین می‌گوییم، هرگاه بصورت مجموع یک تابع خطی و یک تابع ثابت باشد. به عنوان مثال $f(x) = Ax + b$ که $A \in \mathbb{R}^{l\times n}$ و $b \in \mathbb{R}^l$ تابع آفین است.*

تعریف ۳.۲. *مجموعه $E \subset \mathbb{R}^n$ را محدب می‌گوییم اگر:*

$$\forall x, y \in E, \lambda \in (۰, ۱) \ : \ \lambda x + (۱ - \lambda)y \in E.$$

تعریف ۴.۲. *تابع $f : \mathbb{R}^n \to \mathbb{R}$ بر روی مجموعه نقاط واقع در مجموعه محدب $E \subset \mathbb{R}^n$ یک تابع محدب نامیده می‌شود، اگر:*

$$\forall x, y \in E, \ \lambda \in (۰, ۱) \ : \ f(\lambda x + (۱ - \lambda)y) \leq \lambda f(x) + (۱ - \lambda)f(y).$$

تابع f را روی E مقعر گوییم هرگاه $-f$ محدب باشد.

تعریف ۵.۲. *اگر $\bar{x} \in X$ و $\varepsilon-$همسایگی از $\bar{x}$ مثل $N_\varepsilon(\bar{x})$ موجود باشد که به ازای هر $x \in X \cap N_\varepsilon(\bar{x})$ داشته باشیم $f(\bar{x}) \leq f(x)$ آنگاه، $\bar{x}$ یک کمینه موضعی برای f است.*

در توسعه برنامه‌ریزی غیرخطی، تابع محدب همواره به عنوان یک مفهوم اصلی ریاضی مورد نیاز است.

تعریف ۶.۲. *تابع $f(x) = f(x_۱, x_۲, ..., x_n) \in \mathbb{R}$ بر روی یک مجموعه نقاط واقع در یک مجموعه محدب $E \subset \mathbb{R}^n$ یک تابع محدب نامیده می‌شود اگر:*

$$\forall \ x, y \in E, \lambda \in (۰, ۱) \ : \ f(\lambda x + (۱ - \lambda)y) \leq \lambda f(x) + (۱ - \lambda)f(y).$$

تابع f را روی E مقعر گوییم هر گاه $-f$ محدب باشد.

تعریف ۷.۲. *تابع $\|.\| : \mathbb{R}^n \to \mathbb{R}$ را تابع نرم گوییم هرگاه به ازای هر $x, y \in \mathbb{R}$ و $\alpha \in \mathbb{R}$:*

$$\begin{cases} \|x\| \geqslant ۰, \quad \forall \ x, \\ \|x\| = ۰ \Leftrightarrow x = ۰, \\ \|\alpha x\| = |\alpha|\|x\|, \\ \|x + y\| \leqslant \|x\| + \|y\|. \end{cases}$$

برای بردار $x \in \mathbb{R}^n$ نرم‌های L_1، L_2 و L_r به صورت زیر تعریف می‌شوند،

$$\|x\|_1 = |x_1| + ... + |x_n|,$$
$$\|x\|_2 = \sqrt{x^T x} = \sqrt{x_1^2 + ... + x_n^2},$$
$$\|x\|_r = \sqrt[r]{x_1^r + ... + x_n^r}, \quad r \geq 1.$$

تعریف ۸.۲. یک زیر مجموعه E از فضای متریک X را یک مجموعه فشرده می‌گوییم اگر هر پوشش باز E یک زیر پوشش باز متناهی داشته باشد.

شرایط لازم و کافی برای مسائل نامقید

قضیه ۱.۲. (شرط لازم مرتبه اول [۳۲]): فرض کنید X یک مجموعه باز در $\mathbb{R}^n$ باشد. مسأله برنامه‌ریزی زیر را در نظر بگیرید:

$$\text{minimize} \qquad f(x) \tag{۱.۲}$$
$$\text{subject to}$$
$$x \in X, \tag{۲.۲}$$

که در آن $f : \mathbb{R}^n \to \mathbb{R}$ در $\hat{x}$ مشتق‌پذیر است. اگر $\hat{x}$ یک کمینه موضعی باشد، آنگاه $\nabla f(\hat{x}) = 0$.

تعریف ۹.۲. در مسأله (۱.۲)، اگر $\bar{x} \in X$ و $\varepsilon-$همسایگی از $\bar{x}$ مثل $N_\varepsilon(\bar{x})$ موجود باشد که به ازای هر $x \in X \cap N_\varepsilon(\bar{x})$ داشته باشیم $f(\bar{x}) \leq f(x)$ آنگاه $\bar{x}$ یک کمینه موضعی برای مسأله (۱.۲) است.

قضیه ۲.۲. (شرط لازم مرتبه دوم [۳۲]): فرض کنید مشتق دوم $f : \mathbb{R}^n \to \mathbb{R}$ در $\hat{x}$ موجود باشد. اگر $\hat{x}$ یک کمینه موضعی باشد، آنگاه $\nabla f(\hat{x}) = 0$ و ماتریس هسین [۴] f در $\hat{x}$ نیمه معین مثبت است.

قضیه ۳.۲. (شرط کافی [۳۲]): فرض کنید مشتق دوم $f : \mathbb{R}^n \to \mathbb{R}$ در $\hat{x}$ وجود باشد. اگر $\nabla f(\hat{x}) = 0$ و ماتریس هسین f در $\hat{x}$ معین مثبت باشد، آنگاه $\hat{x}$ یک کمینه موضعی اکید است.

[۴] Hessian

قضیه ۴.۲. ([۳۲]) . *فرض کنید مجموعه* E *بصورت زیر باشد:*

$$E = \{x \in \mathbb{R}^n \mid g_k(x) \leq ۰,\ k = ۱, \cdots, m,\ h_p(x) = ۰,\ p = ۱, \cdots, l\}$$

اگر $g_k(x)$ ، $k = ۱, \cdots, m$ *توابعی محدب و* $h_p(x)$ ، $p = ۱, \cdots, l$ *آفین باشند آنگاه،* E *یک مجموعه محدب است.*

قضیه ۵.۲. ([۳۲]): *اگر* $f(x)$ *بر مجموعه محدب* E *تابعی محدب باشد آنگاه* $f(x)$ *دارای یک کمینه موضعی*[۵] *است. اگر چنین کمینه‌ای یافت شود، یک کمینه سراسری*[۶] *است و بر روی مجموعه محدب به‌دست آورده می‌شود.*

تعریف ۱۰.۲. *یک ماتریس* $M(X)$ *در اندازه* $n \times n$ *که عناصر آن* m_{ij} $(i, j = ۱, ..., n)$ *توابعی روی* $E \subset \mathbb{R}^n$ *هستند،*

الف. نیمه معین مثبت[۷] *روی* E *نامیده می‌شود، اگر*

$$\forall\, V \in \mathbb{R}^n, V \neq ۰, X \in E\ :\ V^T M(X) V \geq ۰.$$

ب. معین مثبت[۸] *روی* E *نامیده می‌شود، اگر*

$$\forall\, V \in \mathbb{R}^n, V \neq ۰, X \in E\ :\ V^T M(X) V > ۰.$$

ج. معین مثبت قوی روی E *نامیم ، اگر عددی مثبت مانند* α *وجود داشته باشد به‌طوری که*

$$\forall\, V \in \mathbb{R}^n, X \in E\ :\ V^T M(X) V \geq \alpha \|V\|^۲.$$

تعاریف فوق را می‌توان بر اساس مفهوم مقدار ویژه نیز ارائه داد.

قضیه ۶.۲. [۳۶]*اگر* $\gamma(x)$ *کوچکترین مقدار ویژه ماتریس* $M(X)_{n\times n}$ *باشد آنگاه:*

الف. $M(X)_{n\times n}$ *روی* E *نیمه معین مثبت اگر و فقط اگر به ازای هر* $\gamma(x) \geq ۰, x \in E$

ب. $M(X)_{n\times n}$ *روی* E *معین مثبت اگر و فقط اگر به ازای هر* $\gamma(x) > ۰, x \in E$

ج. $M(X)_{n\times n}$ *روی* E *معین مثبت قوی اگر و فقط اگر به ازای هر* $\gamma(x) \geq \alpha \geq ۰, x \in E$

[۵] Local minimum

[۶] Global minimum

[۷] Positive semi-defnite

[۸] Positive defnite

شرایط لازم و کافی برای مسائل مقید

قضیه ۷.۲. (*شرایط لازم کروش-کان-تاکر* [۹] $(K.K.T)$، [۳۲]): *فرض کنید X یک مجموعه باز در $\mathbb{R}^n$ باشد. مسأله کمینه‌سازی مقید زیر را در نظر بگیرید:*

$$\begin{aligned} &\text{minimize} \qquad f(x) && (۳.۲)\\ &\text{subject to} \\ &\begin{cases} g_k(x) \le ۰, & k = ۱, ..., m, \\ h_j(x) = ۰, & j = ۱, ..., l, \\ x = (x_۱, x_۲, ..., x_n)^T \in X, \end{cases} && (۴.۲) \end{aligned}$$

که در آن $f : \mathbb{R}^n \to \mathbb{R}$ ، $g_k : \mathbb{R}^n \to \mathbb{R}$ و $h_j : \mathbb{R}^n \to \mathbb{R}$. فرض کنید $\hat{x}$ یک جواب شدنی این مسأله باشد و $K = \{k : g_k(\hat{x}) = ۰\}$. همچنین فرض کنید f و g_k برای $k \in K$ در $\hat{x}$ مشتق پذیر ، g_k برای $k \notin K$ در $\hat{x}$ پیوسته و h_j برای $(j = ۱, ..., l)$ دارای مشتق پیوسته باشند. به‌علاوه فرض کنید $\nabla g_k(\hat{x})$ برای $k \in K$ و $\nabla h_j(\hat{x})$ برای $(j = ۱, ..., l)$ مجموعاً مستقل‌خطی باشند. اگر $\hat{x}$ کمینه موضعی برای مساله (۳.۲)و (۴.۲)باشد، آنگاه اسکالرهای یکتای $u_k \ge ۰$ برای $k \in K$ و $v_j \ge ۰$ برای $(j = ۱, ..., l)$ موجود هستند به‌طوری که:

$$\begin{cases} \nabla f(\hat{x}) + \sum_{k \in K} \hat{u_k} \nabla g_k(\hat{x}) + \sum_{j=۱}^{l} \hat{v_j} \nabla h_j(\hat{x}) = ۰, \\ \hat{u_k} \ge ۰, \quad k \in K. \end{cases}$$

علاوه بر فرض‌های بالا اگر g_k برای $k \notin K$ در $\hat{x}$ مشتق‌پذیر باشد، آنگاه شرایط کروش-کان-تاکر می‌تواند به‌صورت زیر نوشته شود:

$$\begin{cases} \nabla f(\hat{x}) + \sum_{k=۱}^{m} \hat{u_k} \nabla g_k(\hat{x}) + \sum_{j=۱}^{l} \hat{v_j} \nabla h_j(\hat{x}) = ۰, \\ u_k g_k(\hat{x}) = ۰, \quad k = ۱, ..., m, \\ \hat{u_k} \ge ۰, \quad k = ۱, ..., m. \end{cases}$$

[۹] Karush-Kuhn-Tucker

همچنین شرایط کروش-کان-تاکر می‌تواند به‌فرم ماتریسی زیر نوشته شود:

$$\begin{cases} \nabla f(\hat{x}) + \nabla g(\hat{x})^T u + \nabla h(\hat{x})^T v = ۰, \\ u^T g(\hat{x}) = ۰, \\ u \geq ۰, \end{cases}$$

که در آن $\nabla g(\hat{x})$ ماتریس ژاکوبی $m \times n$ و $\nabla h(\hat{x})$ ماتریس ژاکوبی $l \times n$ است و $\hat{u}$ یک بردار m تایی و $\hat{v}$ یک بردار l تایی است. $(\hat{u}^T, \hat{v}^T)^T$ بردار ضرایب لاگرانژ نامیده شده است.

قضیه ۸.۲. (شرایط کافی کروش-کان-تاکر $(K.K.T)$، [۳۲]): فرض کنید X یک مجموعه باز در $\mathbb{R}^n$ باشد، مسأله (۳.۲) و (۴.۲) را در نظر بگیرید. فرض کنید $\hat{x}$ یک جواب شدنی باشد و $K = \{k : g_k(\hat{x}) = ۰\}$ ، اگر شرایط کروش-کان-تاکر در $\hat{x}$ برقرار باشد، یعنی اسکالرهای $u_k \geq ۰$ برای $k \in K$ و $v_j \geq ۰$ برای $(j = ۱, ..., l)$ موجود باشند به‌طوری که ،

$$\nabla f(\hat{x}) + \sum_{k \in K} \hat{u}_k \nabla g_k(\hat{x}) + \sum_{j=۱}^{l} \hat{v}_j \nabla h_j(\hat{x}) = ۰.$$

آنگاه $\hat{x}$ یک کمینه موضعی برای (۳.۲) و (۴.۲) خواهد بود.

قضیه ۹.۲. ([۳۶]):اگر در مساله (۳.۲) و (۴.۲)، g_k ها $(k = ۱, \cdots, m)$ محدب و h_j ها $(j = ۱, \cdots, l)$ آفینی باشند، شرایط کروش-کان-تاکر لازم و کافی هستند.

تعریف ۱۱.۲. ([۳۶]): مساله برنامه‌ریزی درجه دوم[۱۰] در حالت کلی بصورت زیر بیان می‌شود:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & q(x) = \tfrac{۱}{۲} x^T Q x + x^T c \\ \text{subject to} \quad & \\ & a_i^T x_i = b_i, \qquad i \in \varepsilon, \\ & a_i^T x_i \geq b_i, \qquad i \in \eta. \end{aligned}$$

که در آن Q یک ماتریس متقارن $n \times n$ است. ε و η مجموعه متناهی از اندیس‌ها هستند و c ، x و $\{a_i\}$ $i \in \varepsilon \cup \eta$ بردارهایی در R^n هستند. برنامه‌ریزی درجه دو همیشه می تواند قابل حل بودن آن (یا نشدنی بودن آن) در تعدادی متناهی محاسبات نشان داده شود. اگر ماتریس Q

[۱۰] Quadratic programming

نیمه‌معین‌مثبت باشد آنگاه مساله (QP)محدب است و اگر ماتریس Q نامعین باشد آنگاه مساله کمی چالش برانگیز خواهد شد، چون مساله می‌تواند چندین نقطه مینیمم موضعی داشته باشد.

۳.۲ توابع جریمه‌ای

مساله زیر را در نظر بگیرید:

$$\text{minimize} \qquad f(x) \tag{۵.۲}$$

$$\text{subject to}$$

$$x \in S. \tag{۶.۲}$$

که در آن f تابعی پیوسته بر $\mathbb{R}^n$و S یک مجموعه قیدی در $\mathbb{R}^n$ است. روش تابع جریمه‌ای مبتنی بر این است که به جای مساله(۵.۲) و (۶.۲) یک مساله نامقید به شکل زیر در نظر می‌گیریم.:

$$\text{minimize} \qquad f(x) + cP(x) \tag{۷.۲}$$

که در آن c یک ثابت مثبت و P تابعی بر $\mathbb{R}^n$ با خواص زیر است:

۱- Pپیوسته است

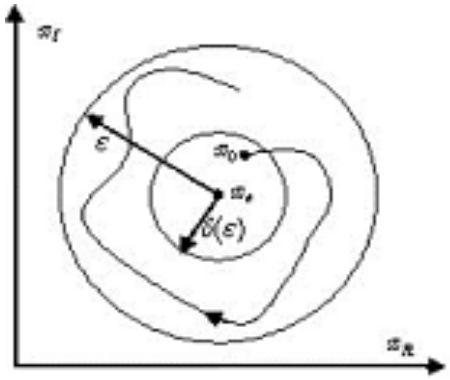
۲- به ازای هر $x \in \mathbb{R}^n$ ، $P(x) \geq ۰$.

۳- $P(x) = ۰$ اگر و فقط اگر $x \in S$.

۴.۲ پایداری

سیستم دینامیکي زیر را در نظر بگیرید:

$$\begin{cases} \dot{x} = f(x(t)), \\ x(t_۰) = x_۰ \in \mathbb{R}^n. \end{cases} \tag{۸.۲}$$



شکل ۱.۲: پایداری لیاپانوف

که در آن $f : \mathbb{R}^n \to \mathbb{R}^n$ است. x^* یک نقطهٔ تعادل [۱۱] برای ۸.۲ نامیده می‌شود اگر $f(x^*) = ۰$.

تعریف ۱۲.۲. *فرض کنید که $x(t)$ یک جواب ۸.۲ باشد، نقطهٔ تعادل تنها x^*، پایدار به مفهوم لیاپانوف [۱۲] است با توجه به شکل ۱.۲ ، اگر برای هر $x(t_۰) = x_۰$ و هر $\varepsilon > ۰$ ، یک $\delta > ۰$ وجود داشته باشد به‌طوری که اگر $\| x(t) - x_۰ \| \leq \delta$ آنگاه*

$$\forall t \geq t_۰, \quad \| x(t) - x^* \| < \varepsilon .$$

تعریف ۱۳.۲. *سیستم دینامیکی (۸.۲) همگرای سراسری [۱۳] به مجموعه*

$$\Omega^* = \{x \mid \text{جواب دارد (۸.۲) سیستم}\}.$$

گفته می‌شود اگر هر جواب دلخواه $x(t)$ از سیستم در رابطه زیر صدق کند:

$$\lim_{t \to \infty} dist(x(t), \Omega^*) = ۰.$$

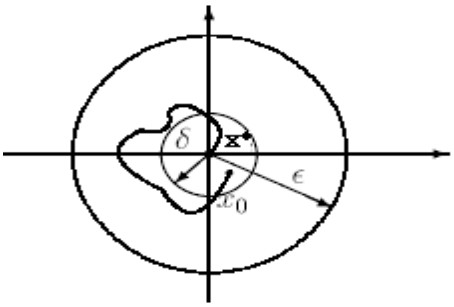
که در آن

$$dist(x, \Omega^*) = \inf_{y \in \Omega^*} \| x - y \| .$$

[۱۱] Equilibrium point

[۱۲] Lyapunov

[۱۳] Global convergance



شکل ۲.۲: پایداری مجانبی

تعریف ۱۴.۲. *سیستم دینامیکی در نقطهٔ تعادل یکتا x^* پایدار مجانبی سراسری [۱۴] نامیده می‌شود اگر x^* به مفهوم لیاپانوف پایدار باشد و در شرط زیر صدق کند:*

$$\lim_{t\to\infty} x(t) = x^*.$$

شکل (۲.۲) را ببینید.

تعریف ۱۵.۲. *مجموعه $M \subseteq \mathbb{R}^n$ ، یک مجموعه پایدار [۱۵] نسبت به سیستم (۸.۲) گفته می‌شود اگر $x(t_\circ) \in M$ و $t_\circ \geq \circ$، آنگاه به ازای هر $t \geq t_\circ$، $x(t) \in M$ باشد.*

تعریف ۱۶.۲. *نگاشت F در شرط لیپ شیتز [۱۶] صدق می‌کند اگر عدد ثابت L وجود داشته باشد، به‌طوری که برای هر دو نقطه $x, y \in \mathbb{R}^n$ داشته باشیم:*

$$\| F(x) - F(y) \| \leq L \| x - y \| .$$

F را لیپ شیتز محلی روی $\mathbb{R}^n$ نامیم اگر به ازای هر نقطه از $\mathbb{R}$، یک همسایگی مانند $D_\circ \subset \mathbb{R}^n$ وجود داشته باشد به‌طوری که نامساوی بالا برای هر دو نقطه $x, y \in D_\circ$ برقرار باشد.

تعریف ۱۷.۲. *نگاشت $F : \mathbb{R}^n \to \mathbb{R}^n$ یکنوا [۱۷] گفته می‌شود، اگر برای هر زوج از نقاط $x, y \in \mathbb{R}^n$ داشته باشیم:*

$$(x - y)^T (F(x) - F(y)) \geq \circ .$$

---

[۱۴] Globally asympotically stable

[۱۵] Invariant set

[۱۶] Lipschitz

[۱۷] Monotone

همچنین روی $\mathbb{R}^n$ اکیداً یکنوا [۱۸] است، اگر نامساوی فوق به‌صورت اکید برای هر $x \neq y$ برقرار باشد. و F روی $\mathbb{R}^n$ یکنوای قوی است اگر برای هر زوج از نقاط $x, y \in \mathbb{R}^n$ ثابت $\beta > 0$ وجود داشته باشد، به‌طوری که:

$$(x-y)^T(F(x)-F(y)) \geq \beta \parallel x-y \parallel^2 .$$

قضیه ۱۰.۲. ([۷]): *فرض می‌کنیم که $F : K \subset \mathbb{R}^n \to \mathbb{R}^n$ ، یک تابع به‌طور پیوسته مشتق پذیر روی K باشد و ماتریس ژاکوبین F که لزوماً متقارن نیست، نیمه معین مثبت (معین مثبت) باشد، آنگاه $F(x)$ یکنوا (یکنوای قوی) است.*

قضیه ۱۱.۲. ([۷]): *در سیستم دینامیکی (۸.۲) فرض می‌کنیم که $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ یک تابع پیوسته، آنگاه برای هر $t_0 > 0$ و $x_0 \in \mathbb{R}^n$ ، یک جواب محلی [۱۹] $x(t)$ به ازای $t \in [t_0, \tau)$ که $\tau > t_0$ وجود دارد. علاوه بر این اگر f در x_0 در شرط لیپ شیتز محلی صدق کند آنگاه جواب یکتا خواهد بود و اگر f در $\mathbb{R}^n$ در شرط لیپ شیتز صدق کند آنگاه τ می‌تواند تا $+\infty$ توسعه داده شود.*

تذکر:

همیشه می‌توان نقطهٔ تعادل را به‌صورت $x^* = 0$ در نظر گرفت. برای هر نقطهٔ تعادل دیگر می‌توان با استفاده از تغییرمتغیر، یک سیستم جدید با نقطهٔ تعادل y^* تعریف کرد. برای این منظور تعریف می‌کنیم:

$$y = x - x^* \Rightarrow \dot{y} = \dot{x} = f(x) \Longrightarrow f(x) = f(y + x^*) = g(y).$$

در نتیجه نقطه تعادل سیستم جدید $\dot{y} = g(y), y^* = 0$ است زیرا

$$g(0) = f(0 + x^*) = f(x^*) = 0.$$

در نتیجه x^* نقطهٔ تعادل برای سیستم $\dot{x} = f(x)$ است اگر و فقط اگر $y = 0$ نقطهٔ تعادل سیستم $\dot{y} = g(y)$ باشد.

---

[۱۸] Strictly monotonic

[۱۹] Local solution

۵.۲ تابع انرژی

قضیه ۱۲.۲. ([۷]): *فرض کنید که $x = ۰$ یک نقطهٔ تعادل برای سیستم (۸.۲) و تابع $E : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ به‌طور پیوسته مشتق پذیر باشد اگر*

۱. $E(۰) = ۰$ ،

۲. *به ازای هر* $x \in \Omega \subseteq \mathbb{R}^n \setminus \{۰\}$ ، $E(X) > ۰$ ،

۳. *به ازای هر* $x \in \Omega \subseteq \mathbb{R}^n \setminus \{۰\}$ ، $\dot{E}(X) \leq ۰$ ،

آنگاه $x = ۰$ نقطهٔ پایداری سیستم خواهد بود و $E(x)$ را« تابع لیاپانوف» یا «تابع انرژی» برای سیستم (۸.۲) نامیم.

قضیه ۱۳.۲. : [۷]*اصل تغییرناپذیری (ناوردای) لازال* [۲۰] *: فرض می‌کنیم تابع $V : \mathbb{R}^n \to \mathbb{R}$ به‌طور پیوسته مشتق پذیر باشد، همچنین فرض می‌کنیم*

۱. $M \subseteq \mathbb{R}^n$ *یک مجموعه فشرده و پایدار نسبت به جواب سیستم (۸.۲) باشد،*

۲. *به ازای هر* $x \in M$ ، $\dot{V}(x) \leq ۰$،

۳. E *مجموعه همه نقاط M باشد به‌طوری که* $\dot{V}(x) = ۰$،

۴. E *بزرگترین مجموعه پایدار در N ، باشد*

آنگاه به ازای هر $x(t_۰) \in M$ که $t_۰ \geq ۰$ باشد، داریم:

$$\lim_{t\to\infty} dist(x(t), N) = ۰.$$

بیانی دیگر از اصل تغییر ناپذیری لازال:

قضیه ۱۴.۲. (*اصل تغییر ناپذیری لازال* ، [۲۲]). *سیستم دینامیکی $\dot{x} = f(x)$ را که $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ در نظر گرفته و فرض کنید $\Omega \subset D$ مجموعه‌ای فشرده و تغییرناپذیر مثبت باشد. همچنین، فرض کنید تابع $V : D \to \mathbb{R}$ بطور پیوسته مشتق‌پذیر بوده و روی Ω داشته باشیم $\dot{V}(x) \leq ۰$. فرض کنید E مجموعه نقاطی از D باشد که $\dot{V}(x) = ۰$ است. اکنون اگر M بزرگترین مجموعه تغییرناپذیرمثبت در E باشد، آنگاه هر جواب این سیستم دینامیکی با نقطه شروعی در مجموعه Ω، پایدار مجانبی وابسته به مجموعه M است.*

[۲۰] Lasalle's invariance principle

حال نتیجه مهم زیر را که مستقیما از اصل تغییرناپذیری لازال به‌دست می‌آید بیان می‌کنیم. این نتیجه قضیه پایداری لیاپانوف را کامل می‌کند.

نتیجه ۱۵.۲. ([۲۲]). *فرض کنید مبدأ نقطه تعادل سیستم دینامیکی $\dot{x} = f(x)$ باشد و تابع $V : D \to \mathbb{R}$ روی $D \subset \mathbb{R}^n$ پیوسته مشتق‌پذیر و شامل مبدأ باشد بطوریکه روی مجموعه D داشته باشیم $\dot{V}(x) \leq 0$. همچنین مجموعه $E = \{x \in D \mid \dot{V}(x) = 0\}$ را در نظر می‌گیریم. اگر E فقط شامل نقطه $x(t) \equiv 0$ باشد، آنگاه مبدأ به عنوان یک نقطه تعادل سیستم دینامیکی بالا، پایدار مجانبی خواهد بود.*

فصل ۳

مقدمه و تاریخچه‌ای بر شبکه‌های عصبی

۱.۳ مقدمه

در این فصل به بیان مقدمه‌ای از مدل‌های شبکه‌های عصبی می‌پردازیم. در ابتدا ساختار شبکه عصبی و مدل ریاضی یک سلول عصبی را بیان می‌کنیم و در ادامه به بیان تاریخچه‌ای از شبکه‌های عصبی در حل مسائل بهینه سازی می‌پردازیم. در انتها به بیان مدل‌های ارائه شده پیشین برای حل مسائل بهینه‌سازی خواهیم پرداخت.

می‌دانیم کامپیوترها می‌توانند بعضی از کارها که ما آنها را در مدت زمان قابل ملاحظه‌ای انجام می‌دهیم مانند انجام چهار عمل اصلی، در کمترین زمان انجام دهند و یا می‌توانند نام‌ها و آدرس‌ها را ماه‌ها بعد به‌درستی به‌یاد بیاورند. با این اوصاف آیا باز هم می‌توان گفت انسان‌ها از کامپیوتر ها باهوش‌ترند؟ و اینکه چرا کامپیوترها نمی‌توانند کارهایی را که ما انجام می‌دهیم، انجام دهند؟

اگر به درون مغز نگاه کنیم، بررسی اولیه‌ی ما چیزی جز مجموعه‌ای گره‌خورده از ماده‌ای خاکستری رنگ نشان نمی‌دهد. بررسی بیشتر روشن می‌کند که مغز از اجزایی ریز تشکیل شده است. لیکن این اجزاء به شیوه‌ای بی‌نهایت پیچیده، مرتب شده‌اند و هر جزء به هزاران جزء دیگر متصل است. شاید این تفاوت در شیوه‌ی ساختار، علت اصلی اختلاف بین مغز و کامپیوتر است[۲۱].کامپیوترها طوری طراحی شده‌اند که یک عمل را بعد از عمل دیگر با سرعت بسیار زیاد انجام دهند. لیکن مغز با تعداد اجزای بیشتر اما با سرعت کمتر کار می‌کند. در حالیکه سرعت عملیات در کامپیوترها به میلیون‌ها محاسبه در ثانیه بالغ می‌شود، سرعت عملیات در مغز تقریبا بیش‌تر از ده بار در ثانیه نمی‌باشد. لیکن مغز در یک لحظه با تعداد زیادی اجزاء به‌طور هم‌زمان کار

می‌کند، کاری که از عهده کامپیوتر بر نمی‌آید. مغز می‌تواند کامپیوتر را در مسائلی مانند دیدن و شنیدن که اعمال کاملاً موازیند و در آن‌ها داده‌های متضاد و متفاوت، هر کدام باعث اثرات و ظهور خاطرات متفاوتی در مغز می‌گردند شکست دهد.

نرون[۱] عنصر اصلی مغز است و به تنهایی مانند یک واحد پردازش منطقی عمل می‌کند. مغز تقریباً دارای $۱۰^{۱۰}$ واحد پایه نرون است و هر نرون تقریباً به $۱۰^{۴}$ نرون دیگر اتصال دارد. نرون‌ها دونوع هستند، نرون‌های داخلی مغز که در فاصله‌های حدود ۱۰۰ میکرون به یکدیگر متصل‌اند و نرون‌های خارجی که قسمت‌های مختلف مغز را به یکدیگر و مغز را به ماهیچه‌ها و اعضای حسی را به مغز متصل می‌کنند. نحوه‌ی عملیات نرون‌ها بسیار پیچیده است و هنوز در سطح میکروسکوپی چندان شناخته شده نیست. هر نرون ورودی‌های متعددی را پذیراست که با یکدیگر به طریقی جمع می‌شوند. اگر در یک لحظه تعداد ورودی‌های فعال نرون به حد کفایت برسد نرون نیز فعال شده و آتش می‌کند. در غیراین‌صورت نرون به صورت غیر فعال و آرام باقی می‌ماند.

بدنه‌ی نرون، سوما [۲] نامیده می‌شود. به سوما رشته‌های نامنظم طولانی متصل است که به آنها دندریت[۳] می‌گویند. قطر این رشته‌ها اغلب از یک میکرون نازک‌تر است و اشکال شاخه‌ای پیچیده‌ای دارند. شکل ظریف آن‌ها شبیه شاخه‌های درخت بدون برگ هستند که هر شاخه بارها و بارها به شاخه‌های نازک‌تر منشعب می‌شود.

دندریت‌ها نقش اتصالاتی را دارند که ورودی‌ها را به نرون‌ها می‌رسانند. یکی از عناصر متصل به هسته‌ی نرون، آکسون [۴] نامیده می‌شود. این عنصر برخلاف دندریت از نظر الکتریکی فعال است و بعنوان خروجی نرون عمل می‌کند. نمایشی از ویژگی‌های عمده‌ی نرون در شکل (۱.۳) آمده است. رشته‌ی آکسون در نقطه‌ی تماس معینی به نام سیناپس [۵] قطع می‌شود و در این مکان به دندریت سلول دیگر وصل می‌گردد. در واقع این تماس به صورت اتصال مستقیم نیست بلکه از طریق ماده‌ی شیمیایی موقتی صورت می‌گیرد. سیناپس پس از آنکه پتانسیل آن از طریق پتانسیل‌های فعالیت دریافتی از طریق آکسون به اندازه‌ی کافی افزایش یافت از خود ماده‌ی شیمیایی به نام منتقل‌کننده عصبی [۶] ترشح می‌کند. یادگیری‌ها قدرت اتصال‌های سیناپسی را افزایش می‌دهند. تصور می‌شود یادگیری هنگامی صورت می‌گیرد که شدت اتصال یک سلول و سلول دیگر در محل سیناپس‌ها

---

[۱] Neuron

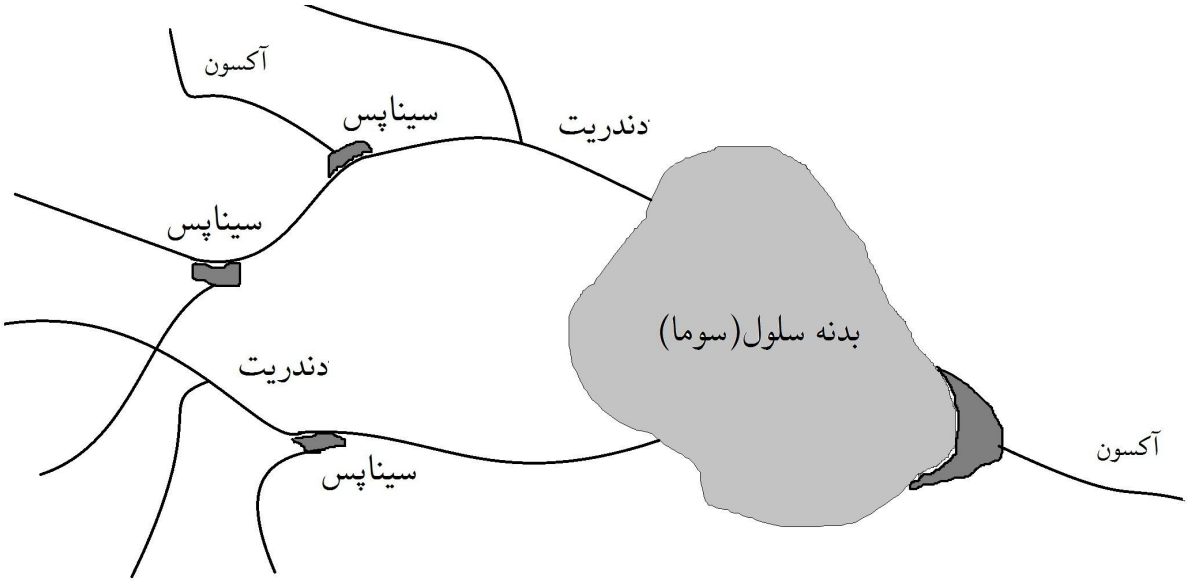
[۲] Soma

[۳] Dendrite

[۴] Axon

[۵] Synaps

[۶] Neurotransmitter

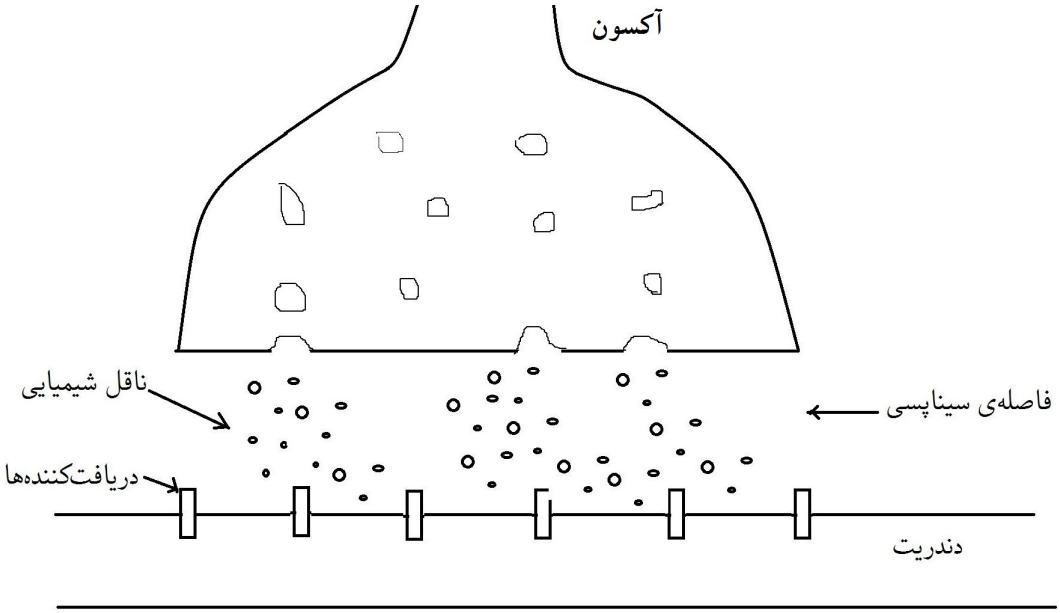


شکل ۱.۳: مشخصات اصلی یک نرون بیولوژیک.

اصلاح می‌گردد. شکل (۲.۳) ویژگی‌های مهم سیناپس را با جزئیات بیش‌تر نشان می‌دهد.

به‌نظر می‌رسد که این مقصود از طریق ایجاد سهولت بیش‌تر در میزان آزاد شدن ناقل شیمیایی حاصل می‌گردد. این حالت باعث می‌شود که دروازه‌های بیش‌تری روی دندریت‌های سمت مقابل باز شود و به این‌صورت باعث افزایش میزان اتصال دو سلول شود. تغییر میزان اتصال نرون‌ها به‌صورتی که باعث تقویت تماس‌های مطلوب شود از مشخصه‌های مهم در مدل‌های شبکه‌های عصبی است.

گفتیم مغز را می‌توان بصورت مجموعه‌ای بسیار متصل و شبکه‌ای از عناصر پردازشی نسبتاً ساده درنظر گرفت. به مدلی نیاز داریم که بتواند ویژگی‌های مهم سیستم‌های عصبی را کسب کند. به‌این منظور که بتواند رفتار مشابهی را از خود بروز دهد. لیکن اگر بخواهیم این مدل، به اندازه‌ی کافی برای فهمیدن و به‌کارگیری، ساده باشد باید بسیاری از جزئیات را عمداً نادیده بگیریم. استخراج تعداد محدودی ویژگی‌های مهم و نادیده‌گرفتن بقیه‌ی ویژگی‌ها از ضروریت‌های معمولی مدل‌سازی است. هدف مدل‌سازی اصولاً ایجاد نمونه‌ی ساده‌تری از سیستم است که رفتار عمومی سیستم را حفظ کرده و کمک کند که سیستم با سهولت بیش‌تر قابل درک باشد.



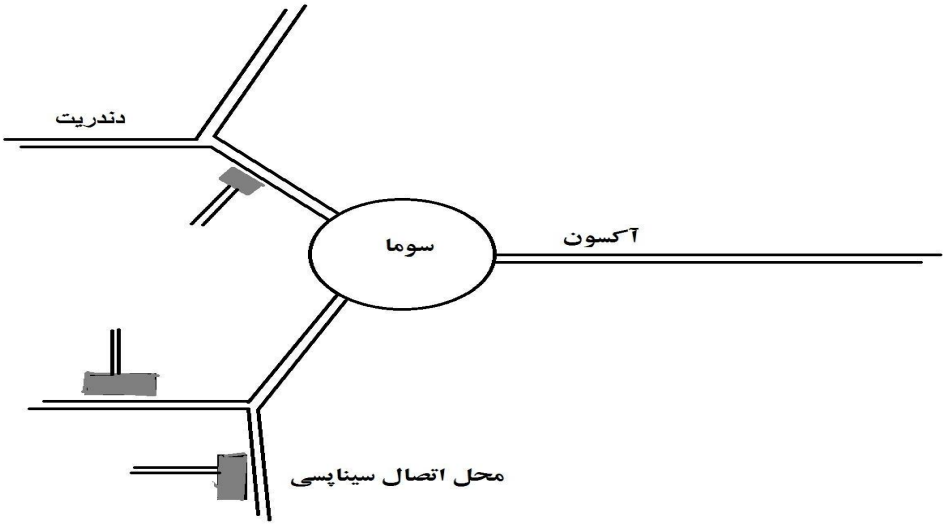
شکل ۲.۳: ناقل شیمیایی آزاد شده از شکاف سیناپس می‌گذرد ودریافت‌کننده‌های دندریت نرون دیگر را تحریک می‌کند.

۲.۳ مدل‌سازی نرون تنها

شبکه‌های عصبی مصنوعی از یک سری واحدهای ساختمانی اولیه تشکیل می‌شوند. هر واحد دارای چندین ورودی است که با هم ترکیب شده و پس از انجام عملیات پردازش یک خروجی را به‌دست می‌دهند. این واحدهای اولیه به‌هم متصل هستند به‌طوری‌که خروجی هر واحد به عنوان ورودی واحدهای دیگر مورد استفاده قرار می‌گیرد. واحدهای ساختمانی را سلول عصبی، نرون یا گره نیز می‌نامند.

شبکه‌های عصبی اولیه تنها دو لایه (ورودی وخروجی) داشتند و به‌همین دلیل تنها در مسائل خطی مفید بودند. ساختارهای توسعه یافته امروزی شامل سه و یا تعداد بیشتری لایه هستند. رایج‌ترین ساختار شبکه، سه لایه‌ای است (لایه ورودی، یک لایه پنهان و لایه خروجی) و عموماً قادر به حل اکثر مسائل پیچیده است.

ابتدا مشخصات یک نرون تنها و نحوه‌ی مدلسازی آن را بررسی می‌کنیم. نقش اصلی یک نرون بیولوژیکی، عمل جمع ورودی‌های خود تا جایی است که مجموع ورودی‌ها از حدی که به آن آستانه



شکل ۳.۳: مشخصات اصلی یک نرون بیولوژیک

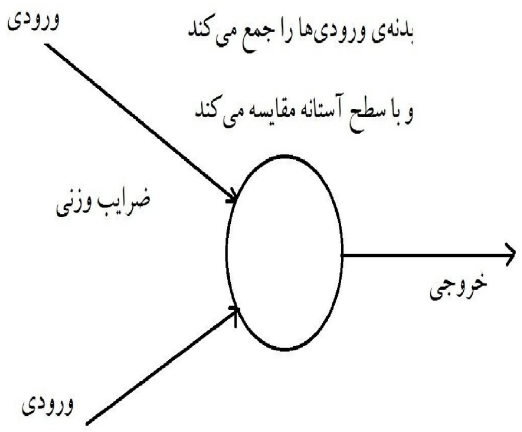
[۷] می‌گوییم تجاوز نکند و آنگاه تولید یک خروجی است. ورودی‌های نرون از طریق دندریت‌ها که به خروجی‌های نرون‌های دیگر توسط نقاط اتصال (سیناپس‌ها) متصل است وارد می‌شوند. سیناپس ها کارایی سیگنال‌های دریافتی را تغییر می‌دهند. بدنه‌ی سلولی کلیه‌ی ورودی‌ها را دریافت می‌کند و هنگامی که مجموع ورودی‌ها از حد آستانه تجاوز کرد سیگنالی را آتش می‌کند. این نرون بیولوژیکی ساده در شکل (۳.۳) نشان داده شده است.

مدلی که از نرون می‌سازیم باید مشخصه‌های زیر را داشته باشد. به‌طور خلاصه:

- خروجی یک نرون یا فعال است (یک) و یا غیرفعال است (صفر).
- خروجی تنها به ورودی‌ها بستگی دارد. میزان ورودی‌ها باید به حدی برسد که خروجی نرون را فعال سازد.

کارایی سیناپس‌ها در انتقال سیگنال‌های ورودی به بدنه‌ی سلول را می‌توان با استفاده از ضریبی که در ورودی‌های نرون ضرب می‌شود مدل‌سازی کرد. سیناپس‌های قوی‌تر که سیگنال بیشتری را منتقل می‌کنند دارای ضریب‌های بسیار بزرگتری هستند درحالی که سیناپس‌های ضعیف ضریب‌های

[۷] Threshold



شکل ۴.۳: نمای مدل اصلی نرون

کوچک‌تری دارند. بدین صورت مدل ما به‌صورتی خواهد بود که در شکل (۴.۳) آمده است. این مدل ابتدا مجموع وزنی ورودی‌های خود را محاسبه کرده سپس آن را با سطح آستانه‌ی داخلی خود مقایسه می‌کند و چنانچه از آن تجاوز کرد فعال می‌شود. درغیر اینصورت غیرفعال باقی می‌ماند. چون ورودی‌ها برای تولید خروجی از میان نرون عبور می‌کنند به این سیستم پیش‌خور [۸] می‌گوییم.

این عمل را باید به طریق ریاضی نشان دهیم. اگر تعداد ورودی‌ها n باشد آنگاه هر خط ورودی دارای یک ضریب وزنی مربوط به خود است. نرون مدل‌سازی شده ورودی خود را محاسبه می‌کند. ابتدا اولین ورودی را در ضریب وزنی مربوط به خط ارتباطی آن ورودی ضرب می‌کند.

۳.۳ شبکه های عصبی در یک نگاه

شبکه‌های عصبی نقاط ضعف و قوت بسیاری دارند و اگر بخواهیم از آنها بطور صحیح استفاده کنیم باید این نقاط را مد نظر قرار دهیم. اگرچه شبکه‌های عصبی اغلب به نام کامپیوترهای عصبی خوانده می‌شوند ولی در واقع آنها کامپیوتر نیستند بلکه اساساً حافظه‌هایی هستند که نتایج را حفظ می‌کنند. به عنوان مثال وقتی یک شخص این واقعیت را حفظ می‌کند که حاصلضرب چهار در شش، بیست و چهار است، این واقعیت برای همیشه در حافظه او باقی می‌ماند. در حالی که ارزانترین ماشین

[۸] Feed forward

حساب‌های رقمی هر بار که اعداد به آنها داده شود باید حاصلضرب را محاسبه کنند.
شبکه‌های عصبی اطلاعات را به روش مبتنی بر حافظه ذخیره می‌کنند که با ذخیره اطلاعات در یک جدول جستجو، متفاوت بوده و از آن انعطاف‌پذیرتر است. در یک شبکه عصبی درست مانند مغز، اطلاعات در همه جای شبکه توزیع می‌شوند.بنابراین در این روش، اطلاعات در قسمت‌های مختلف شبکه و به طور جداگانه ذخیره می‌شوند و اگرچه این کار دشواری است ولی این قابلیت بسیار مهم را در شبکه‌های عصبی بوجود می‌آورد که بتوانند نتایج را تعمیم دهند. علاوه بر آن، کمبود تعداد کمی از اعصاب(چه مصنوعی و چه واقعی) اثر چندانی بر اطلاعات ذخیره شده نخواهد گذاشت. کار بر روی شبکه‌های عصبی مصنوعی، با الهام از عملکرد مغز انسان از آنجا شروع شد که دانشمندان دریافتند مغز بشر به‌گونه‌ای کاملاً متفاوت از کامپیوترهای دیجیتالی مرسوم محاسبات را انجام می‌دهد. مغز یک سیستم پردازش داده بسیار پیچیده است که از واحدهای ساختاری به نام سلول عصبی یا نرون تشکیل شده است. شبکه عصبی سیستمی پویا و غیرخطی است که از تعداد زیادی واحد پردازش (نرون ها) و اتصالات بین این واحدهای پردازش تشکیل می‌شود.
شبکه‌های عصبی برای حل مسائلی به کار می‌رود که فرمول حل آنها ناشناخته است و مدل علت ومعلولی یا برای آنها وجود ندارد و یا ابهام قابل ملاحظه‌ای در آن دیده می‌شود. علت نبود روابط ریاضی لازم برای تشریح چنین مسائلی این است که حتی خود مسأله به‌طور کامل و بدون ابهام شناخته نشده است.
یک شبکه عصبی بر خلاف کامپیوتر که نیازمند دستورهای کاملاً صریح و مشخص است، به مدل‌های ریاضی محض نیازی ندارد، بلکه مانند انسان تجربه کسب کرده و سپس نتیجه این تجربیات را تعمیم می‌دهد. امروزه این شبکه‌ها تبدیل به ابزاری قدرتمند و عمومی شده‌اند که برای حل مسائل از قبیل تخمین (تقریب)، تشخیص الگو[۹] ، رباتیک، کنترل و به‌طور کلی در هر جا که نیاز به یادگیری نگاشت خطی و یا غیرخطی باشد، به‌کار می‌روند[۱۹] .

۱.۳.۳ شبکه عصبی مصنوعی محاسبه ای

شبکه عصبی روشی برای محاسبه است که بر پایه اتصال و به هم پیوسته چندین واحد پردازشی ساخته می شود. شبکه از تعداد دلخواهی سلول یا گره یا واحد یا نرون تشکیل می‌شود که مجموعه ورودی را به خروجی ربط می دهند شبکه عصبی مصنوعی روشی عملی برای یادگیری توابع گوناگون نظیر توابع با مقادیر حقیقی، توابع با مقادیر گسسته و توابع با مقادیر برداری میباشد. یادگیری شبکه

[۹] Pattern recognition

عصبی در برابر خطاهای داده های آموزشی مصون بوده و اینگونه شبکه ها با موفقیت به مسائلی نظیر شناسائی گفتار، شناسائی و تعبیر تصاویر، و یادگیری روبات اعمال شده است.همچنین شبکه های عصبی قابلیت محاسبه یک تابع معلوم ،تقریب یک تابع ناشناخته شناسائی الگو [۱۰] ،پردازش سیگنال [۱۱] و یادگیری را دارا هستند.

۲.۳.۳ ایده ی اصلی عملکرد شبکه های عصبی مصنوعی

هر گره دارای دو وضعیت فعال و غیرفعال است (صفر یا یك) و هر یال نیز دارای یك وزن می‌باشد . یال‌های با وزن مثبت بین دو گره تا گره فعال دیگری را تحریك می‌کنند و یال‌های با وزن منفی بین دو گره، گره فعال دیگری را غیر فعال می‌سازند. نحوه عملکرد شبکه بدین صورت است که ابتدا یك گره به تصادف انتخاب می‌شود. اگر یك یا بیشتر از همسایه‌های آن گره فعال بودند جمع وزن‌دار یال‌های منتهی به آن گره‌ها حساب می‌شود. اگر این جمع مثبت بود گره فعال می‌شود و در غیر این صورت گره مذکور غیرفعال باقی خواهد ماند. سپس مجددا یك گره دیگر به تصادف انتخاب شده و همین عملیات آنقدر تکرار می‌شود تا شبکه به یك حالت پایدار برسد.

تز اصلی هاپفیلد : از هر حالت ابتدایی و با هر وزنی از یال‌ها که شروع کنیم، شبکه در نهایت به حالت پایدار خواهد رسید.

۴.۳ مدل ریاضی یک سلول عصبی

شکل (۵.۳) یک مدل سلول عصبی را نشان می‌دهد. بدنه این سلول از دو بخش تشکیل شده است اولین بخش را تابع ترکیب یا انتشار [۱۲] می‌گویند. وظیفه تابع ترکیب این است که تمام ورودی‌ها را ترکیب و یک عدد تولید کند. مطابق شکل (۵.۳) هر ورودی دارای وزن مختص به خود است. ورودی‌ها در وزن‌های مربوطه ضرب و سپس با هم جمع می‌شوند. مجموع حاصل را مجموع موزون [۱۳] گویند که رایج ترین تابع ترکیب است.

بخش دوم سلول عصبی، تابع انتقال [۱۴] نام دارد که تابع ترکیب را به خروجی سلول تبدیل و سپس

---

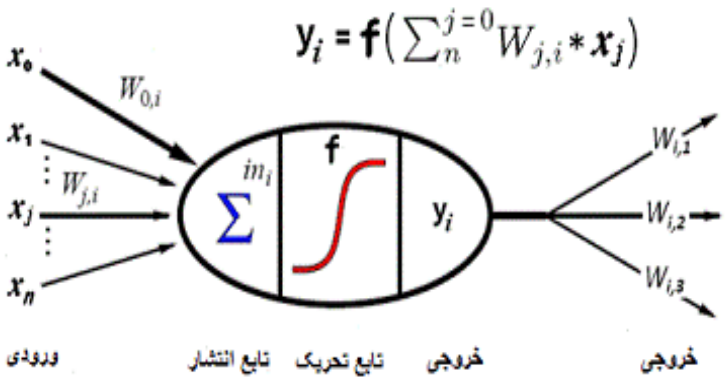
[۱۰] Pattern recognition

[۱۱] Signal processing

[۱۲] Combination function

[۱۳] Weighted sum

[۱۴] Transfer function



شکل ۵.۳: مدل ساده یک سلول عصبی

انتقال می‌دهد. تابع انتقال را تابع تحریک [۱۵] نیز می‌نامند.

رایج‌ترین توابع تحریک، بر پایه مدل‌های بیولوژیک استوار گردیده است (برای توضیحات بیشتر مراجع [۱۹]را ببینید). اگر ورودی سلول i ام واقع در لایه ورودی را x_i در نظر بگیریم، این ورودی برای اتصال به سلول j ام لایه بعد در وزن w_{ji} ضرب می‌شود. حرف j در w_{ji} نشان دهنده شماره سلول در لایه بعد و حرف دوم اندیس معرف شماره سلول لایه قبلی است. شکل (۶.۳) مدل بلوکی یک سلول عصبی را نشان می‌دهد سلول مذکور دارای یک ورودی اضافی است که به آن اریبی [۱۶] می‌گویند و با b_j نشان داده می‌شود. نقش اریبی افزایش یا کاهش مجموع موزون است.

به بیان ریاضی، سلول j بوسیله دو رابطه زیر تعریف می‌شود:

$$u_j = \sum_{i=۱}^{m} w_{ji} x_i,$$

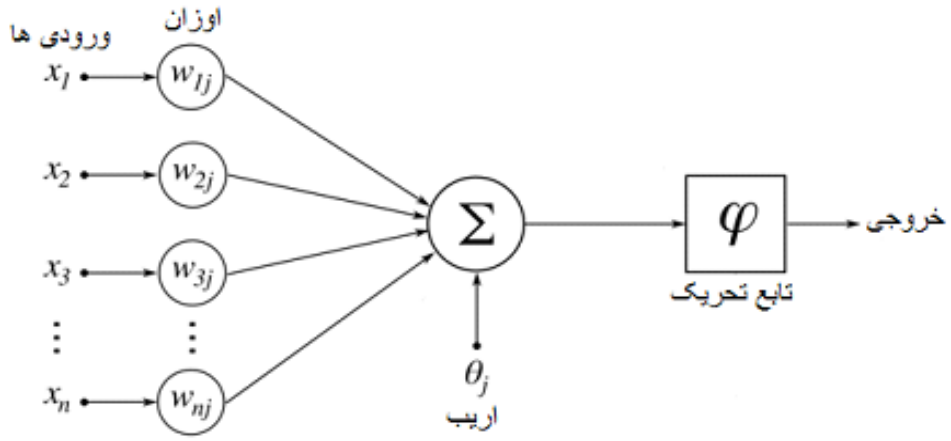
$$y_j = \phi(u_j + b_j),$$

که $x_۱, x_۲, ..., x_m$ داده‌های ورودی، $w_{j۱}, w_{j۲}, ..., w_{jm}$ وزن‌های اتصال ورودی‌های $m, \cdots, ۲, ۱$ به سلولj ، u_j ترکیب خطی ورودی‌ها، b_j اریبی، ϕ تابع تحریک و y_j خروجی سلول است. می‌توان مجموع u_j و b_j را به‌صورت زیر نمایش داد:

$$v_j = (u_j + b_j).$$

[۱۵] Activation function

[۱۶] Bias



شکل ۶.۳: مدل بلوکی سلول عصبی

تابع تحریک یا تابع انتقال که با ϕ نشان داده می‌شود بر روی مجموع وزن‌ها (شامل اریبی) عمل می‌کند. این تابع می‌تواند خطی یا غیرخطی باشد و بر اساس نیاز خاص حل یک مسأله انتخاب می‌شود. انواع مختلفی از توابع تحریک وجود دارند که مهم‌ترین آنها بشرح زیر می‌باشند:

۱. تابع حد آستانه [۱۷] : این تابع به‌صورت زیر بیان می‌شود

$$\phi(\nu) = \begin{cases} ۱, & \nu \geq ۰, \\ ۰, & \nu < ۰. \end{cases}$$

۲. تابع قطعه‌ای [۱۸] : این تابع به‌صورت زیر تعریف می‌شود:

$$\phi(\nu) = \begin{cases} ۱, & \nu \geq \frac{۱}{۲}, \\ \nu, & -\frac{۱}{۲} < \nu < \frac{۱}{۲}, \\ ۰, & \nu \leq -\frac{۱}{۲}. \end{cases}$$

۳. تابع سیگموئید[۱۹] : نام این تابع به علت شکل S مانند آن است و پرکاربردترین تابع تحریک

[۱۷] Threshold function

[۱۸] Piecewise-Linear function

[۱۹] Sigmoid function

محسوب می‌شود. این تابع بین رفتار خطی و غیرخطی، به آرامی تغییر می‌کند. مثالی از تابع سیگموئید تابع لجستیک [۲۰] است که به‌صورت زیر بیان می‌شود:

$$\phi(\nu) = \frac{۱}{۱ + e^{-a\nu}}.$$

که در آن a پارامتر شیب تابع سیگموئید است، با تغییر پارامترa ، توابع مختلفی با شیب‌های متفاوت بدست می‌آید.

۵.۳ اهداف و انگیزه‌ها

هدف یادگیری ماشینی این است که کامپیوتر (در کلی‌ترین مفهوم آن) بتواند به تدریج و با افزایش باده‌ها بازدهی بالاتری در وظیفه مورد نظر پیدا کند. گسترده این وظیفه می‌تواند از تشخیص خودکار چهره با دیدن چند نمونه از چهره مورد نظر تا فراگیری شیوه گام‌برداری برای روبات‌های دو پا با دریافت سیگنال پاداش و تنبیه باشد. طیف پژوهش‌هایی که در یادگیری ماشینی می‌شود گسترده‌است. در سوی نظری آن پژوهش‌گران بر آن‌اند که روش‌های یادگیری تازه‌ای به وجود بیاورند و امکان‌پذیری و کیفیت یادگیری را برای روش‌های‌شان مطالعه کنند و در سوی دیگر عده‌ای از پژوهش‌گران سعی می‌کنند روش‌های یادگیری ماشینی را بر مسایل تازه‌ای اعمال کنند. البته این طیف گسسته نیست و پژوهش‌های انجام‌شده دارای مولفه‌هایی از هر دو رویکرد هستند.

۶.۳ تقسیم‌بندی مسایل

یکی از تقسیم‌بندی‌های متداول در یادگیری ماشینی، تقسیم‌بندی بر اساس نوع داده‌های در اختیار عامل هوش‌مند است. به سناریوی زیر توجه کنید: فرض کنید به تازگی رباتی سگ‌نما خریده‌اید که می‌تواند توسط دوربینی دنیای خارج را مشاهده کند، به کمک میکروفن‌هایش صداها را بشنود، با بلندگوهایی با شما سخن بگوید (گیریم محدود) و چهارپایش را حرکت دهد. هم‌چنین در جعبهٔ این ربات دستگاه کنترل از راه دوری وجود دارد که می‌توانید انواع مختلف دستورها را به ربات بدهید. در پاراگراف‌های آینده با بعضی از نمونه‌های این دستورات آشنا خواهید شد. اولین کاری که می‌خواهید بکنید این است که اگر ربات شما را دید خرناسه بکشد اما اگر غریبه‌ای را مشاهده کرد با صدای بلند عوعو کند. فعلاً فرض می‌کنیم که ربات توانایی تولید آن صداها را دارد اما

[۲۰] Logistic function

هنوز چهرهٔ شما را یاد نگرفته‌است. پس کاری که می‌کنید این است که جلوی چشم‌های‌اش قرار می‌گیرید و به کمک کنترل از راه دورتان به او دستور می‌دهید که چهره‌ای که جلوی‌اش می‌بیند را با خرناسه‌کشیدن مربوط کند. این‌کار را برای چند زاویهٔ مختلف از صورت‌تان انجام می‌دهید تا مطمئن باشید که ربات در صورتی که شما را از مثلاً نیم‌رخ ببیند به‌تان عوعو نکند. هم‌چنین شما چند چهرهٔ غریبه نیز به او نشان می‌دهید و چهرهٔ غریبه را با دستور عوعوکردن مشخص می‌کنید. در این حالت شما به کامپیوتر ربات گفته‌اید که چه ورودی‌ای را به چه خروجی‌ای مربوط کند. دقت کنید که هم ورودی و هم خروجی مشخص است و در اصطلاح خروجی برچسب‌دار است. به این شیوهٔ یادگیری، یادگیری بانظارت [۲] می‌گویند. اینک حالت دیگری را فرض کنید. برخلاف دفعهٔ پیشین که به ربات‌تان می‌گفتید چه محرک‌ای را به چه خروجی‌ای ربط دهد، این‌بار می‌خواهید ربات خودش چنین چیزی را یاد بگیرد. به این صورت که اگر شما را دید و خرناسه کشید به نحوی به او پاداش دهید (مثلاً به کمک همان کنترل از راه دورتان) و اگر به اشتباه به شما عوعو کرد، او را تنبیه کنید (باز هم با همان کنترل از راه دورتان). در این حالت به ربات نمی‌گویید به ازای هر شرایطی چه کاری مناسب است، بلکه اجازه می‌دهید ربات خود کاوش کند و تنها شما نتیجهٔ نهایی را تشویق یا تنبیه می‌کنید. به این شیوهٔ یادگیری، یادگیری تقویتی [۳] می‌گویند. در دو حالت پیش قرار بود ربات ورودی‌ای را به خروجی‌ای مرتبط کند. اما گاهی وقت‌ها تنها می‌خواهیم ربات بتواند تشخیص دهد که آن‌چه می‌بیند (یا می‌شنود و...) را به نوعی به آن‌چه پیش‌تر دیده‌است ربط دهد بدون این‌که به طور مشخص بداند آن‌چیزی که دیده شده‌است چه چیزی است یا این‌که چه کاری در موقع دیدن‌اش باید انجام دهد. ربات هوش‌مند شما باید بتواند بین صندلی و انسان تفاوت قایل شود بی‌آنکه به او بگوییم این نمونه‌ها صندلی‌اند و آن نمونه‌های دیگر انسان. در این‌جا برخلاف یادگیری بانظارت هدف ارتباط ورودی و خروجی نیست، بلکه تنها دسته‌بندی‌ی آن‌ها است. این نوع یادگیری که به آن یادگیری بی نظارت می گویند بسیار مهم است چون دنیای ربات پر از ورودی‌هایی است که کسی برچسبی به آن‌ها اختصاص نداده اما به وضوح جزیی از یک دسته هستند. یادگیری بی‌نظارت را می‌توان به صورت عمل کاهش بعد در نظر گرفت. از آن‌جا که شما سرتان شلوغ است، در نتیجه در روز فقط می‌توانید مدت محدودی با ربات‌تان بازی کنید و به او چیزها را نشان دهید و نام‌شان را بگویید (برچسب‌گذاری کنید). اما ربات در طول روز روشن است و داده‌های بسیاری را دریافت می‌کند. در این‌جا ربات می‌تواند هم به خودی‌ی خود و بدون نظارت یاد بگیرد و هم این‌که هنگامی که شما او را راه‌نمایی می‌کنید، سعی کند از آن تجارب شخصی‌اش استفاده کند و از آموزش شما بهره بیش‌تری ببرد. ترکیبی که عامل هوش‌مند هم از داده‌های بدون برچسب و هم از داده‌های با برچسب استفاده می‌کند یادگیری نیمه نظارتی می‌گویند.

۷.۳ یادگیری با نظارت و تقویتی

یادگیری تحت نظارت، یک روش عمومی در یادگیری ماشین است که در آن به یک سیستم، مجموعه ای از جفت‌های ورودی — خروجی ارائه شده و سیستم تلاش می‌کند تا تابعی از ورودی به خروجی را فرا گیرد. یادگیری تحت نظارت نیازمند تعدادی داده ورودی به منظور آموزش سیستم است. با این حال رده‌ای از مسائل وجود دارند که خروجی مناسب که یک سیستم یادگیری تحت نظارت نیازمند آن است، برای آن‌ها موجود نیست. این نوع از مسائل چندان قابل جوابگویی با استفاده از یادگیری تحت نظارت نیستند. یادگیری تقویتی مدلی برای مسائلی از این قبیل فراهم می‌آورد. در یادگیری تقویتی[۷]، سیستم تلاش می‌کند تا تقابلات خود با یک محیط پویا را از طریق آزمون و خطا بهینه نماید. یادگیری تقویتی مسئله‌ای است که یک عامل که می‌بایست رفتار خود را از طریق تعاملات آزمون و خطا با یک محیط پویا فرا گیرد، با آن مواجه است. در یادگیری تقویتی هیچ نوع زوج ورودی- خروجی ارائه نمی‌شود. به جای آن، پس از اتخاذ یک عمل، حالت بعدی و پاداش بلافصل به عامل ارائه می‌شود. هدف اولیه برنامه‌ریزی عامل‌ها با استفاده از تنبیه و تشویق است بدون آنکه ذکری از چگونگی انجام وظیفه آن‌ها شود.

۱.۷.۳ شبکه های عصبی در مقابل کامپیوترهای معمولی

شبکه های عصبی نسبت به کامپیوتر های معمولی مسیر متفاوتی را برای حل مسئله طی می کنند. کامپیوتر های معمولی یک مسیر الگو ریتمی را استفاده می کنند به این معنی که کامپیوتر یک مجموعه از دستورالعمل ها را به قصد حل مسئله پی می گیرد. بدون اینکه، قدم های مخصوصی که کامپیوتر نیاز به طی کردن دارد، شناخته شده باشند کامپیوتر قادر به حل مسئله نیست. این حقیقت قابلیت حل مسئله ی کامپیوتر های معمولی را به مسائلی ،محدود می کند که ما قادر به درک آنها هستیم و می دانیم چگونه حل میشوند. اما اگر کامپیوتر ها می توانستند کار هایی را انجام دهند که ما دقیقا نمیدانیم چگونه انجام دهیم ، خیلی پر فایده تر بودند.

شبکه های عصبی اطلاعات را به روشی مشابه با کاری که مغز انسان انجام می دهد پردازش می کنند. آنها از تعداد زیادی از عناصر پرداز شی(سلول عصبی) که فوق العاده بهم پیوسته اند تشکیل شده است که این عناصر به صورت موازی باهم برای حل یک مسئله مشخص کار می کنند. شبکه های عصبی با مثال کار می کنند و نمی توان آنها را برای انجام یک وظیفه خاص برنامه ریزی کرد مثال ها می بایست با دقت انتخاب شوند در غیر این صورت زمان سودمند، تلف می شود و یا حتی بدتر از این شبکه ممکن است نا درست کار کند. امتیاز شبکه عصبی این است که خودش کشف

می کند که چگونه مسئله را حل کند ، عملکرد آن غیر قابل پیش گویی است.
از طرف دیگر، کامپیوتر های معمولی از یک مسیر مشخص برای حل یک مساله استفاده می کنند. راه حلی که مساله از آن طریق حل می شود باید از قبل شناخته شود و به صورت دستورات کوتاه و غیر مبهمی شرح داده شود. این دستورات سپس به زبان های برنامه نویسی سطح بالا برگردانده می شود و بعد از آن به کدهایی که کامپیوتر قادر به درک آنها است تبدیل می شود. به طور کلی این ماشین ها قابل پیش گویی هستند و اگر چیزی به خطا انجام شود به یک اشتباه سخت افزاری یا نرم افزاری بر می گردد.

شبکه های عصبی و کامپیوتر های معمولی با هم در حال رقابت نیستند بلکه کامل کننده یکدیگرند.وظایفی وجود دارد که بیشتر مناسب روش های الگو ریتمی هستند نظیر عملیات محاسباتی و وظایفی نیز وجود دارد که بیشتر مناسب شبکه های عصبی هستند. حتی فراتر از این، مسایلی وجود دارد که نیازمند به سیستمی است که از تر کیب هر دو روش بدست می آید (بطور معمول کامپیوتر های معمولی برای نظارت بر شبکه های عصبی به کار گرفته می شوند) به این قصد که بیشترین کارایی بدست آید.

شبکه های عصبی معجزه نمی کنند اما اگر خردمندانه به کار گرفته شوند نتایج شگفت آوری را خلق می کنند .

تعریف ۱.۳. *پرسپترون[۲۱] : نوعی از شبکه عصبی برمبنای یک واحد محاسباتی به نام پرسپترون ساخته می شود. یک پرسپترون برداری از ورودیهای با مقادیر حقیقی را گرفته و یک ترکیب خطی از این ورودیها را محاسبه می کند. اگر حاصل از یک مقدار آستانه بیشتر بود خروجی پرسپترون برابر با ۱ و در غیر اینصورت معادل ۱- خواهد بود.*

تعریف ۲.۳. *قانون پرسپترون : برای یک مثال آموزشی $X = (x_1, x_2, \cdots, x_n)$ در هر مرحله وزنها بر اساس قانون پرسپترون بصورت زیر تغییر می‌کند:*

$$w_i = w_i + \Delta w_i.$$

که در آن

$$\Delta w_i = \eta(t - o)x_i.$$

t خروجی مشخص است و o خروجی تولید شده توسط پرسپترون و ηثابتی به نام نرخ یادگیری می‌باشد. اثبات شده است که برای یک مجموعه مثال جداپذیر خطی این روش همگرا شده و پرسپترون قادر به جدا سازی صحیح مثالها خواهد شد.

[۲۱] Perceptron

تعریف ۳.۳. *قانون دلتا [۲۲] : الگوریتم یادگیری با استفاده از قانون دلتا بصورت زیر می باشد. به وزنها مقدار تصادفی نسبت دهید تا رسیدن به شرایط توقف مراحل زیر را ادامه دهید هر وزن Δw_i را با مقدار صفر عدد دهی اولیه کنید.*

برای مثال: وزن Δw_i را بصورت زیر تغییر دهید:

$$\Delta w_i = \Delta w_i + \eta(t - o)x_i.$$

مقدار w_i را بصورت زیر تغییر داده و این روش را ادامه دهید:

$$w_i = w_i + \Delta w_i.$$

تا خطا بسیار کوچک شود .

تعریف ۴.۳. ***مدل ریاضی یک نرون:*** *نرون کوچکترین واحد یک شبکه عصبی مصنوعی است که عملکرد شبکه های عصبی را تشکیل می دهد. بدنه هر سلول عصبی از دو بخش تشکیل می شود، بخش اول را تابع ترکیب می گویند. وظیفه تابع ترکیب این است که تمام ورودی ها را ترکیب و یک عدد تولید می کند. در بخش دوم سلول تابع انتقال قرار دارد که به آن تابع تحریک نیز می گویند. درواقع همان گونه که یک سلول بیولوژیک باید به سطح آستانه تحریک خاصی برسد تا یک سیگنال تولید کند، توابع تحریک نیز تا زمانی که ورودی های ترکیب شده و وزن دار شده به یک حد آستانه ای خاص نرسند مقدار خروجی نظیر بسیار کوچکی تولید میکنند. وقتی ورودی های ترکیب شده به حد آستانه ای خاصی برسند، سلول عصبی تحریک شده و سیگنال خروجی تولید می کند. با مقایسه جواب خروجی شبکه با مقدار مطلوب مورد نظر بردار خطا محاسبه شده و این بردار با استفاده از الگوریتم های مختلف از آخر به سمت ابتدای شبکه پخش شده، به طوری که درسیکل بعد خطا کاهش یابد.*

تعریف ۵.۳. ***روشهای تکراری:*** *این روشها با توجه به سادگی و عدم حساسیتشان نسبت به خطای گرد کردن[۲۳] برای کاربرد با برنامه های کامپیوتری مناسبتر از روشهای مستقیم می باشند، زیرا در مقایسه با روشهای مستقیم حا فظه کمتری را اشغال می کنند و می توانند عملیات تکرار را تا رسیدن به دقت مورد نظر ادامه دهند .*

---

[۲۲] Delta rule

[۲۳] Round error

۸.۳ مبانی شبکه‌های عصبی مصنوعی[۲۴]

- شبکه های عصبی به طور کلی سیستمهای ریاضی یادگیر غیر خطی هستند. طرز کار این شبکه ها از روش کار مغز انسان الگو برداری شده است. در واقع شبکه های عصبی طبق تعریف ماشینی است برای ساخت یک مدل که می توان آن را بوسیله سخت افزار یا نرم افزار شبیه سازی کرد و عملکردی شبیه مغز انسان دارند.

- یک شبکه عصبی بر خلاف کامپیوترهای رقمی که نیازمند دستورات کاملا صریح و مشخص است، به مدل های ریاضی محض نیاز ندارد بلکه مانند انسان قابلیت یادگیری به وسیله تعدادی مثال مشخص را دارد.

- هر شبکه عصبی سه مرحله آموزش، اعتبار سنجی و اجرا را پشت سر می گذارد. در واقع شبکه های عصبی را می توان در حل مسایلی که روابط دقیق ریاضی بین ورودی ها و خروجی های آن برقرار نیست بکار برد.

- آموزش دادن شبکه های عصبی در واقع چیزی جز تنظیم وزن های ارتباطی این نرون ها به ازای دریافت مثال های مختلف نیست بطوریکه خروجی شبکه به سمت خروجی مطلوب همگرا شود.

۱.۸.۳ معایب شبکه‌های عصبی مصنوعی

با وجود برتری هایی که شبکه های عصبی نسبت به سیستم های مرسوم دارند، معایبی نیز دارند که پژوهشگران این رشته تلاش دارند که آن ها را به حداقل برسانند، از جمله، قواعد یا دستورات مشخصی برای طراحی شبکه جهت يك كاربرد اختیاری وجود ندارد. در مورد مسایل مدل‌سازی، نمی‌توان صرفاً با استفاده از شبکه عصبی به فیزیك مسأله پی برد. به عبارت دیگر مرتبط ساختن پارامترها یا ساختار شبکه به پارامترهای فرآیند معمولاً غیرممکن است. دقت نتایج بستگی زیادی به اندازه مجموعه آموزش دارد. آموزش شبکه ممکن است مشکل یا حتی غیرممکن باشد. پیش‌بینی عملکرد آینده شبکه (عمومیت یافتن) آن به سادگی امکان‌پذیر نیست.

[۲۴] Artificial neural network

۹.۳ مسایل مناسب برای یادگیری شبکه های عصبی

- خطا در داده های آموزشی (ورودی) وجود داشته باشد. مانند مسائلی که داده های آموزشی دارای نویز حاصل از داده های سنسورها نظیر دوربین و میکروفن ها هستند.
- مواردی که نمونه ها توسط مقادیر زیادی زوج ویژگی-مقدار نشان داده شده باشند. نظیر داده های حاصل از یک دوربین ویدئوئی RGB
- تابع هدف دارای مقادیر پیوسته باشد.
- زمان کافی برای یادگیری وجود داشته باشد. شبکه های عصبی با تعداد آموزش بیشتر ، جواب دقیق تری به ورودی ها خواهند داد.
- نیازی به تعبیر تابع هدف نباشد. زیرا به سختی میتوان وزنهای یادگرفته شده توسط شبکه را تعبیر نمود.

۱۰.۳ الهام از طبیعت

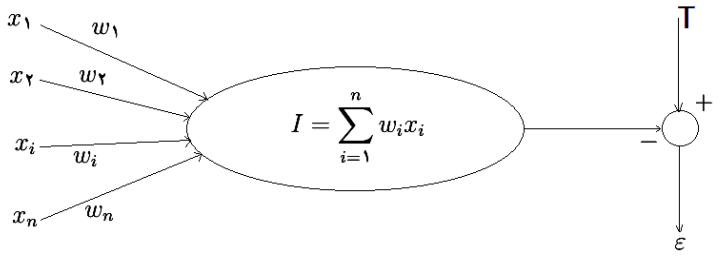
مطالعه شبکه های عصبی مصنوعی تا حد زیادی الهام گرفته از سیستم های یادگیر طبیعی است که در آنها یک مجموعه پیچیده از نرونهای به هم متصل در کار یادگیری دخیل هستند. گمان میرود که مغز انسان از تعداد $۱۰^{۱۱}$نرون تشکیل شده باشد که هر نرون با تقریبا $۱۰^{۴}$ نرون دیگر در ارتباط است. سرعت سوئیچنگ نرونها در حدود$۱۰^{-۳}$ ثانیه است که در مقایسه با کامپیوترها($۱۰^{-۱۰}$ ثانیه) بسیار ناچیز می باشد. با این وجود آدمی قادر است در ۰.۱ ثانیه تصویر یک انسان را بازشناسائی نماید. این قدرت فوق العاده از پردازش موازی توزیع شده در تعدادی زیادی از نرونها بوجود می آید.

۱۱.۳ قاعده یادگیری دلتای ویدرو-هوف

قاعده یادگیری ویدرو-هوف [۲۵] را می‌توان این گونه استنباط کرد که T بردار هدف یا خروجی مطلوب و I نیز حاصلضرب نقطه‌ای وزنها و بردار های ورودی است که طبق رابطه زیر تعریف می‌شود.

$$I = \sum_{i=۱}^{n} w_i x_i. \tag{۱.۳}$$

در اینجا هیچ گونه شیب یا تابع فعالسازی غیر خطی(برای مشتق گرفتن)وجود ندارد، ولی وقتی عوامل غیر خطی زیر به آن اضافه شود، نتیجه کاملاً مطلوب خواهد بود.



شکل ۷.۳: یک عصب با خروجی مطلوب T و خطای εو بدون تابع فعالسازی.

در شکل ۷.۳یک تابع خطا εمشاهده می‌کنید که تابعی از همه وزنها است و تابع مربع خطاها یعنی $\varepsilon^۲$ را می‌توان به این صورت نوشت.

$$\varepsilon = (T - I). \tag{۲.۳}$$

$$\varepsilon^۲ = (T - I)^۲. \tag{۳.۳}$$

گرادیان بردار مربع خطاها عبارتست از مشتق جزئی نسبت به هر یک از اوزان:

$$\frac{\partial \varepsilon^۲}{\partial w_i} = -۲(T - I)\frac{\partial I}{\partial w_i} = -۲(T - I)x_i. \tag{۴.۳}$$

چون این گرادیان فقط شامل iامین وزن است ، پس مجموع حاصلضرب نقطه‌ای اوزان و ورودی‌ها که در رابطه۱.۳ آمده، در گرادیان ظاهر نمی‌شود.

[۲۵] Widrow-Hoff delta learning rule

برای اثبات این موضوع اجازه دهید عصبی را در نظر بگیریم که فقط دارای دو ورودی $x_۱$و$x_۲$ باشد. مربع خطاها برای این عصب عبارتست از:

$$\begin{aligned}\varepsilon^۲ &= [T - w_۱x_۱ - w_۲x_۲]^۲ \\ &= T^۲ + w_۱^۲x_۱^۲ + w_۲^۲x_۲^۲ - ۲Tw_۱x_۱ - ۲Tw_۲x_۲ + ۲w_۱x_۱w_۲x_۲ \\ &= w_۱^۲[x_۱^۲] + w_۱[-۲x_۱(T - w_۲x_۲)] + [(T - w_۲x_۲)^۲] \\ &= w_۲^۲[x_۲^۲] + w_۲[-۲x_۲(T - w_۱x_۱)] + [(T - w_۱x_۱)^۲].\end{aligned} \tag{۵.۳}$$

مربع خطاها زمانی به حداقل خود می‌رسد که مشتقات جزئی مربع خطاها نسبت به وزنهای $w_۱$،$w_۲$ معادل صفر باشند یعنی:

$$\frac{\partial\varepsilon^۲}{\partial w_۱} = -۲[T - w_۱x_۱ - w_۲x_۲]x_۱ = ۰, \tag{۶.۳}$$

$$\frac{\partial\varepsilon^۲}{\partial w_۲} = -۲[T - w_۱x_۱ - w_۲x_۲]x_۲ = ۰. \tag{۷.۳}$$

چون $x_۱$و $x_۲$نمی‌توانند صفر باشند مقادیر داخل براکتها که در هر دو رابطه یکسانند باید صفر باشند پس

$$T - w_۱x_۱ - w_۲x_۲ = ۰. \tag{۸.۳}$$

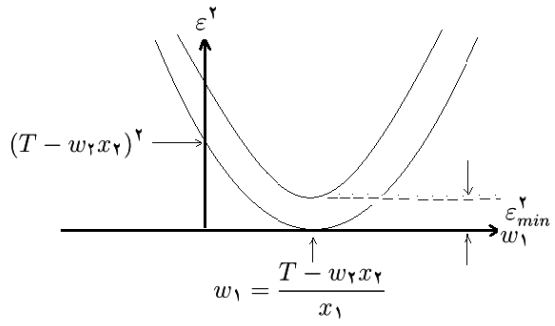
با توجه به رابطه فوق وقتی خطا به حداقل برسد مقادیر$w_۱$ و $w_۲$عبارت خواهند بود از

$$w_۱ = \frac{T - w_۲x_۲}{x_۱}, \tag{۹.۳}$$

$$w_۲ = \frac{T - w_۱x_۱}{x_۲}. \tag{۱۰.۳}$$

اگر هر یک از این مقادیر را در رابطه ۵.۳قرار دهیم مربع خطاها به حداقل مقدار خود یعنی صفر خواهد رسید. از نقطه نظر ریاضی این مطلب صحیح است ولی در دنیای واقعی به دلایلی همچون غیر خطی بودن توابع و اغتشاش و نقص داده ها، حداقل مربع خطاها هرگز صفر نخواهد بود. به عنوان مثال وجود اغتشاش در تابع فعالسازی سیگموئیدی یا sباعث می‌شود حداقل مربع خطاها صفر نشود،بلکه مقداری است که آن را$\varepsilon_{min}^۲$ می‌نامیم.

بررسی رابطه ۵.۳نشان می دهد که تابع $\varepsilon^۲$نسبت به $w_۱$و $w_۲$بصورت سهمی است. منحنی سهمی وار $\varepsilon^۲$نسبت به $w_۱$در شکل ۸.۳برای دو حالت رسم شده است، یکی زمانی که مینیمم مربع خطاها صفر است وحالت دوم موقعیست که مینیمم مربع خطاها معادل $\varepsilon_{min}^۲$باشد. در هر دو حالت مینیمم مربع خطاها برای یک مقدار خاص از$w_۱$اتفاق می افتد که این مقدار در رابطه ۹.۳مشخص شده

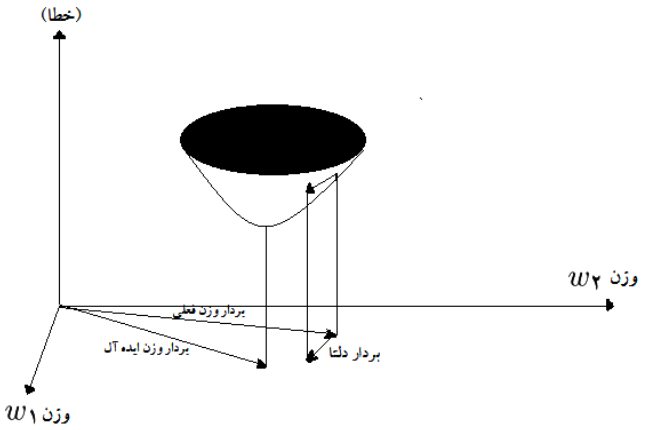


شکل ۸.۳: مینیمم کردن مربع خطاها در خلال آموزش ویدرو-هوف

است. تابع مربع خطاها نسبت به w_2نیز کاملاً مشابه تابع فوق است و حداقل این تابع به ازای مقدار خاصی ازw_2 بدست می‌آید که در رابطه ۱۰.۳مشخص شده است. پس سطح حداقل مربع خطاها برای وزن دوبعدی (نسبت به دو وزن w_1وw_2) بصورت مخروطی است که از چرخش تابع ε^2در صفحه (w_2وw_1)بدست می‌آید. یک تفسیر هندسی از قاعده دلتا این است که این قاعده برای به حداقل رسانیدن مربع خطاها از یک الگوریتم کاهش گرادیان استفاده می‌کند. اگر مربع خطاها را به صورت سه بعدی در نظر بگیریم (یعنی w_1، w_2و ε^2)سطح مربع خطاها یک مخروط چرخشی است که در آن بردار وزن در طول بردار گرادیان مماس بر سطح مخروط، به سوی نقطه مینیمم در حال پایین آمدن است. تصویر بردار گرادیان برروی صفحه $w_1 - w_2$ بردار دلتا را تشکیل می دهد که در شکل ۹.۳نیز نشان داده شده است. قاعده دلتا، بردار وزنی را در طول گرادیان منفی منحنی بسوی بردار وزنی ایده ال حرکت می دهد. چون در این روش از گرادیان استفاده می‌شود آن را الگوریتم کاهش گرادیان یا کاهش شیب می‌نامند. از آنجا که گرادیان بهترین راه برای رسیدن به نقطه پایین سطح منحنی است پس قاعده دلتا نیز بهترین روش برای یافتن حداقل مربع خطاهاست. ولی به این نکته مهم نیز باید توجه داشت:مطلب فوق در صورتی صحیح است که بردار وزنی به سوی یک مینیمم کلی در حال نزول باشد. در مسائل چند بعدی مینیمم‌های موضعی زیادی وجود دارد و باید از تکنیک‌های دیگری استفاده کنیم تا مطمئن شویم که جواب مساله در یکی از مینیمم‌های موضعی به تله نیفتاده باشد.

قانون یادگیری دلتای ویدرو-هوف نشان می‌دهد که میزان تغییر در هر بردار وزنی متناسب با منفی گرادیان آن است:

$$\Delta w_i = -K \frac{\partial \varepsilon^2}{\partial w_i} = 2K(T - I)x_i = 2K\varepsilon x_i^2. \qquad (11.3)$$



شکل ۹.۳: تعبیر هندسی قاعده دلتا

در این رابطه Kمقدار تناسب است. چون این فرایند یک فرایند حداقل سازی است پس علامت منفی در جلوی آن آمده است. معمولاً بردار ورودی جزئی x_iرا با تقسیم بر$|X|^۲$ نرمالایز می‌کنند. در این صورت رابطه ۱۱.۳به شکل زیر در خواهد آمد.

$$\Delta w_i = [۲K|X|^۲]\frac{\varepsilon x_i}{|X|^۲} = \frac{\eta \varepsilon x_i}{|X|^۲}. \qquad (۱۲.۳)$$

حال اگر ما مقدار ثابت یادگیری η را مساوی با مقدار داخل براکت تعریف کنیم این رابطه معادل$\Delta w_i = \frac{\eta \varepsilon x_i}{|X|^۲}$خواهد شد.

$$\eta = ۲K|X|^۲. \qquad (۱۳.۳)$$

۱۲.۳ یاد گیری پس انتشار برای شبکه عصبی چند لایه

قبل از بحث درباره جزئیات فرایند پس انتشار، اجازه دهید که به مزایای لایه‌های میانی در شبکه عصبی مصنوعی بپردازیم. یک شبکه عصبی مصنوعی دو لایه (لایه ورودی و لایه خروجی)فقط می‌تواند ورودی را با همه اطلاعاتی که در آن موجود است نشان دهد. از این رو اگر داده‌ها منفصل بوده و یا به طور خطی تفکیک پذیر باشند در نمایش درونی داده ها (در درون شبکه) تناقص ایجاد می‌شود و شبکه نمی‌تواند انطباق ورودی-خروجی را یاد بگیرد. اضافه کردن یک لایه سوم به شبکه به آن اجازه می‌دهد تا این انطباق را در درون خود نشان دهد. در واقع این نمایش درونی انطباقها که بسیار پیچیده و در عین حال توانمند است به شبکه اجازه می‌دهد که هر نوع انطباقی را یاد بگیرد،

نه اینکه فقط انطباقهای خطی و منفصل انجام دهد.
با بکارگیری تئوری کولموگروف [۲۶]در مورد شبکه های عصبی می‌توان راهنمایی خوبی درباره تعداد عصب های لایه میانی (لایه پنهان)کسب نمود. در هر شبکه عصبی هدف این است که یک بردار m بعدی به یک بردار n بعدی تبدیل شود.حال فرض کنید که بردار های ورودی همگی در فاصله صفر تا یک هستند ولی برای بردار خروجی هیچ محدودیتی وجود ندارد. تئوری کولموگروف به ما می‌گوید که یک شبکه عصبی سه لایه وجود دارد که این انطباق را به طور دقیق (و نه تقریبی) انجام دهد. این شبکه در لایه ورودی دارای m عصب است، در لایه خروجی دارای n عصب و در لایه میانی دارای ۲m+۱ عصب است. بنابراین تئوری کولموگروف تضمین می‌کند که یک شبکه عصبی سه لایه می‌تواند همه مسایل غیرخطی تفکیک پذیر را حل کند. آنچه تضمین نمی‌شود این است که:

- این شبکه ، بهترین شبکه برای انجام این انطباق است،
- یک شبکه کوچکتر نمی‌تواند این انطباق را انجام دهد ،
- یک شبکه ساده تر نمی‌تواند این انطباق را به همان خوبی انجام دهد.

متاسفانه این تئوری، برای یافتن و ساخت شبکه‌ای که به بهترین شکل این انطباق را انجام دهد جزئیات کافی ارایه نمی‌دهد ولی به هر حال تضمین می‌کند که شبکه‌های عصبی می‌توانند انطباق این دو بردار را انجام دهند.
حال شبکه سه لایه شکل۱۰.۳ را در نظر بگیرید. که همه توابع فعال سازی آن از نوع توابع لجستیک هستند. ذکر این نکته ضروری است که روش پس انتشار را می‌توان برای شبکه‌های عصبی با هر تعداد لایه میانی بکار برد. هدف یادگیری این است که وزن‌ها را بگونه‌ای تعدیل کنیم که با ارایه یک مجموعه از ورودی‌های، خروجی‌های مطلوب بدست آیند. برای انجام این کار معمولا شبکه را توسط تعداد زیادی از زوج‌های ورودی-خروجی که آنها را مثال می‌نامیم آموزش می‌دهند.
روش آموزش به شرح زیر است:

۱. وزن ها را به صورت اعداد تصادفی کوچک (هم مثبت و هم منفی) انتخاب کنید تا اطمینان حاصل شود که شبکه توسط وزن های بزرگ اشباع نمی‌شود (اگر در ابتدای کار مقادیر وزن ها مساوی باشند و خروجی مطلوب نیاز به وزن های غیر مساوی داشته باشد شبکه هرگز آموزش نخواهد دید).

[۲۶] kolmogorov 's rheorem

۲. یک جفت ورودی-خروجی آموزشی از مجموعه آموزشی(مجموعه مثال‌ها) انتخاب کنید ،

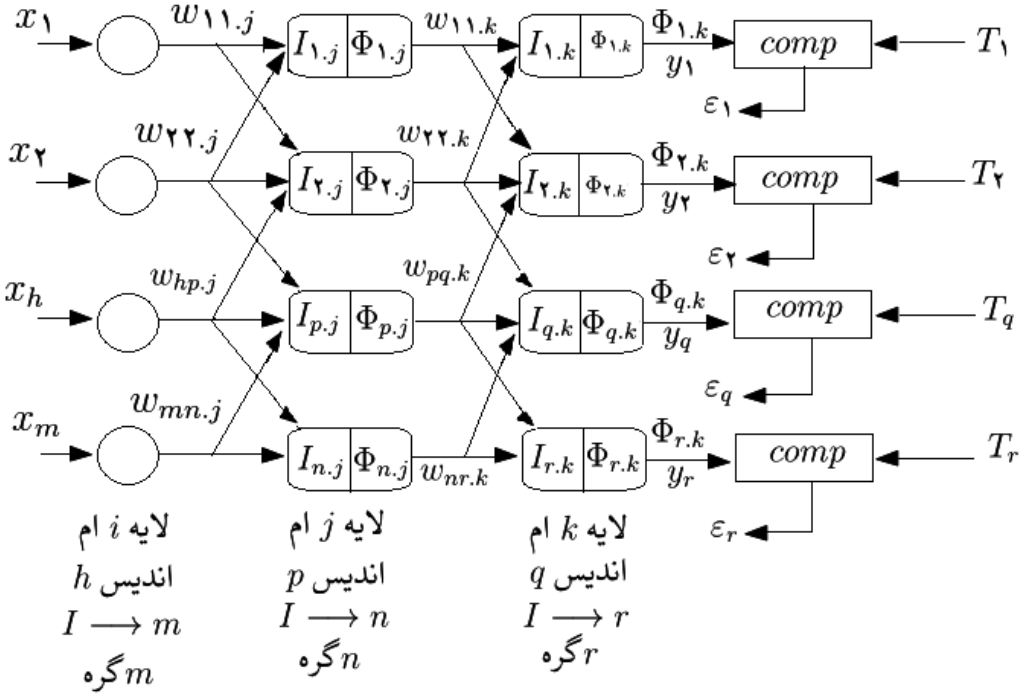
۳. بردار ورودی را وارد شبکه کنید،

۴. خروجی شبکه را محاسبه کنید،

۵. میزان خطا یعنی تفاوت بین خروجی واقعی شبکه و خروجی مطلوب را محاسبه کنید،

۶. وزن های شبکه را بگونه‌ای تعدیل کنید که خطا حداقل شود،

۷. گام‌های ۲ تا ۶ برای هر یک از زوج های آموزشی موجود در مجموعه مثال ها تکرار کنید تا آنجا که خطاهای کل سیستم تا حد قابل قبولی کم شود.



شکل ۱۰.۳: نمای یک شبکه عصبی چند لایه همراه با علائم و اندیس‌های الگوریتم پس انتشار

آموزش شبکه عصبی شامل دو عبور است. در عبور رو به جلو، مقادیر ورودی، از ورودی شبکه به سوی خروجی انتشار می‌یابند. در عبور رو به عقب مقادیر خطای محاسبه‌شده به عقب

منتشر می‌شود تا از آنها برای تعدیل وزن‌ها استفاده می‌شود. در عبور رو به جلو، محاسبه خروجی بصورت لایه به لایه انجام می‌شود و خروجی هر لایه، ورودی لایه بعدی به حساب می‌آید. در عبور معکوس ابتدا وزن‌های لایه خروجی تعدیل می‌شود زیرا برای هر یک از اعصاب لایه خروجی، خروجی مطلوب وجود دارد و می‌توان با استفاده از آن و قاعده دلتا، وزن‌ها را تعدیل نمود. حال وزن های لایه میانی را تعدیل می‌کنیم. مشکل اینجاست که برای عصب‌های لایه میانی، مقادیر خروجی مطلوب تعریف نشده است. به همین دلیل کار آموزش پیچیده تر می‌شود زیرا این خطا باید به سمت عقب در شبکه منتشر شود و لایه به لایه توابع غیرخطی را در بر گیرد.

۱.۱۲.۳ محاسبه وزن‌ها برای اعصاب لایه خروجی

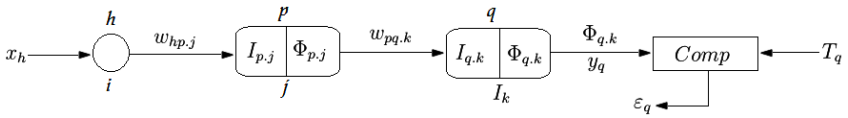
حال اجازه دهید جزئیات فرایند یادگیری پس انتشار را برای وزن‌های لایه خروجی مطرح کنیم.شکل ۱۱.۳نمایشی از آموزش عصب‌ها است.در این شکل لایه خروجی با زیر نویس k مشخص شده و عصب‌ها با p و q نشان داده شده‌اند.همچنین خروجی‌ها(خروجی اعصاب)با $\Phi_{q.k}(I)$ و $\Phi_{p.j}(I)$ ،وزن‌های ورودی با $w_{hp.j}$و $w_{pq.k}$ و مقدار هدف(خروجی مطلوب)با T مشخص شده است.برای سهولت کار در نوشتن $\Phi_{q.k}(I)$ از(I) صرف نظر می‌کنیم.تفاوت خروجی عصب لایه امk و مقدار هدف، محاسبه شده و به توان می‌رسد تا مربع خطا بدست آید که برای لایه k ام عبارت است از

$$\varepsilon = \varepsilon_q = [T_q - \Phi_{q.k}]. \qquad (۱۴.۳)$$

که در آن فقط یک خطای خروجی وجود دارد. پس

$$\varepsilon^۲ = \varepsilon_q^۲ = [T_q - \Phi_{q.k}]^۲. \qquad (۱۵.۳)$$

قاعده دلتا نشان می‌دهد که میزان تغییر یک وزن متناسب با میزان تغییر مربع خطاها نسبت به آن



شکل ۱۱.۳: نمایی از آموزش عصب‌ها برای محاسبه میزان تغییر در وزن عصب لایه خروجی با استفاده از روش پس انتشار

وزن است یعنی

$$\Delta w_{pq.k} = -\eta_{p.q}\frac{\partial \varepsilon_q^{۲}}{\partial w_{pq.k}}. \qquad (۱۶.۳)$$

در این رابطه $\eta_{p.q}$ ثابت تناسب است که نرخ یادگیری نام دارد. برای محاسبه این مشتق جزئی از قاعده دیفرانسیل جزء به جزء استفاده می‌کنیم:

$$\frac{\partial \varepsilon_q^{۲}}{\partial w_{pq.k}} = \frac{\partial \varepsilon_q^{۲}}{\partial \Phi_{q.k}}\frac{\partial \Phi_{q.k}}{\partial I_{q.k}}\frac{\partial I_{q.k}}{\partial w_{pq.k}}. \qquad (۱۷.۳)$$

عبارات رابطه فوق به ترتیب محاسبه می‌شوند. مشتق جزئی رابطه نسبت به $\Phi_{q.k}$عبارت است از :

$$\frac{\partial \varepsilon_q^{۲}}{\partial \Phi_{p.k}} = -۲[T_q - \Phi_{q.k}]. \qquad (۱۸.۳)$$

از رابطه

$$\frac{\partial \Phi(I)}{\partial I} = \alpha\frac{۱ - \Phi(I)}{\Phi(I)}\Phi^{۲}(I) = \{\alpha[۱ - \Phi(I)]\Phi(I)\} = \alpha(۱ - \Phi)\Phi. \qquad (۱۹.۳)$$

نتیجه می‌شود که

$$\frac{\partial \Phi_{q.k}}{\partial I_{q.k}} = \alpha\Phi_{q.k}[۱ - \Phi_{q.k}]. \qquad (۲۰.۳)$$

در شکل ۱۰.۳می‌بینیم که $I_{q.k}$، مجموع ورودی‌های وزنداری است که از لایه میانی آمده‌اند پس

$$I_{q.k} = \sum_{p=۱}^{n} w_{pq.k}\Phi_{p.j}. \qquad (۲۱.۳)$$

که اگر از آن نسبت به $w_{pq.k}$مشتق جزئی بگیریم خواهیم داشت:

$$\frac{\partial I_{q.k}}{\partial w_{pq.k}} = \Phi_{p.j}. \qquad (۲۲.۳)$$

اگر ما فقط با یک وزن سر و کار داشته باشیم فقط یکی از جملات در رابطه مجموع ۲۱.۳ باقی می‌ماند. با جایگزینی روابط ۱۹.۳، ۲۰.۳ و۲۲.۳ در رابطه ۱۶.۳خواهیم داشت :

$$\frac{\partial \varepsilon_q^{۲}}{\partial w_{pq.k}} = -۲\alpha[T_q - \Phi_{q.k}]\Phi_{q.k}[۱ - \Phi_{q.k}]\Phi p.j = -\delta_{pq.k}\Phi_{p.j}. \qquad (۲۳.۳)$$

که در آن مقدار $\delta_{pq.k}$بصورت زیر تعریف می‌شود

$$\delta_{pq.k} \equiv ۲\alpha[T_q - \Phi_{q.k}]\Phi_{q.k}[۱ - \Phi_{q.k}] = ۲\varepsilon_q\frac{\partial \Phi_{q.k}}{\partial I_{q.k}}. \qquad (۲۴.۳)$$

همچنین با جایگزینی رابطه ۲۳.۳ در رابطه ۱۶.۳ داریم

$$\Delta w_{pq.k} = -\eta_{p.q}\frac{\partial \varepsilon_q^2}{\partial w_{pq.k}} = -\eta_{p.q}\delta_{pq.k}\Phi_{p.j}, \qquad (۲۵.۳)$$

$$w_{pq.k}(N+1) = w_{pq.k}(N) - \eta_{p.q}\delta_{pq.k}\Phi_{p.j}. \qquad (۲۶.۳)$$

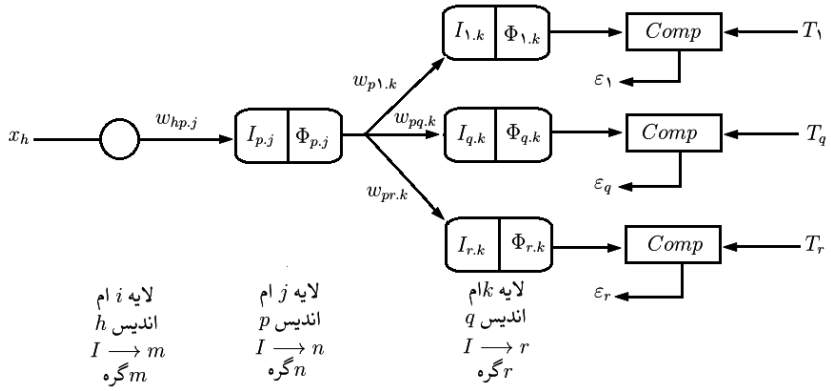
که N تعداد دفعات تکرار فرایند است. فرایند مشابهی برای تک تک وزنهای لایه خروجی انجام می‌شود تا مقادیر تعدیل شده وزنها بدست آیند. با استفاده از میزان خطا یعنی $\delta_{pq.k}$ که در رابطه ۲۴.۳ بدست می‌آید، می توان وزنها را با استفاده از روابط ۲۵.۳ و ۲۶.۳ تعدیل نمود. لازم است در این خصوص بحث کنیم که چرا مشتق تابع فعالسازی (تابع محرک) در این فرایند دخیل است. ما در رابطه ۲۴.۳ خطایی را محاسبه می‌کنیم که باید در جهت معکوس در شبکه منتشر شود. این خطا به این دلیل وجود دارد که عصب خروجی، خروجی اشتباه تولید کرده است و علت‌های آن عبارتند از (۱) وزنهای مربوط به عصب خروجی، مقادیر نادرستی داشته‌اند و یا (۲) عصب‌های لایه میانی خروجی اشتباه تولید کرده اند. برای اینکه این خطا به وزنهای ارتباطی یا به اعصاب لایه میانی تخصیص یابد، ما باید خطا را از طریق همان ارتباطات و وزنهایی که خروجی لایه میانی را به لایه خروجی می‌فرستند در جهت معکوس منتشر کنیم.

اگر وزن بین عصب لایه میانی و عصب لایه خروجی مقدار بزرگی داشته باشد و خطای عصب لایه خروجی نیز زیاد باشد، ممکن است به وزنهای اعصاب لایه میانی، خطای بسیار بزرگی تخصیص داده شود. حتی اگر آن عصب لایه میانی ، خروجی بسیار کوچکی داشته باشد و سهم زیادی در خطای عصب خروجی نداشته باشد باز هم این موضوع صادق است. این خطای بزرگ با استفاده از مشتق تابع فشرده سازی، کوچکتر می‌شود و میزان تغییر در اوزان لایه میانی را نیز کمتر خواهد کرد. علت این امر شکل زنگوله مانند منحنی تابع مشتق است.

۱۳.۳ محاسبه وزنها برای اعصاب لایه پنهان(میانی)

از آنجا که لایه های پنهان هیچگونه بردار هدف یا خروجی مطلوبی ندارند، مشکل تعدیل وزن‌های لایه‌های میانی برای چندین سال مانع از پیشرفت کار شبکه‌ها شده بود تا اینکه الگوریتم پس‌انتشار بوجود آمد. پس‌انتشار، خطای تعدیل یافته را در جهت معکوس و بصورت لایه به لایه در شبکه منتشر می‌کند و با رسیدن به هر لایه، وزنهای آن را تعدیل می‌نماید. روابط موجود برای آموزش لایه پنهان مشابه روابطی است که برای لایه خروجی تعریف شد. با استفاده از این تفاوت که خطای $\delta_{hp.j}$ باید بدون بردار هدف تعریف شود. ما بایدخطای $\delta_{hp.j}$ برای هر عصب لایه میانی را بگونه‌ای

محاسبه کنیم که سهم ناشی از خطای هر یک از اعصاب لایه خروجی را که به آن متصل است در نظر بگیرد. حال یک عصب منفرد را در یک لایه پنهان در نظر بگیریدکه درست قبل از لایه خروجی قرار گرفته است. این عصب را با زیر نویس p نشان می‌دهیم (شکل ۱۱.۳)در عبور رو به جلو این عصب، خروجی خود را از طریق وزنهای ارتباطی $w_{pq.k}$ به عصب‌های q در لایه خروجی می‌فرستد. در فرایند آموزش، این وزنها در جهت معکوس عمل می‌کنند و خطای $\delta_{pq.k}$ را از لایه خروجی به لایه پنهان می‌فرستند.هر یک از این وزنها در مقدار خروجی عصب p ضرب می‌شوند و به آن دسته از اعصاب لایه خروجی که به عصب p متصل هستند وارد می‌شوند. مقدار خطای $\delta_{hp.j}$ که برای عصب لایه پنهان مورد نیاز است از مجموع این حاصلضربها بدست می‌آید.



شکل ۱۲.۳: نمایش آموزش عصبها برای محاسبه میزان وزن یک عصب لایه میانی(پنهان) در الگوریتم پس انتشار

شکل ۱۲.۳خطاهایی را نشان می‌دهد که باید در جهت معکوس منتشر شوند تا میزان تغییر(تعدیل) در وزن $w_{hp.j}$مشخص شود. از آنجا که با تک تک خطا های لایه خروجی سروکار داریم پس مشتق جزئی باید از مجموع خروجیr گرفته شود. محاسبات$\delta_{hp.j}$ کاملاً شبیه محاسبات $\delta_{pq.k}$است . حال از مربع خطاها نسبت به وزنهای لایه میانی که باید تعدیل شوند مشتق می‌گیریم. بنابراین با استفاده از قاعده یادگیری دلتا رابطه شبیه رابطه۱۶.۳ خواهیم داشت:

$$\Delta w_{hp.j} = -\eta_{h.p}\frac{\partial \varepsilon^2}{\partial w_{hp.j}} = -\eta_{h.p}\sum_{q=1}^{r}\frac{\partial \varepsilon_q^2}{\partial w_{hp.j}}. \qquad (۲۷.۳)$$

حال میانگین مربع خطای کل یعنی بصورت زیر تعریف می‌شود.

$$\varepsilon^2 = \sum_{q=1}^{r}\varepsilon_q^2 = \sum_{q=1}^{r}[T_q - \Phi_{q.k}]^2. \qquad (۲۸.۳)$$

رابطه فوق ممکن است شامل چندین خطای خروجی باشد. مقدار ثابت یادگیری $\eta_{h.p}$معمولا (و نه ضرورتاً)مساوی $\eta_{p.q}$است. ما مجدداً می‌توانیم رابطه ۲۷.۳را با استفاده از قاعده دیفرانسیل جزءبه جزء برسی کنیم:

$$\frac{\partial \varepsilon_q^2}{\partial w_{hp.j}} = \sum_{q=1}^{r} \frac{\partial \varepsilon_q^2}{\partial \Phi_{q.k}} \frac{\partial \Phi_{q.k}}{\partial I_{q.k}} \frac{\partial I_{q.k}}{\partial \Phi_{p.j}} \frac{\partial \Phi_{p.j}}{\partial I_{p.j}} \frac{\partial I_{p.j}}{\partial w_{hp.j}}. \tag{۲۹.۳}$$

جملات فوق شبیه رابطه ۲۷.۳ است و روش محاسبه آن نیز به همان صورت است.

دو جمله اول که در روابط ۱۸.۳ و ۲۰.۳آمده‌اند عبارتند از:

$$\frac{\partial \varepsilon_q^2}{\partial \Phi_{q.k}} = -2(T_q - \Phi_{q.k}) = -2\varepsilon_q \ . \tag{۳۰.۳}$$

$$\frac{\partial \Phi_{q.k}}{\partial I_{q.k}} = \alpha \Phi_{q.k}(1 - \Phi_{q.k}) \ . \tag{۳۱.۳}$$

حال رابطه ۲۱.۳را در نظر بگیرید:

$$I_{q.k} = \sum_{p=1}^{n} w_{pq.k} \Phi_{p.j} \ . \tag{۳۲.۳}$$

اگر از این رابطه نسبت به $\Phi_{p.j}$ مشتق جزئی بگیریم خواهیم داشت:

$$\frac{\partial I_{q.k}}{\partial \Phi_{p.j}} = w_{pq.k} \ . \tag{۳۳.۳}$$

در اینجا دیگر علامت جمع بر روی pوجود ندارد زیرا فقط یک ارتباط(یک وزن ارتباطی)مد نظر است. اگر در رابطه ۲۰.۳اندیس‌های مربوط به لایه میانی را جایگزین اندیس‌های خروجی کنیم خواهیم داشت:

$$\frac{\partial \Phi_{p.j}}{\partial \Phi_{p.j}} = \alpha \Phi_{p.j}[1 - \Phi_{p.j}]. \tag{۳۴.۳}$$

همچنین اگر اندیس های رابطه۲۱.۳ را تغییر دهیم و ورودی i امین لایه یعنی x_h را جایگزین خروجی j امین لایه یعنی $\Phi_{p.j}$ نماییم خواهیم داشت:

$$I_{p.j} = \sum_{h=1}^{m} w_{hp.j} x_h \ . \tag{۳۵.۳}$$

اگر از رابطه۳۴.۳ مشتق بگیریم :

$$\frac{\partial I_{p.j}}{\partial w_{hp.j}} = x_h \ . \tag{۳۶.۳}$$

مجدداً در رابطه ۳۴.۳علامت جمع بر روی h از بین رفته است چون فقط یک وزن ارتباطی را در نظر

داریم. اگر روابط۳۱.۳ تا ۳۶.۳را در رابطه ۲۹.۳جاگذاری کنیم و از رابطه۲۸.۳ و تعریف$\delta_{pq.k}$در رابطه۲۴.۳ استفاده کنیم خواهیم داشت:

$$\begin{aligned}\frac{\partial \varepsilon^{۲}}{\partial w_{hp.j}} &= \sum_{q=۱}^{r}(-۲)\alpha(T_q - \Phi_{q.k})[\Phi_{q.k}(۱ - \Phi_{q.k})]w_{pq.k}\alpha[\Phi_{p.j}(۱ - \Phi_{p.j})]x_h \\ &= -\sum_{q=۱}^{r}\delta_{pq.k}w_{pq.k}\frac{\partial \Phi_{p.j}}{\partial I_{p.j}}x_h \ .\end{aligned} \tag{۳۷.۳}$$

اگر ما $\delta_{hp.j}$را به صورت زیر تعریف کنیم:

$$\delta_{hp.j} \equiv \delta_{pq.k}w_{pq.k}\frac{\partial \Phi_{p.j}}{\partial I_{p.j}} \ . \tag{۳۸.۳}$$

در آن صورت رابطه ۳۷.۳بصورت زیر در خواهد آمد.

$$\frac{\partial \varepsilon^{۲}}{\partial w_{hp.j}} = -\sum_{q=۱}^{r}\delta_{hp.j}x_h \ . \tag{۳۹.۳}$$

همان طور که رابطه ۲۷.۳ نشان می‌دهد میزان تغییر وزنها متناسب با منفی نرخ تغییر مربع خطاها نسبت به همان وزنهاست. بنابراین با جایگزینی روابط ۳۷.۳و ۳۸.۳در رابطه ۲۷.۳خواهیم داشت:

$$\begin{aligned}\Delta w_{hp.j} &= -\eta_{h.p}\frac{\partial \varepsilon^{۲}}{\partial w_{hp.j}} = \eta_{h.p}\sum_{q=۱}^{r}\delta_{pq.k}w_{pq.k}\frac{\partial \Phi_{p.j}}{\partial I_{p.j}}x_h \\ &= \eta_{h.p}x_h\sum_{q=۱}^{r}\delta_{hp.j} \ .\end{aligned} \tag{۴۰.۳}$$

و بنابراین

$$w_{hp.j}(N+۱) = w_{hp.j}(N) + \eta_{h.p}x_h\sum_{q=۱}^{r}\delta_{hp.j} \ . \tag{۴۱.۳}$$

اگر بیش از یک لایه میانی وجود داشته باشد این فرایند به طور لایه به لایه انجام می‌شود و وزنها را تعدیل می‌کند تا به لایه ورودی برسد وقتی فرایند به اتمام رسید ورودی آموزشی جدید، وارد شبکه می‌شود و کل فرایند یادگیری برای این ورودی انجام می‌شود. این کار آنقدر ادامه می‌یابد تا خطا تا حد قابل قبولی کاهش یابد.در این مرحله شبکه کاملاً آموزش دیده است.

مثال ۱.۳. ***تعدیل وزنها با استفاده از الگوریتم پس انتشار*** *در شکل ۱۳.۳ یک شبکه عصبی ساده پیش‌خور کاملاً مرتبط نشان داده شده است که در آن به هر یک از اعصاب C، D و E یک ورودی*

موثر ۱+ همراه با وزنهای ارتباطی $w_{\backslash C}$ ، $w_{\backslash D}$ و $w_{\backslash E}$ وارد شده است همه عصب ها دارای یک تابع فعال سازی لجستیک با $\alpha = ۱$ و ثابت یادگیری $\eta = ۰/۵$ هستند.
خروجی مطلوب عصب E معادل ۱.۰ است.وزنهایی که نشان داده شده‌اند به طور تصادفی انتخاب شده‌اند و فرایند آموزش شروع شده است. سپس فرایند پس انتشار در جهت معکوس شبکه انجام می‌شود تا میزان تغییر در شش وزن ارتباطی و مقادیر جدید وزنها محاسبه شود.

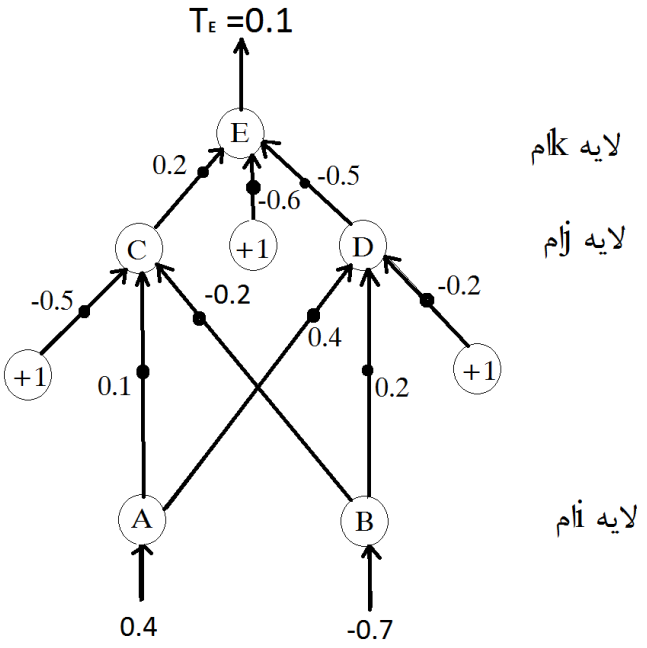
$$\Delta w_{pq.k} = -\eta_{p.q}[-۲\alpha[T_q - \Phi_{q.k}]\Phi_{q.k}[۱ - \Phi_{q.k}]\Phi_{p.j}] \; . \qquad (۴۲.۳)$$

همچنین با جایگزینی رابطه ۳۷.۳ در رابطه۴۰.۳ وزنها بین لایه i و لایه j به صورت زیر تغییر می‌کنند

$$\Delta w_{hp.j} = -\eta_{h.p}\left[\sum_{q=۱}^{r} -۲\alpha[T_q - \Phi_{q.k}]\Phi_{q.k}[۱ - \Phi_{q.k}]w_{pq.k}\alpha\Phi_{p.j}(۱ - \Phi_{p.j})x_h\right] \; . \qquad (۴۳.۳)$$

زیرنویس‌های q ، p، h و اندیس‌های m ، n و r در شکل ۱۳.۳ تعریف شده‌اند. زیر نویس ۱ به ورودی موثر ۱+ اشاره می‌کند. ابتدا خروجی عصبها با استفاده از تابع لجستیک که مقدار α آن ۱ است محاسبه می‌شود.

$$I_C = ۰/۴ \times ۰/۱ + (-۰/۷) \times (-۰/۲) + ۱ \times (-۰/۵) = ۰/۰۴ + ۰/۱۴ - ۰/۵ = -۰/۳۲$$
$$\Phi(I_C) = ۰/۴۲$$
$$I_D = ۰/۴ \times ۰/۴ + (-۰/۷) \times ۰/۲ + ۱ \times (-۰/۲) = ۰/۱۶ - ۰/۱۴ - ۰/۲ = -۰/۱۸$$
$$\Phi(I_D) = ۰/۴۶$$
$$I_E = ۰/۴۲ \times ۰/۲ + ۰/۴۶ \times (-۰/۵) + ۱ \times (-۰/۶) = ۰/۰۸ - ۰/۲۳ - ۰/۶۰ = -۰/۷۵$$
$$\Phi(I_E) = ۰/۳۲$$



شکل ۱۳.۳: تعدیل (آموزش) وزنها در یک شبکه عصبی ساده

با جایگذاری این مقادیر و سایر پارامترهای شبکه در روابط ۴۲.۳ *و* ۴۳.۳ *خواهیم داشت.*

$$\Delta w_{CE} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱۰ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲) \times ۰/۴۲ = -۰/۰۲۰$$

$$\Delta w_{DE} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱۰ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲) \times ۰/۴۶ = -۰/۰۲۲$$

$$\Delta w_{IE} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱۰ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲) = -۰/۰۴۸$$

$$\Delta w_{AE} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱۰ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲) \times ۰/۲ \times ۱ \times ۰/۴۲ \times (۱ - ۰/۴۲) \times ۰/۴ = -۰/۰۰۰۹۳ = -۰/۰۰۱$$

$$\Delta w_{AD} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲) \times (-۰/۵) \times ۱ \times ۰/۴۶ \times (۱ - ۰/۴۶) \times ۰/۴ = ۰/۰۰۲۳۸ = ۰/۰۰۲$$

$$\Delta w_{BC} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲) \times ۰/۲ \times ۱ \times ۰/۴۲ \times (۱ - ۰/۴۲) \times (-۰/۷) = ۰/۰۰۱۶۳ = ۰/۰۰۲$$

$$\Delta w_{BD} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲)$$
$$\times (-۰/۵) \times ۱ \times ۰/۴۶ \times (۱ - ۰/۴۶) \times (-۰/۷) = -۰/۰۰۱۴۲ = ۰/۰۰۱$$
$$\Delta w_{IC} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲)$$
$$\times ۰/۲ \times ۱ \times ۰/۴۲ \times (۱ - ۰/۴۲) \times ۱ = -۰/۰۰۲۳ = -۰/۰۰۲$$
$$\Delta w_{IC} = -۰/۵ \times (-۲) \times ۱ \times (۰/۱ - ۰/۳۲) \times ۰/۳۲ \times (۱ - ۰/۳۲)$$
$$\times (-۰/۵) \times ۱ \times ۰/۴۶ \times (۱ - ۰/۴۶) \times ۱ = ۰/۰۰۵۹ = ۰/۰۰۶$$

وزنهای جدید اعمال این تغییرات بر روی وزنهای اصلی بدست می‌آیند.

$$w_{CE} = ۰/۲۰۰ - ۰/۰۲۰ = ۰/۱۸۰$$
$$w_{DE} = -۰/۵۰۰ - ۰/۰۲۲ = -۰/۵۲۲$$
$$w_{IE} = -۰/۶۰۰ - ۰/۰۴۸ = -۰/۶۴۸$$
$$w_{AC} = ۰/۱۰۰ - ۰/۰۰۱ = ۰/۰۹۹$$
$$w_{AD} = ۰/۴۰۰ + ۰/۰۰۲ = ۰/۴۰۲$$
$$w_{BC} = -۰/۲۰۰ + ۰/۰۰۲ = -۰/۱۹۸$$
$$w_{BD} = ۰/۲۰۰ - ۰/۰۰۱ = ۰/۱۹۹$$
$$w_{IC} = -۰/۵۰۰ - ۰/۰۰۲ = -۰/۵۰۲$$
$$w_{ID} = -۰/۲۰۰ + ۰/۰۰۶ = -۰/۱۹۴$$

این فرایند آنقدر تکرار می‌شود تا از همه جفت مثالهای موجود در مجموعه آموزشی استفاده شود.بعد از محاسبه میزان تغییر وزنها، کل مربع خطاها محاسبه می‌شود. اگر مقدار مربع خطاها از مقدار مشخصی بیشتر باشد باید الگوریتم یادگیری با استفاده از یک جفت آموزشی دیگر روی شبکه اعمال شود.یک راه حل بهتر اینست که فرایند آموزش را آنقدر ادامه دهیم تا کل مربع خطاها برای یک مجموعه آموزشی شروع به افزایش کند ولی مربع خطاها برای مجموعه آموزشی رو به کاهش باشد.

۱.۱۳.۳ شبکه‌های دلتا-بار-دلتای پیشرفته

مینایی و ویلیامز[۲۷] در سال ۱۹۹۰الگوریتم دلتا-بار-دلتا را بصورت زیر بسط دادند:(۱)اضافه کردن یک عبارت گشتاور که در طی زمان تغییر می‌کرد (۲)اضافه کردن یک نرخ یادگیری که آنهم به مرور زمان تغییر می‌کرد.میزان گشتاور و نرخ یادگیری با توجه به اجرای گرادیان وزنی، به طور نمایی کاهش یافته و بنابراین در قسمتهایی که شیب یا انحنای کمی وجود دارد گشتاور و نرخ یادگیری ، بیشتر افزایش می‌یابد.برای تک تک مقادیر گشتاور و نرخ یادگیری، یک حد بالا در نظر گرفته شده تا از پرش و نوسانهای شدید در فضای وزنها جلوگیری شود.

۱۴.۳ تحلیل حساسیت در شبکه‌های عصبی پس انتشار

تجزیه و تحلیل حساسیت در بسیاری از زمینه‌های کاربردی،ابزار بسیار مفیدی است. اگر آشفتگی کوچکی در یکی از عصب‌های ورودی وجود داشته باشد معمولا در همه عصب‌های خروجی که مستقیم به آن عصب ورودی متصلند آشفتگی ایجاد می‌شود.نسبت میزان آشفتگی در خروجی یک عصب خروجی خاص به میزان آشفتگی در ورودی یک عصب ورودی خاص، به عنوان حساسیت تعریف می‌شود. در شبکه عصبی چند لایه شکل آشفتگی در تمام مقادیر آشفتگی ایجاد می‌کند.پس حساسیت عبارت است از

$$\sigma_{q.h} = \frac{\Delta y_q}{\Delta x_h} = \frac{\partial y_q}{\partial x_h} \ . \qquad (۴۴.۳)$$

در رابطه فوق مشتق جزئی جایگزین نسبت آشفتگیΔ شده است(البته بعد از گرفتن حد لازم).با توجه به شکل ، رابطه۴۴.۳ به این صورت در می‌آید.

$$\sigma_{q.h} = \frac{\partial y_q}{\partial x_h} = \frac{\partial \Phi_{q.k}}{\partial x_h} = \frac{\partial \Phi_{q.k}}{\partial I_{q.k}} \frac{\partial I_{q.k}}{\partial \Phi_{p.j}} \frac{\partial \Phi_{p.j}}{\partial I_{p.j}} \frac{\partial I_{p.j}}{\partial x_h} \ . \qquad (۴۵.۳)$$

عبارت اول، در رابطه ۲۰.۳به صورت زیر تعریف شده است

$$\frac{\partial \Phi_{q.k}}{\partial I_{q.k}} = \alpha \Phi_{q.k}[۱ - \Phi_{q.k}] \ . \qquad (۴۶.۳)$$

عبارت دوم با مشتق گرفتن از رابطه۲۱.۳ نسبت به $\Phi_{p.j}$ بدست می‌آید

$$\frac{\partial \Phi_{q.k}}{\partial I_{p.j}} = \sum_{p=۱}^{n} w_{pq.k} \ . \qquad (۴۷.۳)$$

[۲۷] Minai williams

عبارت سوم نیز در رابطه ۳۴.۳به صورت زیر تعریف شده است

$$\frac{\partial \Phi_{p.j}}{\partial I_{p.j}} = \alpha \Phi_{p.j}[۱ - \Phi_{p.j}] \ . \qquad (۴۸.۳)$$

عبارت چهارم با گرفتن مشتق جزئی از رابطه ۳۵.۳نسبت به x_hبدست می‌آید

$$\frac{\partial I_{p.j}}{\partial x_h} = w_{hp.j} \ . \qquad (۴۹.۳)$$

مجدداً در اینجا علامت جمع از بین رفته است زیرا فقط یک ورودی مد نظر است. با جایگزینی این چهار عبارت در رابطه ۴۵.۳خواهیم داشت

$$\sigma_{q.h} = \alpha \Phi_{q.k}[۱ - \Phi_{q.k}] \sum_{p=۱}^{n} w_{pq.k} \alpha \Phi_{p.j}[۱ - \Phi_{p.j}] w_{hp.j} \ . \qquad (۵۰.۳)$$

که به این صورت در می‌آید

$$\sigma_{q.h} = \alpha^{۲} \Phi_{q.k}[۱ - \Phi_{q.k}] \sum_{p=۱}^{n} w_{pq.k} \Phi_{p.j}[۱ - \Phi_{p.j}] w_{hp.j} \ . \qquad (۵۱.۳)$$

۱.۱۴.۳ ارزیابی تجربی ضرایب حساسیت

پس از اینکه شبکه عصبی آموزش داده شد از یک روش تجربی برای ارزیابی ضرائب حساسیت می‌توان استفاده کرد که گاه روش تشکیک[۲۸] نامیده می‌شود. در این روش به هر ورودی x_i،آشفتگی کوچکی که حدود ۰/۵ درصد است هم در جهت مثبت و هم در جهت منفی اضافه می‌شود.سپس میزان آشفتگی که در هر دو جهت در خروجی y_i بوجود می‌آید اندازه گیری شده و از آنها میانگین گرفته می‌شود.ضرائب حساسیت بصورت زیر محاسبه می‌شود.

$$\sigma_{q.h} = \frac{\overline{\Delta y_q}}{\Delta x_h} \ . \qquad (۵۲.۳)$$

سپس هم روی آشفتگیهای مثبت و هم روی آشفتگیهای منفی میانگین گرفته می‌شود.اگر تعداد hورودی و q خروجی وجود داشته باشد در آنصورت تعدادhq ضریب حساسیت وجود دارد که باید بطور تجربی ارزیابی شود.

بطور کلی میزان سودمندی چنین معیارهایی محدود است.مشکل اصلی اینست که این مقادیر فقط برای قسمتهایی از فضای مساله معتبر هستند که آن قسمت ها مربوط به مقادیر ورودی و خروجی غیر آشفته هستند.

[۲۸] Dither method

۱۵.۳ مدل‌های شبکه عصبی ارائه شده پیشین برای حل مسائل بهینه‌سازی

بحث وبررسی در مورد کاربرد شبکه‌های عصبی مصنوعی در بهینه‌سازی از اوایل سال ۱۹۸۰ میلادی آغاز شده است. نتایج پژوهش‌های انجام شده از آن زمان تاکنون، مواردی چون برنامه‌ریزی خطی، برنامه‌ریزی درجه دوم، برنامه‌ریزی هندسی و برنامه‌ریزی غیرخطی را در برمی‌گیرد. ایده اصلی در استفاده از شبکه‌های عصبی مصنوعی برای مسائل بهینه سازی استفاده از یک تابع انرژی (نامنفی) و یک سیستم دینامیکی است که این دو بیان کننده‌ی مدل‌های شبکه‌های عصبی مصنوعی متناظر با مسائل بهینه‌سازی هستند. سیستم دینامیکی بیان شده معمولاً یک دستگاه معادلات دیفرانسیل غیرخطی مرتبه اول است. انتظار می‌رود که برای یک نقطهٔ آغازین نقطهٔ پایداری دستگاه معادلات دیفرانسیل غیرخطی به‌دست آمده، جواب بهینه مسأله بهینه‌سازی اصلی باشد.

یک اصل اساسی در استفاده از تابع انرژی این است که برای همگرایی دستگاه معادلات دیفرانسیل به نقطهٔ پایداری آن، تابع انرژی متناظر با آن حتماً باید نامنفی باشد. البته می‌توان مدل شبکه‌های عصبی نظیر مسائل بهینه سازی را حتی بدون در نظر گرفتن تابع انرژی نیز بیان کرد. بنابراین برای یک مدل متناظر با مسائل بهینه‌سازی، اصل اساسی استفاده از شبکه‌های عصبی در اینگونه مسائل به‌صورت زیر بیان می‌شود: **یک نقطه‌ی آغازین دلخواه، نقطه‌ی تعادل دستگاه معادلات دیفرانسیل غیرخطی بدست آمده جواب بهینه مسأله اصلی است و برعکس.**

مدل‌های مطرح شده متناظر مسائل مختلف بهینه‌سازی را می‌توان به دو قسمت مدل‌های دوگانی و مدل‌های جریمه‌ای تقسیم نمود. نظریه دوگانی و توابع جریمه‌ای دو نظریه بسیار مهم در مسائل بهینه‌سازی هستند که اکثر این مسائل بر مبنای این دو روش کلاسیک قابل حل هستند. در نظریه توابع جریمه‌ای معمولاً از مدل‌های گرادیانی برای معرفی مدل مورد نظر استفاده می‌شود، ولی در نظریه دوگانی از مدل‌های اولیه –دوگان برای حل این مسائل استفاده می‌شود. اگر بتوان متناظر با هر مسأله بهینه‌سازی، مدل دینامیکی مشخصی را ارائه داد که نقطه‌ی تعادل مدل ارائه شده برای حل آن مسأله شرایط لازم وکافی را برآورده سازد، آنگاه می‌توان متناظر با آن روش، یک مدل شبکه عصبی برای مسأله مورد نظر بسازیم. در زیر به بیان چند مدل که تاکنون برای مسائل برنامه‌ریزی خطی و درجه دوم و غیرخطی ارائه شده‌اند می‌پردازیم.

مدل اول

مسأله برنامه‌ریزی خطی زیر را در نظر بگیرید:

$$\text{minimize} \quad f(x) = a^T x \tag{۵۳.۳}$$

$$\text{subject to}$$

$$g(x) = Dx - b = ۰, \tag{۵۴.۳}$$

$$x \geq ۰, \tag{۵۵.۳}$$

که در آن $D \in \mathbb{R}^{m \times n}$ و $Rank(D) = m$ ، $x, a \in \mathbb{R}^m$ و $b \in \mathbb{R}^n$ است. دوگان (۵۳.۳) - (۵۵.۳) به‌صورت زیر بیان می‌شود:

$$\text{minimize} \quad \bar{f}(y) = b^T y \tag{۵۶.۳}$$

$$\text{subject to}$$

$$\bar{g}(y) = \bar{D}^T y - a \leq ۰, \tag{۵۷.۳}$$

$$y \in \mathbb{R}^n. \tag{۵۸.۳}$$

در [۳۱] مدل متناظر برای حل (۵۳.۳) - (۵۸.۳) بصورت زیر نمایش داده شده است:

$$\begin{cases} \dfrac{dx}{dt} = -[D^T(Dx - b) - \beta(\bar{D}^T y - a)], \\ \dfrac{dy}{dt} = -\beta[(Dx - \bar{D}^T y - a)^+ - b], \end{cases}$$

که در آن $\beta = \| (x + \bar{D}^T y - a)^+ - x \|_۲^۲$ و $(x, y)^T \in \{(x, y)^T \mid x \geq ۰\}$ و $(x + \bar{D}^T y - a)^+ = max\{x + \bar{D}^T y - a, ۰\}$.

مدل دوم

مسأله برنامه‌ریزی درجه دوم زیر را در نظر بگیرید:

$$\text{minimize} \quad f(x) = \frac{۱}{۲} x^T A x + a^T x \tag{۵۹.۳}$$

subject to

$$Dx - b = ۰, \tag{۶۰.۳}$$

$$x \geq ۰, \tag{۶۱.۳}$$

که در آن A یک ماتریس متقارن معین مثبت است. دوگان (۵۹.۳) - (۶۱.۳) بصورت زیر است:

$$\text{minimize} \quad \bar{f}(y) = b^T y - \frac{۱}{۲} x^T A x \tag{۶۲.۳}$$

subject to

$$\bar{D} y - A x - a \leq ۰. \tag{۶۳.۳}$$

در [۳۱] مدل متناظر برای حل (۵۹.۳) - (۶۳.۳) به‌صورت زیر نمایش داده شده است:

$$\begin{cases} \frac{dx}{dt} = -\{D^T(x-b) + \gamma(-D^T y + Ax + a) + \gamma A[x - (x + D^T y - Ax - a)^+]\}, \\ \frac{dy}{dt} = -\gamma\{Dx - b + D[(x + D^T y - Ax - a)^+ - x]\}, \end{cases} \tag{۶۴.۳}$$

که در آن $\gamma = \|(x + D^T y - Ax - a)^+ - x\|_۲^۲$ و $(x, y) \in \{(x, y) \mid x \geq ۰\}$.

مدل سوم

مسأله برنامه‌ریزی غیرخطی زیر را در نظر بگیرید:

$$\text{minimize} \quad f(x) \tag{۶۵.۳}$$

subject to

$$g(x) = [g_۱(x), g_۲(x), ..., g_m(x)]^T \leq ۰, \ i = ۱, ..., m. \tag{۶۶.۳}$$

که در آن $f(x), g(x) \in C^۲$ و $f(x), g_i(x) : \mathbb{R}^n \to \mathbb{R}^۱$. برای حل (۶۵.۳) و (۶۶.۳) هاپفیلد و تنک [۲۹] در [۷۶] یک مدل شبکه عصبی به‌صورت زیر ارائه دادند:

$$\dot{x} = C^{-۱}\{-\nabla f(x) - \nabla g(x) g^{+}(x) - \frac{۱}{s} Q^{-۱} x\} s,$$

که در آن C یک ماتریس قطری $n \times n$ بیان کننده ظرفیت هر نرون و Q یک ماتریس قطری $n \times n$ است که اعضای آن بصورت زیر تعریف می‌شوند:

$$q_{ij} = \frac{۱}{\frac{۱}{\rho_i} + \sum_{j=۱}^{m} -d_{ji}},$$

که در آن $\frac{۱}{\rho_i}$ ضریب هدایت گرمایی هر نرون، d_{ji} وزن‌هاي تخصیص داده شده به نرونهای داخلی و s پارامتر جریمه است. تابع انرژی متناظر به‌صورت زیر انتخاب شده است:

$$E_۱(x) = f(x) + \sum_{j=۱}^{m} [g_i^{+}]^۲ + \sum_{i=۱}^{n} \frac{x_i^۲}{۲ s r_{ii}}.$$

مدل چهارم

برای حل (۶۵.۳) کندی و چوآ [۳۰] به‌صورت زیر ارائه داده‌اند:

$$\dot{x} = C^{-۱}[-\nabla f(x) - \nabla g(x) g^{+}(x)], \qquad (۶۷.۳)$$

که در آن C و s شرایط مدل سوم را دارند. تابع انرژی متناظر این مدل به‌صورت زیر انتخاب شده است:

$$E_۲(x) = f(x) + \frac{s}{۲} \sum_{j=۱}^{m} [g_i^{+}]^۲.$$

مدل پنجم

مجدداً برای حل (۶۵.۳) توسط رودریگز و اسکوئز [۳۱] [۷۷] مدلی به‌صورت زیر ارائه شده است:

$$\dot{x} = -u_x \nabla f(x) - s \nabla g(x) g^{+}(x),$$

---

[۲۹] Hopfield and tank

[۳۰] Kennedy and Chua

[۳۱] Rodríguez

که در آن u_x متناظر اندیس‌های شدنی x است چنان‌که

$$u_x = \begin{cases} ۱, & g(x) \leq ۰, \\ ۰, & o.w. \end{cases}$$

تابع انرژی متناظر این مدل نیز به‌صورت زیر انتخاب شده است:

$$E_۳(x) = -u_x f(x) + \frac{s}{۲} \sum_{j=۱}^{m} [g_j^+(x)]^۲.$$

همان طور که ملاحظه می‌کنید، مدل‌های بیان شده بطور متوالی اصلاح شده‌اند و این به‌دلیل مشکلاتی بوده است که هرکدام از این مدل‌ها داشته‌اند.

مدل ششم

برنامه‌ریزی محدب درجه دوم [۳۲] زیر را در نظر بگیرید:

$$\text{minimize} \quad f(x) = \frac{۱}{۲} x^T Q x + D^T x \tag{۶۸.۳}$$

subject to

$$g(x) = Ax - b \leq ۰, \tag{۶۹.۳}$$

$$h(x) = Ex - f = ۰. \tag{۷۰.۳}$$

که در آن $Q \in \mathbb{R}^{n \times n}$ یک ماتریس متقارن معین مثبت، $A \in \mathbb{R}^{m \times n}$، $b \in \mathbb{R}^m$ ، $E \in \mathbb{R}^{l \times n}$، $f \in \mathbb{R}^l$ و $x \in \mathbb{R}^n$ است.

بر طبق شرایط کاروش- کان- تاکر، $x^* \in \mathbb{R}^n$ جواب بهینه (۶۸.۳) - (۷۰.۳) است اگر وتنها اگر بردارهای $u^* \in \mathbb{R}^m$ و $v^* \in \mathbb{R}^l$ وجود داشته باشند به‌طوری‌که در رابطه زیر صدق کنند:

$$\begin{cases} u^* \geq ۰, Ax^* - b \leq ۰, u^{*T}(Ax^* - b) = ۰, \\ Qx^* + D + A^T u^* + E^T v^* = ۰, \\ Ex^* - f = ۰. \end{cases} \tag{۷۱.۳}$$

[۳۲] Convex quadratic programming

برای حل مسأله (۶۸.۳)-(۷۰.۳) تابع زیر را که معروف به تابع فیشر-برمیستر [۳۳] [۶۴] است، معرفی می‌کنیم:

$$\begin{cases} \phi : \mathbb{R}^{۲} \to \mathbb{R}^{۱}, \\ \phi(a,b) = \sqrt{a^{۲} + b^{۲}} - a - b. \end{cases}$$

قضیه ۱.۳. [۶۱] :

۱. $\phi(a,b) = ۰$ *اگر وفقط اگر* $a \geq ۰, b \geq ۰, ab = ۰$.

۲. *مربع* $\phi(a,b)$ *بطور پیوسته مشتق پذیر باشد.*

۳. $\phi(a,b)$ *همه جا به جز در مبدأ دو بار بطور پیوسته مشتق پذیر است، ولی در مبدأ قویاً هموار* [۳۴] *است.*

تابع ϕ *که در شرط* (۱) *قضیه* (۱.۳) *صدق کند، مسأله مکمل غیرخطی* [۳۵] *نامیده می‌شود.*

متناظر با مسائل (۶۸.۳) - (۷۰.۳) تابع زیر را در نظر می‌گیریم:

$$\phi(x,u,v) = \begin{bmatrix} \phi(\alpha, Qx + D + A^T u + E^T v) \\ \phi(u, Ax - b) \\ \phi(\beta, Ex - f) \end{bmatrix}, \qquad (۷۲.۳)$$

که در آن $\alpha = (\alpha_{۱}, ..., \alpha_m)^T > ۰$، $u = (u_{۱}, ..., u_m)$ و $\beta = (\beta_{۱}, ..., \beta_l)^T > ۰$ است. مدل شبکه عصبی متناظر با (۷۲.۳) به‌صورت زیر بیان شده است:

$$\begin{cases} \dfrac{dy}{dt} = -\tau \bigtriangledown E(y(t)), \quad \tau > ۰, \\ y(t_{۰}) = y_{۰} \in \mathbb{R}^{n+m+l}, \end{cases} \qquad (۷۳.۳)$$

که در آن $y = (x^T, u^T, v^T)^T$ و τ ضریبی برای افزایش سرعت همگرایی و کاهش تعداد تکرارها در روش عددی انتخاب شده برای حل دینامیکی (۷۲.۳) است. تابع انرژی متناظر این مدل به‌صورت

[۳۳] Fisher-bermister

[۳۴] Strongly smooth

[۳۵] Nonlinear complementarity problem(NCP)

زیر انتخاب شده است:

$$E(y) = \frac{۱}{۲}\| \phi(y) \|^{۲}.$$

مدل هفتم

برای طراحی مدل هفتم، مجدداً شرایط کاروش- کان- تاکر را برای حل مسائل (۶۸.۳) - (۷۰.۳) به‌صورت زیر در نظر می‌گیریم:

$$\begin{cases} (u + Ax - b)^{+} = u, \\ Qx + D + A^{T}u + E^{T}v = ۰, \\ Ex - f = ۰. \end{cases}$$

آنگاه برای حل (۶۸.۳) - (۷۰.۳) دستگاه زیر تعریف می‌شود:

$$U(x,u,v) = \begin{bmatrix} -(Qx + D + A^{T}u + E^{T}v), \\ (u + Ax - b)^{+} - u, \\ (Ex - f). \end{bmatrix}. \tag{۷۴.۳}$$

مدل شبکه عصبی برای حل (۶۸.۳) - (۷۰.۳) در [۶۱] به‌صورت زیر بیان شده است:

$$\begin{cases} \dfrac{dy}{dt} = \kappa U(y), \\ y(t_{۰}) = y_{۰}. \end{cases} \tag{۷۵.۳}$$

که در آن κ ضریب همگرایی سیستم است.

مدل هشتم

مسأله برنامه ریزی درجه دوم (۶۸.۳) - (۷۰.۳) را در نظر بگیرید. که در آن فرض می‌کنیم $Q \in \mathbb{R}^{n\times n}$ یک ماتریس نیمه معین مثبت است. برای حل مسئله (۶۸.۳) - (۷۰.۳) در [۶۲]

یک مدل شبکه عصبی به‌صورت زیر ارائه می‌شود:

$$\frac{dy}{dt} = \kappa(I + M^T) \begin{bmatrix} -(Qx + D + A^T u + E^T v), \\ (u + Ax - b)^+ - u, \\ (Ex - f). \end{bmatrix}, \tag{۷۶.۳}$$

که در آن κ نرخ همگرایی است و

$$M = \begin{bmatrix} -Q & A^T & E^T \\ A & \circ_{m\times m} & \circ_{m\times l} \\ E & \circ_{l\times m} & \circ_{l\times l} \end{bmatrix}.$$

مدل نهم

مسأله برنامه‌ریزی درجه دوم زیر را در نظر بگیرید:

$$\text{minimize} \quad f(x) = \frac{۱}{۲} x^T A x + a^T x \tag{۷۷.۳}$$

subject to

$$Dx \leq b, \tag{۷۸.۳}$$

$$x \geq \circ. \tag{۷۹.۳}$$

دوگان آن عبارت است از:

$$\text{minimize} \quad \bar{f}(w) = b^T w - \frac{۱}{۲} x^T A x \tag{۸۰.۳}$$

subject to

$$\bar{D}^T w - Ax \leq a, \tag{۸۱.۳}$$

$$w \geq \circ, \tag{۸۲.۳}$$

که در آن $A \in \mathbb{R}^{n\times n}$ متقارن و نیمه معین مثبت است، $a \in \mathbb{R}^n$ ، $b \in \mathbb{R}^m$ و $D \in \mathbb{R}^{m\times n}$ است. با توجه به شرایط کاروش- کان- تاکر برای مسائل برنامه‌ریزی محدب، $(x^{*T}, w^{*T})^T$ به‌ترتیب یک جواب بهینه برای مسائل (۷۷.۳) - (۷۹.۳) است اگر و فقط اگر $(x^{*T}, w^{*T})^T$ در شرایط کاروش- کان- تاکر زیر صدق کند:

$$\begin{cases} w^* \geq 0, Dx^* - b \leq 0, w^{*T}(Dx^* - b) = 0, \\ Ax^* + a + D^T w^* \geq 0, x \geq 0, \\ x^{*T}(Ax^* + a + D^T w^*) = 0. \end{cases} \tag{۸۳.۳}$$

مدل متناظر با (۷۷.۳) و (۸۰.۳) در [۶۲] به‌صورت زیر بیان شده است:

$$\dot{u} = B(I + M^T)\{(u - (Mu + q)^+ - u\},$$

که در آن

$$u = \begin{bmatrix} x \\ w \end{bmatrix}, q = \begin{bmatrix} a \\ b \end{bmatrix}, M = \begin{bmatrix} A & D^T \\ -D & 0 \end{bmatrix}.$$

M یک ماتریس نیمه معین مثبت است زیرا $u^T M u = x^T A x \geq 0$. همچنین $B = \lambda I$ است که $\lambda > 0$ و با افزایش λ سرعت همگرایی افزایش می‌یابد.

فصل ۴

مبانی و تعاریف ماشین بردار پشتیبان

۱.۴ کاربرد های SVM

ماشین بردار پشتیبانی [۱] یکی از روش‌های یادگیری بانظارت است که از آن برای طبقه‌بندی و رگرسیون استفاده می‌کنند. این روش از جمله روش‌های نسبتاً جدیدی است که در سال‌های اخیر کارایی خوبی نسبت به روش‌های قدیمی‌تر برای طبقه‌بندی از جمله شبکه‌های عصبی پرسپترون استفاده شده است. مبنای کاری دسته‌بندی کننده SVM دسته‌بندی خطی داده‌ها است. در تقسیم خطی داده‌ها سعی می‌کنیم خطی را انتخاب کنیم که حاشیه اطمینان بیشتری داشته باشد. حل معادله پیدا کردن خط بهینه برای داده‌ها به وسیله روش‌های برنامه ریزی درجه دوم که روش‌های شناخته شده‌ای در حل مسائل محدودیت‌دار هستند صورت می‌گیرد. قبل از تقسیمِ خطی برای اینکه ماشین بتواند داده‌های با پیچیدگی بالا را دسته‌بندی کند داده‌ها را به وسیله تابعِ phi به فضای با ابعاد خیلی بالاتر می‌بریم. برای اینکه بتوانیم مساله ابعاد خیلی بالا را با استفاده از این روش‌ها حل کنیم از قضیه دوگانی لاگرانژ برای تبدیلِ مساله مینیمم‌سازی مورد نظر به فرم دوگانی آن که در آن به جای تابع پیچیده phi که ما را به فضایی با ابعاد بالا می‌برد، تابعِ ساده‌تری به نامِ تابع هسته که ضرب برداری تابع phi است ظاهر می‌شود استفاده می‌کنیم. از توابع هسته مختلفی از جمله هسته‌های نمایی، چندجمله‌ای و سیگموید می‌توان استفاده کرد[۱] . الگوریتم SVM اولیه در ۱۹۶۳ توسط ولادیمر واپنیک [۲] ابداع شد و در

[۱] Support vector machine

[۲] Vladimir Vapnik

سال ۱۹۹۵ توسط Vapnik و کورینا کورتس[۳] برای حالت غیرخطی تعمیم داده شد.

۲.۴ خلاصه استفاده عملی از SVM

ماتریس الگو را آماده می کنیم. تابع کرنلی را برای استفاده انتخاب می کنیم. پارامتر تابع کرنل و مقدار C را انتخاب می کنیم. برای محاسبه ی مقادیر a_i الگوریتم آمورشی را با استفاده از حل کننده های برنامه ریزی درجه دوم اجرا می کنیم. داده های جدید با استفاده از مقادیر a_i و بردارهای پشتیبان می توانند دسته بندی شوند، که دارای مزایا و معایبی از جمله آموزش نسبتا ساده است و برخلاف شبکه های عصبی در ماکزیمم های محلی گیر نمی افتد. برای داده های با ابعاد بالا تقریبا خوب جواب می دهد. مصالحه بین پیچیدگی دسته بندی کننده و میزان خطا به طور واضح کنترل می شود. به یک تابع کرنل خوب و انتخاب پارامتر C نیاز داریم. الگوریتم های مبتنی بر رگرسیون به صورت‌های زیر تقسیم می‌شوند.

۱. با ناظر

- رگرسیون خطی
- شبکه‌های عصبی
- رگرسیون فرایندهای گوسی[۴]

۲. بدون ناظر

- تحلیل مولفه‌های اصلی[۵]
- LCA

الگوریتم های کلاس بندی(الگوریتم های با ناظر پیشگو)

۱. درخت تصمیم و لیست تصمیم

۲. ماشین های بردار پشتیبانی

---

[۳] Corinna cortes

[۴] Regression Process Gaussian

[۵] Principal Components Analysis = PCA

۳. شبکه های عصبی

۴. پرسپترون $k-$نزدیکترین همسایگی

۳.۴ ماشین بردار پشتیبان خطی

ما مجموعه داده آموزشی D شامل n عضو را در اختیار داریم که به صورت زیر تعریف می شود.

$$D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^p, y_i \in \{-۱, ۱\}\}_{i=۱}^{n},$$

که در آن y برابر ۱ یا-۱ و هر x_i یک بردار حقیقی $p-$بعدی است. هدف پیدا کردن ابرصفحه جداکننده با بیشترین فاصله از نقاط حاشیه ای است که نقاط با$y_i = ۱$ را از نقاط با $y_i = -۱$ جدا کند. هر ابر صفحه می تواند به صورت مجموعه ای از نقاط x که شرط زیر را صدق می کند نوشت:

$$w.x - b = ۰.$$

که در آن". "علامت ضرب است.w بردار نرمال است، که به ابرصفحه عمود است. ما می خواهیم w و b را طوری انتخاب کنیم که بیشترین فاصله بین ابر صفحه های موازی که داده ها را از هم جدامی کنند، ایجاد شود. این ابرصفحه ها با استفاده از رابطه زیر توصیف می شوند:

$$w.x - b = ۱,$$

و

$$w.x - b = -۱.$$

اگر داده های آموزشی جدایی پذیر خطی باشند، ما می توانیم دو ابر صفحه در حاشیه نقاط به طوری که هیچ نقطه مشترکی نداشته باشند، در نظر بگیریم و سپس سعی کنیم، فاصله آنها را ماکسیمم کنیم. با استفاده از هندسه، فاصله این دو صفحه $\frac{۲}{\|w\|}$ است. بنابراین ما باید $\|w\|$ را مینیمم کنیم. برای اینکه از ورود نقاط به حاشیه جلو گیری کنیم، شرایط زیر را اضافه می کنیم:

$$w^T.x_i - b \geq ۱, \qquad \textit{برای } x_i \textit{ ها در اولین دسته}$$

یا

$$w^T.x_i - b \leq -۱, \qquad \textit{برای } x_i \textit{ ها در دومین دسته}$$

این می تواند به صورت زیر نوشته شود:

$$y_i(w^T.x_i - b) \geq ۱, \qquad ۱ \leq i \leq n. \tag{۱.۴}$$

با کنار هم قرار دادن این دو یک مسئله بهینه سازی به صورت زیر دست می آید [۵۰] :

$$\text{minimize} \quad c(w,b) = \|w\|$$

$$\text{subject to}$$

$$y_i^T(w.x_i - b) \geq ۱, \quad i = ۱, \cdots, n.$$

فرم اولیه مسئله بهینه سازی مشاهده شده در قسمت قبل، مسئله سختی، برای حل کردن است، زیرا به$\|w\|$وابسته است . خوشبختانه می‌توانیم، بدون تغییر در مسئله $\|w\|$را با $\frac{۱}{۲}\|w\|^۲$جایگزین کنیم (عبارت $\frac{۱}{۲}$ برای آسودگی در محاسبات ریاضی آورده می‌شود). این یک مساله بهینه سازی برنامه ریزی غیر خطی است. به طور واضح تر :

$$\text{minimize} \qquad c(w,b) = \frac{۱}{۲}\|w\|^۲$$

$$\text{subject to}$$

$$y_i^T(w.x_i - b) \geq ۱, \quad i = ۱, \cdots, n.$$

می توان عبارت قبل را با استفاده از ضرایب نا منفی لاگرانژ به صورت زیر نوشت که در آن α_i ها ضرایب لاگرانژ هستند :

$$\underset{w,b,\alpha}{\text{minimize}}\{\frac{۱}{۲}\|w\|^۲ - \sum_{i=۱}^{n}\alpha_i[y_i^T(w.x_i - b) - ۱]\}.$$

فرض کنید ما بتوانیم خانواده‌ای از ابرصفحات که نقاط را دسته‌بندی می کنند پیدا کنیم، پس $\alpha_i[y_i^T(w.x_i - b) - ۱] \geq ۱$. بنابراین ما می توانیم مینیمم را با میل دادن همه α_iها به ∞ پیدا کنیم. با این حال می تواند به صورت زیر بیان شود:

$$\underset{w,b}{\text{minimize}} \quad \underset{\alpha}{\text{maximize}}\{\frac{۱}{۲}\|w\|^۲ - \sum_{i=۱}^{n}\alpha_i[y_i^T(w.x_i - b) - ۱]\}.$$

ما به دنبال نقاط زینی هستیم. حالا می‌توان این مسئله را به کمک برنامه ریزی غیرخطی استاندارد حل کرد. جواب می تواند به صورت ترکیب خطی از بردارهای آموزشی [۶] زیر بیان شود :

$$w = \sum_{i=۱}^{n}\alpha_i y_i x_i.$$

[۶] Learning vector

تنها برخی α_i ها بزرگتر از صفر خواهد بود. x_i متناظر، دقیقا همان بردار پشتیبان خواهد بود و در شرط بالا صدق خواهد کرد. از این رو می توان نتیجه گرفت که بردارهای پشتیبان شرط زیر را نیزدارا هستند: $y_i^T(w.x_i - b) = ۱$ که اجازه می دهد مقدار b تعریف شود. در عمل الگوریتم مقاوم‌تر خواهد بود اگر از تمام N_{SV} بردار پشتیبان میانگین گرفته شود:

$$b = \frac{۱}{N_{SV}} \sum_{i=۱}^{N_{SV}} (w.x_i - y_i).$$

فرم دوگان با استفاده از این واقعیت که $\|w\|^۲ = w.w$ و جانشینی $w = \sum_{i=۱}^{n} \alpha_i y_i x_i$ می توان نشان داد که دوگان SVM به صورت مساله بهینه سازی زیر داده می شود:

$$\begin{aligned}
&\text{minimize} \quad \alpha_i \\
&\text{subject to} \\
&\hat{L}(\alpha) = \sum_{i=۱}^{n} \alpha_i - \frac{۱}{۲} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
&= \sum_{i=۱}^{n} \alpha_i - \frac{۱}{۲} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j), \qquad (۲.۴) \\
&i = ۱, \cdots, n, \\
&\alpha_i \geq ۰, \\
&\sum_{i=۱}^{n} \alpha_i y_i = ۰.
\end{aligned}$$

در اینجا هسته به صورت $k(x_i, x_j) = x_i.x_j$ تعریف می شود. عبارت α تشکیل یک دوگان برای بردار وزن ها مجموعه آموزشی می دهد:

$$\sum_i \alpha_i y_i x_i.$$

۴.۴ ماشین بردار پشتیبان چند کلاسی

SVM اساسا یک جداکننده دودویی است. در بخش قبلی پایه های تئوری ماشین های بردار پشتیبان برای دسته بندی دو کلاس تشریح شد. یک تشخیص الگوی چند کلاسی می تواند به وسیله ی ترکیب ماشین های بردار پشیبان دو کلاسی حاصل شود. به طور معمول دو دید برای این هدف وجود دارد. یکی از آنها استراتژی "یک در مقابل همه " برای دسته بندی هر جفت کلاس و کلاس

های باقی مانده است. دیگر استراتژی "یک در مقابل یک" برای دسته بندی هر جفت است. در شرایطی که دسته بندی اول به دسته بندی مبهم منجر می شود.برای مسائل چند کلاسی , رهیافت کلی کاهش مساله‌ی چند کلاسی به چندین مساله دودویی است. هریک از مسائل با یک جداکننده دودویی حل می شود. سپس خروجی جداکننده های دودویی SVM با هم ترکیب شده و به این ترتیب مساله چند کلاس حل می شود.

۱.۴.۴ ماشین های بردار پشتیبان غیرخطی

ابرصفحه جداکننده بهینه اولین بار توسط واپنیک [۷] در سال ۱۹۶۳ ارائه شد که یک دسته کننده خطی بود. در سال ۱۹۹۲ ، برنهارد بزر[۸]، ایسابل گوین [۹]و واپنیک راهی را برای ایجاد دسته بند غیرخطی، با استفاده قرار دادن هسته برای پیدا کردن ابرصفحه با حاشیه بیشتر، پیشنهاد دادند. الگوریتم نتیجه شده ظاهرا مشابه است، به جز آنکه تمام ضرب های نقطه ای با یک تابع هسته غیرخطی جایگزین شداند. این اجازه می‌دهد، الگوریتم برای ابرصفحه با بیشترین حاشیه در یک فضای ویژگی تغییرشکل داده، مناسب باشد. ممکن است، تغییرشکل غیرخطی باشد و فضای تغییر یافته، دارای ابعاد بالاتری باشد. به هر حال دسته کننده، یک ابرصفحه در فضای ویژگی با ابعاد بالا است، که ممکن است در فضای ورودی نیز غیرخطی باشد.
اگر از هسته با تابع گوسیین استفاده شود، فضای ویژگی متناظر، یک فضای هیلبرت نامتناهی است. دسته کننده‌ی بیشترین حاشیه، خوش ترتیب است، بنابراین ابعاد نامتناهی، نتیجه را خراب نمی کند. هسته های متداول به صورت زیر هستند:

- چند جمله‌ای (همگن): $k(x_i, x_j) = (x_i.x_j)^d$ ،
- چند جمله‌ای (ناهمگن): $k(x_i, x_j) = (x_i.x_j + ۱)^d$ ،
- گوسیین :برای $\gamma > ۰$ $k(x_i, x_j) = e^{(-\gamma \|x_i - x_j\|^۲)}$ ،
- تانژانت هذلولوی : برای $c < ۰$ $k > ۰$ $k(x_i, x_j) = \tanh(kx_i.x_j + c)^d$.

هسته با انتقال $\varphi(x_i)$ با تساوی $k(x_i, x_j) = \varphi(x_i).\varphi(x_j)$در ارتباط است. همچنین مقدار w در فضای انتقال یافته برابر$w = \sum_i \alpha_i y_i \varphi(x_i)$ است. ضرب نقطه ای با w می تواند توسط هسته

[۷] Vapnik

[۸] Bernhard boser

[۹] Isabelle guyon

محاسبه شود یعنی$w.\varphi(x) = \sum_i \alpha_i y_i k(x_i, x)$. به هر حال در حالت عادی w'وجود ندارد، به طوری که $w.\varphi(x) = k(w', x)$.

یادگیری ماشینی به عنوان یکی از شاخه‌های وسیع و پرکاربرد هوش مصنوعی است. یادگیری ماشینی [۱۰] به تنظیم و اکتشاف شیوه‌ها و الگوریتم‌هایی می‌پردازد که بر اساس آنها رایانه‌ها و سامانه‌ها توانایی تعلم و یادگیری پیدا می‌کنند.

۲.۴.۴ رگرسیون بردار پشتیبان [۱۱]

مسئله تقریب یک مجموعه از داده های

$$\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}.$$

با تابع رگرسیونی [۱۲]به شکل

$$\phi(x) = \sum_{i=1}^{m} \alpha_i \Phi_i(x) + b. \qquad (۳.۴)$$

را در نظر می‌گیریم که در آن $\{\Phi_i(x)\}_{i=1}^m$ توابع مشخصه در فضای ابعادی بالا نامیده می‌شود.b و $\{\alpha_i\}_{i=1}^m$ پارامترهای مدل هستند که باید تخمین زده شوند. یک SVR پیدا کردن پارامترهای مجهول توسط مینیمم کردن تابع مخاطره منظم [۱۳] زیر است.

$$R_{emp}(\alpha) = \frac{1}{n} \sum_{i=1}^{n} L(\phi(x_i) - y_i) + c\|\alpha\|. \qquad (۴.۴)$$

که در آن $L(\phi(x) - y)$ تابع کمبود نامیده می‌شوند که اختلاف بین $\phi(x)$ و y را نشان می‌دهد. در اینجا ما از تابع کمبود هابر[۱۴] به صورت زیر استفاده می‌کنیم.

$$L(\phi(x) - y) = \begin{cases} \frac{1}{2}(\phi(x, \alpha) - y)^2, & if|\phi(x, \alpha) - y)| < \varepsilon, \\ \varepsilon|\phi(x, \alpha) - y)| - \frac{\varepsilon}{2}, & \text{otherwise.} \end{cases}$$

که در آن ε ثابت مقیاس نامیده می‌شود. اگر تابع کرنل $K(x, y)$ در شرط

---

[۱۰] Machine learning

[۱۱] Support vector regression

[۱۲] Regression function

[۱۳] Regularized risk function

[۱۴] Huber loos function

$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$صدق کند، آنگاه تابع رگرسیون بصورت زیر نمایش داده می‌شود:

$$\phi(x) = \sum_{i=۱}^{n} (\zeta_i - \alpha_i) K(x, x_i) + b, \tag{۵.۴}$$

که در آن ζ_i و α_i جواب‌های بهینه مسئله برنامه ریزی درجه دو زیر هستند:

$$\begin{aligned} \text{maximize} \; & -\tfrac{۱}{۲} \sum_{i=۱}^{n} \sum_{j=۱}^{n} (\alpha_i - \zeta_i)(\alpha_j - \zeta_j) K(x_i, x_j) \\ & + \sum_{i=۱}^{n} (\alpha_i - \zeta_i) y_i - \tfrac{\varepsilon}{۲c} \sum_{i=۱}^{n} [(\alpha_i)^۲ + (\zeta_i)^۲] \end{aligned} \tag{۶.۴}$$

$$\text{subject to} \begin{cases} \sum_{i=۱}^{n} (\zeta_i - \alpha_i) = ۰, \\ ۰ \le \zeta_i, \alpha_i \le c, \qquad i = ۱, \cdots, n. \end{cases} \tag{۷.۴}$$

با استفاده از خاصیت مکمل نتیجه می‌شود$\zeta_i \alpha_i = ۰$ و بنابراین داریم $(\alpha_i)^۲ + (\zeta_i)^۲ = (\alpha_i - \zeta_i)^۲$. آنگاه مساله بهینه سازی(۶.۴) و (۷.۴)به طور معادل بصورت زیر بیان شود.

$$\begin{aligned} \text{minimize} \; & \tfrac{۱}{۲} \sum_{i=۱}^{n} \sum_{j=۱}^{n} (\alpha_i - \zeta_i)(\alpha_j - \zeta_j) K(x_i, x_j) \\ & - \sum_{i=۱}^{n} (\alpha_i - \zeta_i) y_i + \tfrac{\varepsilon}{۲c} \sum_{i=۱}^{n} (\alpha_i - \zeta_i)^۲ \end{aligned} \tag{۸.۴}$$

$$\text{subject to} \begin{cases} \sum_{i=۱}^{n} (\zeta_i - \alpha_i) = ۰, \\ ۰ \le \zeta_i, \alpha_i \le c, \qquad i = ۱, \cdots, n. \end{cases} \tag{۹.۴}$$

اکنون اگر $\theta_i = \alpha_i - \zeta_i$ آنگاه $-c \le \theta_i \le c$و (۸.۴)و (۹.۴)به صورت زیر نوشته می‌شود.

$$\text{minimize} \; \tfrac{۱}{۲} \sum_{i=۱}^{n} \sum_{j=۱}^{n} \theta_i \theta_j K(x_i, x_j) - \sum_{i=۱}^{n} \theta_i y_i + \tfrac{\varepsilon}{۲c} \sum_{i=۱}^{n} (\theta_i)^۲ \tag{۱۰.۴}$$

$$\text{subject to} \begin{cases} \sum_{i=۱}^{n} \theta_i = ۰, \\ -c \le \theta_i \le c, \qquad i = ۱, \cdots, n. \end{cases} \tag{۱۱.۴}$$

بنابراین مساله یادگیری SVR [۱۵] معادل مساله برنامه‌ریزی درجه‌دوم در (۱۰.۴)و (۱۱.۴) با nمتغیر کراندار است، که در آن $\varepsilon > ۰$پارامتر دقت مورد نیاز برای تقریب است و $\zeta > ۰$ یک پارامتر از پیش تعیین شده است. همچنین در [۲]نشان داده شده است که $\beta = v^*$ ، که v^*نقطه تعادل مدل شبکه عصبی پیشنهادی برای حل مساله بهینه سازی درجه دوم (۱۰.۴)و (۱۱.۴) است.

۵.۴ یک مدل شبکه عصبی

طبق تحلیل بالا، به نظر می‌رسد که مساله رگرسیون بردار پشتیبان یک مساله برنامه‌ریزی درجه دوم است. بنابراین یک شکل کلی از مسایل برنامه‌ریزی درجه دوم را برسی می‌کنیم.

$$\text{minimize} \quad \frac{۱}{۲}x^TQx + D^Tx \tag{۱۲.۴}$$

subject to

$$Ax - b \leq ۰, \tag{۱۳.۴}$$

$$Ex - f = ۰, \tag{۱۴.۴}$$

که در آن $Q \in \mathbb{R}^{n\times n}$ یک ماتریس نیمه معین مثبت و متقارن است، $A \in \mathbb{R}^{m\times n}$ ، $b \in \mathbb{R}^m$ ، $f \in \mathbb{R}^l$, $E \in \mathbb{R}^{l\times n}$, $x \in \mathbb{R}^n$ و $(۰ \leq m, l < n)$ $\operatorname{rank}(A, E) = m + l$.

قضیه ۱.۴ [۹] *$x^* \in \mathbb{R}^n$یک جواب بهینه (۱۲.۴)-(۱۴.۴) است اگر و فقط اگر $u^* \in \mathbb{R}^m$ و $v^* \in \mathbb{R}^l$وجود داشته باشند به طوری که $(x^{*T}, u^{*T}, v^{*T})^T$ در شرایط کروش-کان-تاکر زیر صدق کند.*

$$\begin{cases} u^* \geq ۰, \quad Ax^* - b \leq ۰, \quad u^{*T}(Ax^* - b) = ۰, \\ Qx^* + D + A^Tu^* + E^Tv^* = ۰, \\ Ex^* - f = ۰. \end{cases} \tag{۱۵.۴}$$

x^*یک نقطه KKT (۱۲.۴)-(۱۴.۴) نامیده می‌شود و جفت$(u^{*T}, v^{*T})^T$ بردار ضرایب لاگرانژ متناظر با x^* نامیده می‌شوند.

[۱۵] Support vector regression

قضیه ۲.۴. [۹] *x^* یک جواب بهینه (۱۲.۴)-(۱۴.۴) است اگر و فقط اگر x^* یک نقطه KKT از (۱۲.۴)-(۱۴.۴) باشد.*

حال فرض کنید $x(.)$, $u(.)$ و $v(.)$ متغیرهای وابسته به زمان باشند. هدف ساختن یک سیستم دینامیکی زمان-پیوسته است که بر نقطه KKT مساله (۱۲.۴)-(۱۴.۴)منطبق خواهد شد. یک مدل شبکه عصبی برای حل (۱۲.۴)-(۱۴.۴) و دوگان آن پیشنهاد می‌کنیم که معادله دینامیکی آن به صورت زیر تعریف می‌شود.

$$\frac{dx}{dt} = -\left(Qx + D + A^T(u + Ax - b)^+ + E^T v\right), \tag{۱۶.۴}$$

$$\frac{du}{dt} = (u + Ax - b)^+ - u, \tag{۱۷.۴}$$

$$\frac{dv}{dt} = Ex - f, \tag{۱۸.۴}$$

با نقطه اولیه $(x_۰^T, u_۰^T, v_۰^T)^T$ که

$$(u + Ax - b)^+ = \left([(u + Ax - b)_۱]^+, [(u + Ax - b)_۲]^+, ..., [(u + Ax - b)_m]^+\right),$$
$$[(u + Ax - b)_k]^+ = \max\{(u + Ax - b)_k, ۰\},\ k = ۱, ۲, ..., m.$$

برای سادگی فرض کنید، $y = (x^T, u^T, v^T)^T \in \mathbb{R}^{n+m+l}$ و D^*را به عنوان مجموعه جواب‌های بهینه (۱۲.۴)-(۱۴.۴)و دوگانش در نظر گرفته و تعریف می‌کنیم

$$\Psi(y) = \begin{pmatrix} -\left(Qx + D + A^T(u + Ax - b)^+ + E^T v\right) \\ (u + Ax - b)^+ - u \\ Ex - f \end{pmatrix}. \tag{۱۹.۴}$$

با به کار گیری یک عامل مقیاس ، مدل شبکه عصبی (۱۶.۴)-(۱۸.۴) می‌تواند به صورت زیر نوشته شود:

$$\frac{dy}{dt} = \eta\Psi(y), \tag{۲۰.۴}$$

$$y(t_۰) = y_۰,\ \eta > ۰, \tag{۲۱.۴}$$

که در آن ηیک پارامتر مقیاس است و نرخ همگرایی شبکه عصبی (۲۰.۴) و (۲۱.۴) را نشان می‌دهد. برای سادگی تحلیل خود، فرض کنیم $\eta = ۱$ باشد. یک شاخص که چگونه شبکه عصبی (۲۰.۴) و (۲۱.۴)بر روی سخت افزار می‌تواند اجرا شود در شکل ۱ نشان داده شده است.

۶.۴ خواص همگرایی و پایداری

قضیه ۳.۴. *فرض کنید $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ نقطه تعادل شبکه عصبی (۲۰.۴) و (۲۱.۴)باشد آنگاه x^* یک نقطه* KKT *مساله (۱۲.۴)-(۱۴.۴) است. برعکس، اگر $x^* \in \mathbb{R}^n$یک جواب بهینه مساله (۱۲.۴)-(۱۴.۴)باشد، آنگاه$u^* \in \mathbb{R}^m$ و $v^* \in \mathbb{R}^l$ وجود دارند به طوری که $y^* = (x^{*T}, u^{*T}, v^{*T})^T$یک نقطه تعادل شبکه عصبی پیشنهادی (۲۰.۴) و (۲۱.۴)است .*

اثبات. فرض کنید $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ نقطه تعادل (۲۰.۴) و (۲۱.۴) باشد آنگاه $\frac{dx^*}{dt} = ۰$. $\frac{du^*}{dt} = ۰$ و $\frac{dv^*}{dt} = ۰$

به سادگی نتیجه می‌شود که

$$Qx^* + D + A^T(u^* + Ax^* - b)^+ + E^T v^* = ۰, \tag{۲۲.۴}$$

$$(u^* + Ax^* - b)^+ - u^* = ۰, \tag{۲۳.۴}$$

$$Ex^* - f = ۰. \tag{۲۴.۴}$$

واضح است که $(u^* + Ax^* - b)^+ = u^*$اگر و فقط اگر

$$u^* \geq ۰, \quad Ax^* - b \leq ۰, \quad u^{*T}(Ax^* - b) = ۰. \tag{۲۵.۴}$$

بعلاوه، با جایگذاری (۲۳.۴) در (۲۲.۴)داریم

$$Qx^* + D + A^T u^* + E^T v^* = ۰. \tag{۲۶.۴}$$

از (۲۴.۴)-(۲۶.۴)می‌توان دید که $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ در شرایط KKT صدق می‌کند. اثبات عکس قضیه واضح است. □

لم ۱.۴. *برای هر نقطه اولیه $y(t_۰) = (x(t_۰)^T, u(t_۰)^T, v(t_۰)^T)^T$ ، یک جواب پیوسته منحصر*

به فرد $y(t) = (x(t)^T, u(t)^T, v(t)^T)^T$ *برای سیستم* (۲۰.۴) *و* (۲۱.۴) *وجود دارد.*

اثبات. به سادگی می‌توان تحقیق کرد که
$(u + Ax - b)^+ - u$ $Qx + D + A^T(u + Ax - b)^+ + E^T v$, و $Ex - f$ بر روی مجموعه محدب باز $D \subseteq \mathbb{R}^{n+m+l}$ پیوسته لیپ شیتز محلی هستند. بر اساس قضیه وجود یکتایی جواب یک معادله دیفرانسیل [۵۹] ، شبکه عصبی (۲۰.۴) و (۲۱.۴) یک جواب پیوسته منحصر به فرد $y(t), t \in [t_0, \tau)$ برای $\tau > t_0$ وقتی که $\tau \to \infty$ دارد. □

لم ۲.۴. *ماتریس ژاکوبین* $\nabla\Psi(y)$ *از نگاشت* Ψ *که در* (۱۹.۴) *تعریف شده است یک ماتریس نیمه معین منفی است.*

اثبات. بدون کاسته شدن از کلیت، فرض کنید که $0 < p < m$ به طوری وجود داشته باشد که

$$(u + Ax - b)^+ = ((u + Ax - b)_1, (u + Ax - b)_2, ..., (u + Ax - b)_p, \underbrace{0, 0, ..., 0}_{m-p}).$$

با یک محاسبه ساده، وضوحا نشان داده می‌شود

$$\nabla\Psi(y) = \begin{pmatrix} -(Q + (A^p)^T A^p) & -(A^p)^T & -E^T \\ A^p & \mathcal{W}_{m\times m} & O_{m\times l} \\ E & O_{l\times m} & O_{l\times l} \end{pmatrix},$$

که Oبیانگر یک ماتریس صفر است،

$$A^p = \begin{pmatrix} U_{p\times n} \\ O_{(m-p)\times n} \end{pmatrix} = \begin{pmatrix} A_{1.} \\ A_{2.} \\ \dots \\ \dots \\ A_{p.} \\ O_{1\times n} \\ O_{1\times n} \\ \dots \\ \dots \\ O_{1\times n} \end{pmatrix},$$

و

$$\mathcal{Z}_{m\times m} = \begin{pmatrix} O_{p\times p} & O_{p\times(m-p)} \\ O_{(m-p)\times p} & -I_{(m-p)\times(m-p)} \end{pmatrix}.$$

از [۵۹] می‌بینیم که $(A^p)^T A^p$یک ماتریس نیمه معین مثبت است. ماتریس Q نیز یک ماتریس نیمه معین مثبت فرض شده باشد.بعلاوه، واضح است که ماتریس $\mathcal{Z}_{m\times m}$ نیمه معین منفی است. طبق مطالب بیان شده، می‌توانیم نتیجه بگیریم که ماتریس ژاکوبین $\nabla\Psi(y)$یک ماتریس نیمه معین منفی است.

اگر $p = m$ یعنی،

$$(u + Ax - b)^+ = ((u + Ax - b)_1, (u + Ax - b)_2, ..., (u + Ax - b)_m),$$

آنگاه

$$\nabla\Psi(y) = \begin{pmatrix} -(Q + A^T A) & -A^T & -E^T \\ A & O_{m\times m} & O_{m\times l} \\ E & O_{l\times m} & O_{l\times l} \end{pmatrix}.$$

مشابه حالت قبل، به سادگی ثابت می‌شود که $\nabla\Psi(y)$ یک ماتریس نیمه معین منفی است.
نهایتا، اگر $p = ۰$ ، یعنی $(u + Ax - b)^+ = (\underbrace{۰, ۰, ..., ۰}_{m})$ ، آنگاه داریم :

$$\nabla\Psi(y) = \begin{pmatrix} -Q & O_{n\times m} & -E^T \\ O_{m\times n} & -I_{m\times m} & O_{m\times l} \\ E & O_{l\times m} & O_{l\times l} \end{pmatrix}.$$

در این حالت همچنین به سادگی می‌توان نیمه معین منفی بودن ماتریس $\nabla\Psi(y)$ را تصدیق کرد.
این برهان را کامل می‌کند. □

حال نتایج اصلی را به شرح ذیل بیان می‌کنیم.

قضیه ۴.۴. *فرض کنید که فرض لم ۲.۴ برقرار باشد. آنگاه مدل شبکه عصبی پیشنهادی در (۲۰.۴) و (۲۱.۴) در پایداری لیاپانف است و همگرای سراسری به $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ است، که x^* جواب بهینه (۱۲.۴)-(۱۴.۴) است.*

اثبات. تابع $E : \mathbb{R}^{m+n+l} \to \mathbb{R}$ را به صورت زیر در نظر بگیرید:

$$E(y) = \|\Psi(y)\|^۲ + \frac{۱}{۲}\|y - y^*\|^۲. \qquad (۲۷.۴)$$

از (۱۹.۴)، می‌بینیم که

$$\frac{d\Psi}{dt} = \frac{\partial\Psi}{\partial y}\frac{dy}{dt} = \nabla\Psi(y)\Psi(y).$$

محاسبه مشتق $E(t)$ به همراه جواب $y(t)$از شبکه عصبی (۲۰.۴) و (۲۱.۴) داریم

$$\frac{dE(y(t))}{dt} = (\frac{d\Psi}{dt})^T\Psi + \Psi^T(\frac{d\Psi}{dt}) + (y - y^*)^T\frac{dy(t)}{dt} \qquad (۲۸.۴)$$
$$= \Psi^T(\nabla\Psi(y)^T + \nabla\Psi(y))\Psi + (y - y^*)^T\Psi(y). \qquad (۲۹.۴)$$

با به کار گیری لم ۲.۴ ، بدست می‌آوریم

$$\Psi^T(y)(\nabla\Psi(y)^T + \nabla\Psi(y))\Psi(y) < ۰, \quad \forall y \neq y^*. \qquad (۳۰.۴)$$

بعلاوه، از تعریف ۱۷.۲ و قضیه ۱۰.۲، داریم

$$(y-y^*)^T(\Psi(y)-\Psi(y^*))=(y-y^*)^T\Psi(y)<0,\quad \forall y\neq y^*.$$

بنابراین

$$\frac{dE(y(t))}{dt}\leq 0. \tag{۳۱.۴}$$

این بدین معنی است که شبکه عصبی (۲۰.۴) و (۲۱.۴)پایدار به مفهوم لیاپانف است. چون

$$E(y)\geq \frac{1}{2}\|y-y^*\|^2, \tag{۳۲.۴}$$

یک زیر دنباله همگرا به صورت زیر وجود دارد

$$\{(x(t_k)^T,u(t_k)^T,v(t_k)^T)^T|t_0<t_1<...<t_k<t_{k+1}\},\quad t_k\to\infty\quad k\to\infty,$$

به طوری که $\lim_{k\to\infty}(x(t_k)^T,u(t_k)^T,v(t_k)^T)^T=(\bar{x}^T,\bar{u}^T,\bar{v}^T)^T$ ، که $(\bar{x}^T,\bar{u}^T,\bar{v}^T)^T$در رابطه زیر صدق می‌کند

$$\frac{dE(y(t))}{dt}=0.$$

این نشان می‌دهد که $(\bar{x}^T,\bar{u}^T,\bar{v}^T)^T$یک نقطه ω -حد از $\{(x(t)^T,u(t)^T,v(t)^T)^T|t\geq t_0\}$است. با استفاده از قضیه مجموعه تغییر ناپذیر لازال ، $\{(x(t)^T,u(t)^T,v(t)^T)^T\to M\}$ وقتی که $t\to\infty$، و Mیک مجموعه تغییر ناپذیر بزرگ در $K=\{(x(t)^T,u(t)^T,v(t)^T)^T|\frac{dE(y(t))}{dt}=0\}$ است. از (۲۰.۴) ، (۲۱.۴) و (۳۱.۴) نتیجه می‌شود که ،

$$\frac{du}{dt}=0\,\frac{dx}{dt}=0\qquad\qquad \frac{dv}{dt}=0\Leftrightarrow\frac{dE(y(t))}{dt}=0$$

بنابراین طبق $(\bar{x}^T,\bar{u}^T,\bar{v}^T)^T\in D^*$ و $M\subseteq K\subseteq D^*$. با جایگذاری $x^*=\bar{x}$، $u^*=\bar{u}$و $v^*=\bar{v}$در (۲۷.۴) ، تابع لیاپانف دیگری به صورت زیر تعریف می‌کنیم.

$$\bar{E}(y)=\|\Psi(y)\|^2+\frac{1}{2}\|y-\bar{y}\|^2. \tag{۳۳.۴}$$

آنگاه $\bar{E}(y)$به طور پیوسته مشتق پذیر است و $\bar{E}(\bar{y})=0$. ملاحظه می‌شود که

$$\lim_{k\to\infty}(x(t_k)^T,u(t_k)^T,v(t_k)^T)^T=(\bar{x}^T,\bar{u}^T,\bar{v}^T)^T,$$

بنابراین داریم $\lim_{k\to\infty} \bar{E}(x(t_k)^T, u(t_k)^T, v(t_k)^T)^T = \bar{E}(\bar{x}, \bar{u}, \bar{v})$.

در نتیجه، $\forall \epsilon > 0$ ، $q > 0$ وجود دارد به طوری که برای همه $t \geq t_q$، داریم $\bar{E}(y(t)) < \epsilon$. به طورمشابه $\frac{d\bar{E}(y(t))}{dt} \leq 0$. این نتیجه می‌دهد که برای $t \geq t_q$ ،

$$\frac{1}{2}\|y(t) - \bar{y}\|^2 \leq \bar{E}(y(t)) \leq \epsilon.$$

بنابراین $\lim_{t\to\infty} \|y(t) - \bar{y}\| = 0$ و $\lim_{t\to\infty} y(t) = \bar{y}$. درنتیجه، شبکه عصبی پیشنهادی در (۲۰.۴) و (۲۱.۴) همگرای سراسری به یک نقطه تعادل $\bar{y} = (\bar{x}^T, \bar{u}^T, \bar{v}^T)^T$ است، که $\bar{x}$ جواب بهینه (۱۲.۴)-(۱۴.۴) است. □

یک نتیجه از قضیه ۴.۴ به صورت زیر بیان می‌شود.

نتیجه ۵.۴. *اگر فرض های لم ۲.۴ بر قرار باشد و $D^* = \{(x^{*T}, u^{*T}, v^{*T})^T$ ، آنگاه شبکه عصبی (۲۰.۴) و (۲۱.۴) برای حل (۱۲.۴)-(۱۴.۴) پایدار مجانبی سراسری به نقطه تعادل یکتای $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ است.*

۷.۴ شبیه سازی های عددی

به منظور اثبات کارایی و موثر بودن شبکه عصبی پیشنهادی، در این بخش، ما دو مثال را توسط شبکه عصبی (۲۰.۴)و(۲۱.۴) آزمایش می‌کنیم. شبیه سازی توسط نرم افزار مطلب[۱۶] ۷ اجرا می‌شود. حل کننده معادله دیفرانسیل معمولی ode۴۵ می‌باشد. دقت جواب مثال‌های زیر توسط نرم‌افزار لینگو ۱۱ [۱۷] احراز شده است.

مثال ۱.۴. *داده‌های رگرسیون در جدول ۱.۴ را در نظر می‌گیریم. با استفاده از شبکه عصبی (۲۰.۴)و(۲۱.۴)، توسط یک کرنل RBF برای ماشین بردار پشتیبان برای مساله رگرسیون، تابع گوسین زیر را انتخاب می‌کنیم:*

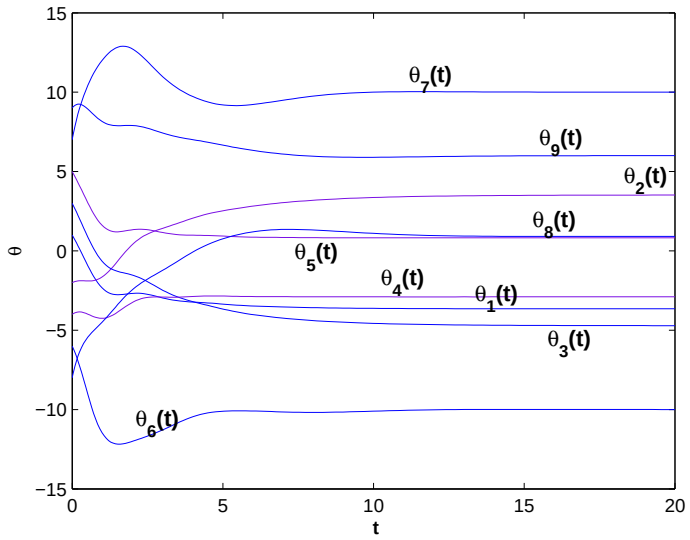
$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\delta^2}\right).$$

[۱۶] Matlab

[۱۷] Lingo

x	۱۲.۷	۱۱.۵	۱۱	۱۰.۲	۳.۸	۱.۶	-۲	-۳	-۵
y	۱۰	۱۰	۹	-۶.۸	۲.۲	-۱.۲	-۱	۱.۸	-۱.۶

جدول ۱.۴: داده رگرسیون



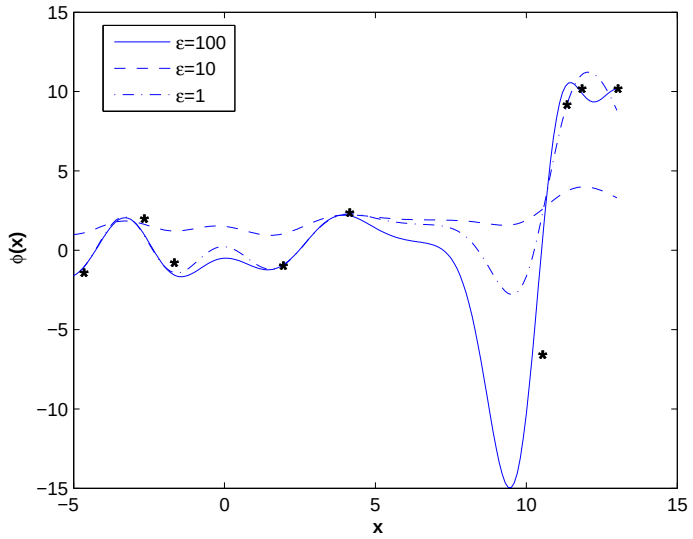
شکل ۱.۴: رفتارهای گذرا از شبکه عصبی (۲۰.۴) و (۲۱.۴) با نقطه شروع $\theta_{\circ} = (۱, -۲, ۳, -۴, ۵, -۶, ۷, -۸, ۹)^T$ در مثال ۱ .

شکل ۱.۴ رفتار همگرایی $\theta(t)$را بر مبنای شبکه عصبی پیشنهادی با نقطه شروع $\theta_{\circ} = (۱, -۲, ۳, -۴, ۵, -۶, ۷, -۸, ۹)^T$ نشان می‌دهد. شکل ۲.۴ نتیجه رگرسیون بردار پشتیبان با $\zeta = ۱۰۰$ ، $\delta = ۱$ و سه پارامتر مختلف ϵ را نشان می‌دهد.

مثال ۲.۴. *در این مثال،یک رگرسیون را بر مبنای داده‌های رگرسیون تیتانیم*[۱۸] [۱] *برای یادگیری* SVR*توسط شبکه عصبی پیشنهادی با تابع گوسین زیر در نظر می‌گیریم.*

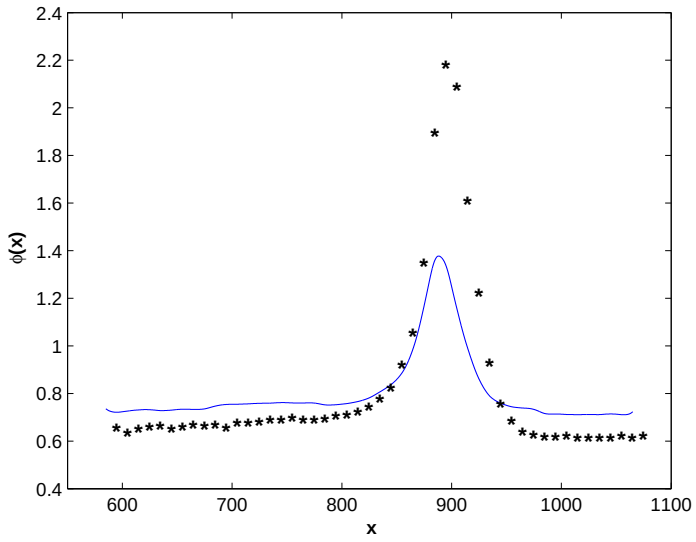
$$K(x, y) = \exp\left(-\frac{\|x - y\|^۲}{۲\delta^۲}\right).$$

[۱۸] Titanium

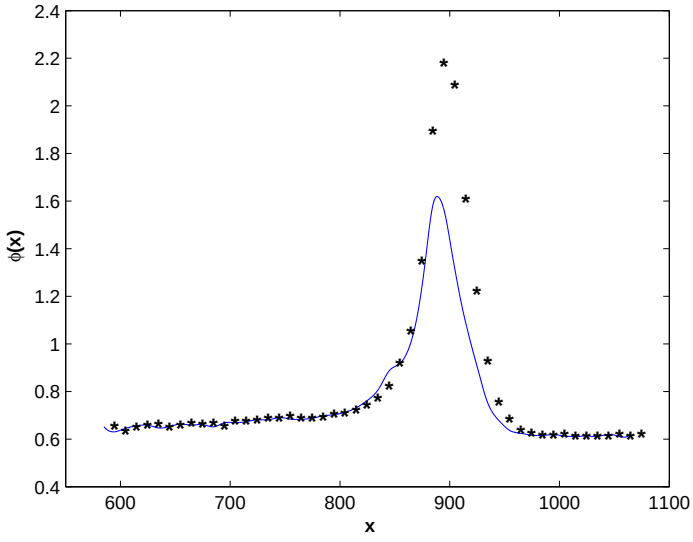


شکل ۲.۴: نتایج رگرسیون بردار پشتیبان با استفاده از شبکه عصبی (۲۰.۴) و (۲۱.۴)با کرنل RBF با $\zeta = ۱۰۰$ ، $\delta = ۱$ و سه پارامتر ϵ مختلف.

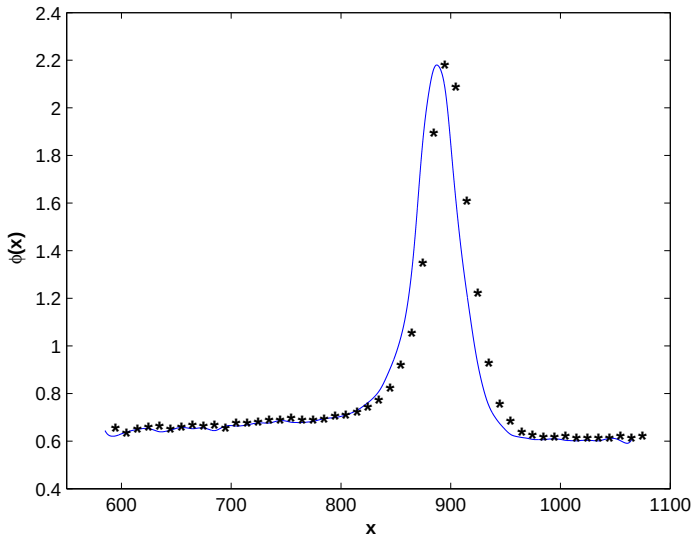
شکل۱.۴ نتیجه SVRرا با $\zeta = ۱۰۰$ ، $\delta = ۶$ و $\epsilon = ۰/۰۱$ نشان می‌دهد. همچنین شبیه سازی عددی با ثابت نگه داشتن ϵ و δ و افزایش ϵ به ازای $\epsilon = ۰/۱$ و $\epsilon = ۱۰۰$ در شکل‌های (۴.۳) و (۵.۳) به ترتیب نشان داده می‌شود. واضح است که به ازای افزایش مقادیر ϵ ، تقریب بهتری از داده‌های تیتانیم توسط تابع رگرسیون بدست می‌آید.



شکل ۳.۴: نتایج رگرسیون بردار پشتیبان با استفاده از شبکه عصبی پیشنهادی (۲۰.۴) و (۲۱.۴) توسط کرنل RBF با $\zeta = ۱۰۰$، $\delta = ۶$ و $\epsilon = ۰/۰۱$.



شکل ۴.۴: نتایج رگرسیون بردار پشتیبان با استفاده از شبکه عصبی پیشنهادی (۲۰.۴) و (۲۱.۴)توسط کرنل RBF با $\zeta = ۱۰۰$، $\delta = ۶$ و $\epsilon = ۰/۱$.



شکل ۵.۴: نتایج رگرسیون بردار پشتیبان با استفاده از شبکه عصبی پیشنهادی (۲۰.۴) و (۲۱.۴)توسط کرنل RBF با $\zeta = ۱۰۰$، $\delta = ۶$ و $\epsilon = ۱۰۰$.

نتایج و پیشنهاداتی برای کارهای آینده

در این پایان‌نامه ابتدا مروری بر شبکه‌های عصبی داشته و با استفاده از یک مدل شبکه عصبی کارا به حل مسأله ی رگرسیون پرداختیم. همچنین پایداری و همگرایی سراسری به جواب بهینه را مورد بررسی قرار دادیم. از جمله مزیت‌های مدل ارائه شده می‌توان به موارد زیر اشاره کرد:

- استفاده از مدل پیشنهادی برای حل مساله طبقه بندی.
- تعمیم مدل پیشنهادی برای داده‌های رگرسیون با ابعاد بالاتر .
- حل مساله رگرسیون ریج [۱۹] با مدل ارایه شده.
- مدل ذکر شده همگرایی تضمین شده دارد.
- این مدل نیاز به هیچگونه پارامتر جریمه ندارد.
- با وجود سادگی مدل، در رده بسیاری از مسائل بهینه‌سازی محدب قابل استفاده است.
- این مدل هیچگونه وابستگی به نقطه شروع نداشته و این نقطه می‌تواند خارج از ناحیه شدنی و نقطه‌ای تصادفی باشد.
- این مدل می‌تواند قادر به حل مسائل وابسته به زمان و مسائل با ابعاد بزرگ باشد.

[۱۹] Ridge regression

مراجع و مآخذ

[1] V. Vapnik , The Nature of Statistical Learning Theory. New York: Springer-Verlag , (1995). 59, 75

[2] Anguita, D., Boni, A., 2002. Improved neural network for SVM learning. IEEE Trans. Neural Networks 13, 1243–1244. 67

[3] C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167. 14, 15

[4] P. Dierckx, Curve and Surface Fitting With Splines. Oxford, U.K.: Clarendon, (1993).

[5] S. Keerthi, D. DeCoste, A modified finite Newton method for fast solution of large scale linear SVMs, Journal of Machine Learning Research 6 (1) (2006) 341–361. 62

[6] O. Mangasarian, A finite Newton method for classification problems, Data Mining Institute, Technical Report, 2001. pp. 01–11.

[7] S. K. Agrawal and B. C. Fabien, Optimization of Dynamic Systems, Kluwer Academic Publishers, Netherlands, (1999). 13, 14

[8] M. Avriel, Nonlinear Programming: Analysis and Methods, Englewood Cliffs, NJ: Prentice-Hall, (1976).

[9] M. S. Bazaraa, H. D. Sherali and C. M. Shetty, Nonlinear Programming- Theory and Algorithms, 2nd ed. New York: Wiley, (1993). 67, 68

[10] D. P. Bertsekas, Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, (1989).

[11] G. Cornuejols, R. Tutuncu, Optimization Methods in Finance, Cambridge University Press, (2006).

[12] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, (2004).

[13] R. Fletcher, Practical Methods of Optimization, New York: Wiley, (1981).

[14] T. Yoshikawa, Foundations of robotics: Analysis and control. Cambridge, MA: MIT Press (1990).

[15] N. Kalouptisidis, Signal processing systems, theory and design. New York: Wiley (1997).

[16] P. T. Harker, J. S. Pang, Finite-dimensional variational inequality and nonlinear complementarity problems: A survey of theory, algorithms, and applications, Math. Progr. ser. B 48 (1990) 161-–220.

[17] B. S. He, L.-Z. Liao, Improvements of some projection methods for monotone nonlinear variational inequalities, Journal of Optimization Theory and Applications 112 (2002) 111–-128.

[18] M. V. Solodov, P. Tseng, "Modified projection-type methods for monotone variational inequalities, SIAM Journal on Control Optimization 34 (1996) 1814-–830.

[19] P. Tseng, A modified forward-backward splitting method for maximal monotone mappings, SIAM Journal on Control Optimization 38 (2000) 431-–446. 22, 24

[20] A. Cichocki, R. Unbehauer, Neural networks for optimization with bounded constraints, IEEE Transactions on Neural Networks 4 (1993) 293–304.

[21] D. W. Tank and J. J. Hopfield, Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming pircuit, IEEE Transactions on Circuits and Systems 33 (1986) 533–541. 16

[22] M. P. Kennedy and L. O. Chua, Neural networks for nonlinear programming, IEEE Transactions on Circuits and Systems 35 (1988) 554–562. 14, 15

[23] A. Cichocki, R. Unbehauer, Neural networks for optimization with bounded constraints, IEEE Transactions on Neural Network 4 (1993) 293–304.

[24] K. Ding and N.-J. Huang, A new class of interval projection neural networks for solving interval quadratic program, Chaos, Solitons and Fractals, 35 (2008) 718-–725.

[25] S. Effati and A. R. Nazemi, Neural network models and its application for solving linear and quadratic programming problems, Applied Mathematics and Computation 172 (2006) 305-–331.

[26] S. Effati, A. Ghomashi and A.R. Nazemi, Application of projection neural network in solving convex programming problems, Applied Mathematics and Computation 188 (2007) 1103–-1114.

[27] M. Forti, P. Nistri, M. Quincampoix, Convergence of neural networks for programming problems via a nonsmooth Lojasiewicz inequality, IEEE Transactions on Neural Networks 17 (2006) 1471- 1486.

[28] T. L. Friesz, D. H. Bernstein, N. J. Mehta, R. L. Tobin, and S. Ganjlizadeh, Day-to-day dynamic network disequilibria and idealized traveler information systems, Operation Research, 42 (1994) 1120-–1136.

[29] X. B. Gao, L.-Z. Liao, L. Q. Qi, A novel neural network for variational inequalities with linear and nonlinear constraints, IEEE Transactions on Neural Networks 16 (2005) 1305–-1317.

[30] X. Hu, Applications of the general projection neural network in solving extended linear-quadratic programming problems with linear constraints, Neurocomputing, 72 (2009) 1131–-1137.

[31] X. Hu and J. Wang, Design of general projection neural networks for solving monotone linear variational inequalities and linear and quadratic optimization problems, IEEE Transactions on Systems, Man, and Cybernetics, Part B, 37 (2007) 1414-–1421.

[32] L. Liao, H. Qi, L. Qi, Solving nonlinear complementarity problems with neural networks: a reformulation method approach, Journal of Computational and Applied Mathematics 131 (2001) 343-359. 6, 7, 8, 9

[33] W. E. Lillo, M. H. Loh, S. Hui and S. H. Zăk, On solving constrained optimization problems with neural networks: A penalty method approach, IEEE Transactions on Neural Networks, 4 (1993) 931-–939.

[34] Q. S. Liu, J. Wang, A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming, IEEE Transactions on Neural Networks 19 (2008) 558–570.

[35] C. Y. Maa and M. A. Shanblatt, Linear and quadratic programming neural network analysis, IEEE Transactions on Neural Networks, 3 (1992) 580–-594.

[36] A. Malek, N. Hosseinipour-Mahani, S. Ezazipour, Efficient recurrent neural network model for the solution of general nonlinear optimization problems, Optimization Methods and Software 25 (2010) 1–18. 7, 9

[37] A. R. Nazemi, A dynamic system model for solving convex nonlinear optimization problems, Communications in Nonlinear Science and Numerical Simulation 17 (2012) 1696–1705.

[38] A. R. Nazemi, A dynamical model for solving degenerate quadratic minimax problems with constraints, Journal of Computational and Applied Mathematics, 236 (2011) 1282–-1295.

[39] A. R. Nazemi, F. Omidi, A capable neural network model for solving the maximum flow problem, Journal of Computational and Applied Mathematics, 236 (2012) 3498–-3513.

[40] A. R. Nazemi, F. Omidi, An efficient dynamic model for solving the shortest path problem, Transportation Research Part C: Emerging Technologies , 26 (2013) 1-–19.

[41] A. R. Nazemi, E. Sharifi, Solving a class of geometric programming problems by an efficient dynamic model, Communications in Nonlinear Science and Numerical Simulation, (In press).

[42] Q. Tao, J. Cao and D. Sun, A simple and high performance neural network for quadratic programming problems, Applied Mathematics and Computation, 124 (2001) 251-–260.

[43] H. Wu, R. Shi, L. Qin, F. Tao, and L. He, A nonlinear projection neural network for solving interval quadratic programming problems and its stability analysis, Mathematical Problems in Engineering (2010) 1-13.

[44] Y. S. Xia, A new neural network for solving linear and quadratic programming problems, IEEE Transactions on Neural Networks 7 (1996) 1544–-1547.

[45] Y. Xia, A new neural network for solving linear and quadratic programming problems, IEEE Transactions on Neural Networks, 7 (1996) 1544–-1547.

[46] Y. Xia, J.Wang, A general projection neural network for solving monotone variational inequality and related optimization problems, IEEE Transactions on Neural Networks, 15 (2004) 318–-328.

[47] Y.S. Xia, J. Wang, A recurrent neural network for solving linear projection equations, Neural Networks 13 (2000) 337–-350.

[48] Y. Xia, J. Wang, A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints, IEEE Transactions on Circuits and Systems 51 (2004) 447–458.

[49] Y. Xia, J. Wang, On the stability of globally projected dynamical systems, Journal of Optimization Theory and Applications, 106 (2000) 129–150.

[50] Y. Xia, G. Feng and J. Wang, A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equation, Neural Networks, 17 (2004) 1003-1015. 62

[51] Y. Xia and G. Feng, An improved network for convex quadratic optimization with application to real-time beamforming, Neurocomputing, 64 (2005) 359-374.

[52] X. Xue and W. Bian, A project neural network for solving degenerate convex quadratic program, Neurocomputing, 70 (2007) 2449–-2459.

[53] X. Xue and W. Bian, A project neural network for solving degenerate quadratic minimax problem with linear constraints, Neurocomputing, 72 (2009) 1826–-1838.

[54] Y. Yang and J. Cao, Solving quadratic programming problems by delayed projection neural network, IEEE Transactions on Neural Networks, 17 (2006) 1630–-1634.

[55] Y. Yang and J. Cao, A delayed neural network method for solving convex optimization problems, International Journal of Neural Systems, 16 (2006) 295–-303.

[56] Y. Yang and J. Cao, A feedback neural network for solving convex constraint optimization problems, Applied Mathematics and Computation, 201 (2008) 340–-350.

[57] T. M. Ortega and W. C. Rheinboldt, Iterative solution of nonlinear equations in Several variables, Academic Press, New York, 1970.

[58] A. Quarteroni, R. Sacco, and F. Saleri. Numerical mathematics, volume 37 of Texts in Applied Mathematics. Springer-Verlag, Berlin, second edition, 2007.

[59] R. K. Miller and A. N. Michel, Ordinary Differential Equations, Academic Press, NewYork, 1982. 70, 71

[60] J. K. Hale, Ordinary Differential Equations. New York: Wiley-Interscience, 1969.

[61] Małafiejski. M, Giaro. K, Janczewski. R, Kubale. M, 2004, Sum coloring of bipartite graphs with bounded degree, Algorithmica 40, 235–244.

[62] On the edge chromatic sum of a graph, Giaro. K, Kubale. M, 1999, ‚János Bolyai Math. Soc, Budapest , 05C75, MR1901884, 58.

[63] S. I. Gass, S. Vinjamuri, Cycling in linear programming problems, Computers & Operations Research 31 (2004) 303–-311. 55, 56

56, 58

[64] Alishahi. M, 2011, On dynamic coloring of graphs, Applied Mathematics and Computation 159 (2-3): 152–156. 55

[65] Y. Xia, J. Wang, A one–layer recurrent neural network for support vector machine learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 34 (2004), 1261-–1269.

[66] Q. Liu, J. Wang, A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming, IEEE Transactions on Neural Networks, (19) 2008, 558–570.

[67] C. Cortes and V. Vapnik , “Support-vector networks,” Mach. Learn., vol. 20, pp. 273–297, 1995. 3

[68] C. J. C. Burges, “A tutorial on support vector machines for patter recognition,” A One-Layer Recurrent Neural Network for Support Vector Machine Learning,

[69] S. R. Gunn, “Support Vector Machines for Classification and Regression,” Univ. Southampton, Southhampton, U.K., Tech. Rep., May 1998. Image Speech and Intelligent Systems Research Group.

[70] J. A. K. Suykens and J. Vandewalle, “Training multilayer perceptron classifiers based on a modified support vector method,” IEEE Trans. Neural Networks, vol. 10, pp. 907–911, July 1999.

[71] E. Osuna, R. Freund, and F. Girosi, “Training support vector machines: An application to face detection,” in Proc. Computer Vision Pattern Recognition, San Juan, Puerto Rico, June 1997.

[72] S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz, “Fusion of face and speech data for person identity verification,” IEEE Trans. Neural Networks, vol. 10, pp. 1065–1074, Sept. 1999.

[73] J. Platt, “Sequential minimal optimization: A fast algorithm for training support vector machines,” in Advances in Kernel Methods—Support Vector Learning, B. Scholkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[74] O. L. Mangasarian and D. R. Musicant, “Successive overrelaxation for support vector machines,” IEEE Trans. Neural Networks, vol. 10, pp. 1032–1037, Nov. 1999.

[75] M. S. Bazaraa, H. D. Sherali and C. M. Shetty, Nonlinear Programming-Theory and Algorithms, 2nd ed. New York: Wiley, 1993.

[76] The conditional coloring numbers of pseudo-Harlin graphs, Meng. X, Miao. L, Gong. Z, Su. B,2006 , Ars Combinatoria, 54. 53

[77] Giaro. K, Kubale. M, 2000, Edge-chromatic sum of trees and bounded cyclicity graphs, Information Processing Letters, 75, 1-2, 65–69. 53

[78] A. R. Nazemi, A dynamic system model for solving convex nonlinear optimization problems, Communications in Nonlinear Science and Numerical Simulation 17 (2012) 1696–1705.

[79] Xin-Yu,Wu.,Yon-shen, Xia.,Jianmin, Li., and Wai-Kai, Chen., a High-Performance Neural Network for Solving Linear and Quadratic Programming Problems, (IEEE Transaction on Neural Network, Vol 7, No 3, May 1996.)

3

abstract

SUPPORT VECTOR MACHINES (SVMs), motivated by statistical learning theory, have recently received considerable attention in the field of machine learning [1]–[17]. Their foundation has been developed by Vapnik and has obtained popularity [1], [2]. The main feature of SVMs is that they use the structural risk minimization rather than the empirical risk minimization. Unlike the well-known multilayer perceptron (MLP) and radial-basis function (RBF) networks, training an SVM is equivalent to solving a linearly constrained convex quadratic programming problem. Since the MLP and RBF networks rely on the minimization of a nonlinear error function which may be nonconvex, the local minima problem in training can be avoided by using the SVM approach. SVMs were first developed to solve the classification problem. Since SVMs have robust properties against noise, SVMs have also been applied to the domain of regression problems [4]. Based on numerical optimization methods, several algorithms and their improvements for support vector classification (SVC) and support vector regression (SVR) have been proposed [9]–[15]. In many engineering and military applications, the demands on real-time data processing is often needed, such as classification in complex electromagnetic envi ronments, recognition in medical diagnostics radar object recognition in strong background clutter, etc. [16]. Therefore, parallel and distributed approaches to training SVMs are necessary and desirable [17]. It is well known that the real-time processing ability of neural networks is one of their most important advantages [18]. In recent years, neural network approaches have demonstrated their great promise for optimization [19]–[23]. Reported results of numerous investigations have shown many advantages over the traditional optimization algorithms, especially in real-time applications [24]–[27]. A two-layers neural network has been applied to SVC [26], [27]. The neural network is suitable for analog hardware implementation and gives a good solution of SVC. This paper further proposes a one-layer neural network for SVC and SVR learning. The advantage of the proposed neural network is twofold. Unlike the existing twolayers neural network, the proposed neural network has a low complexity for implementation. The theoretical analysis shows that the proposed neural network can converge exponentially to the optimal solution of SVM learning. Moreover, the rate of the exponential convergence can be made arbitrarily large by simply

turning up a network parameter. Three illustrative examples shows the superior performance of the proposed neural network for SVM learning.

Shahrood University of Technology

M.S. Thesis

Presents a neural network model for solving the regression problem

By: Abdollmajeed dolati

Supervisor:
Dr. A.R Nazemi

january ,21 , 2014