بسم الله الرحمن الرحیم

تمرین ۴ درس هوش مصنوعی
دکتر فدایی و دکتر یعقوب زاده

مهدی وجهی
۸۱۰۱۰۱۵۵۸

فهرست

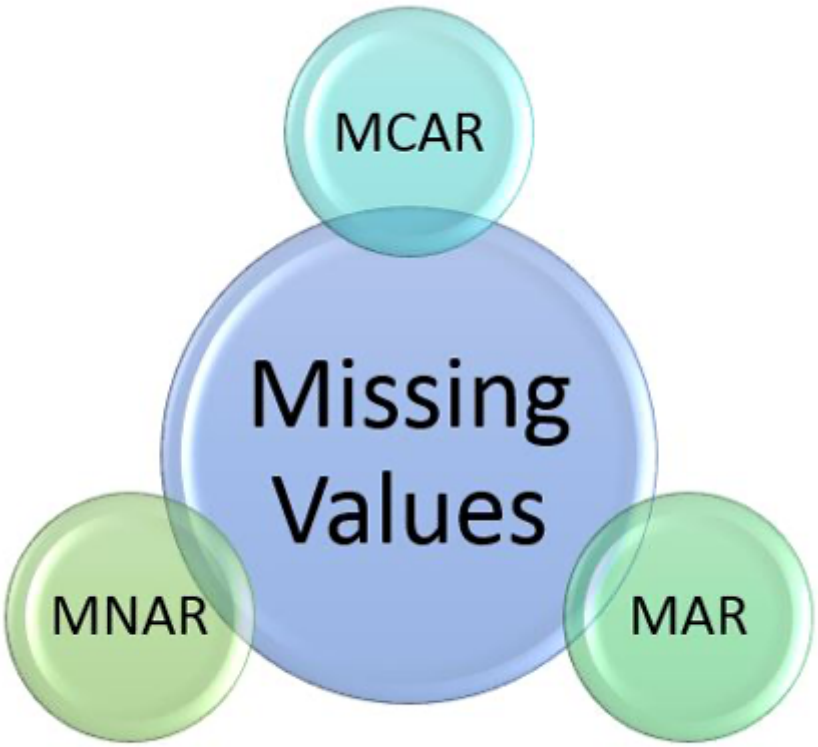
مبحث اول.....3
سوال اول.....3
۱. داده های از دست رفته.....3
۲. نامتعادل بودن کلاس ها.....5
۳. داده های نویز.....6
۴. وجود ویژگی هایی با همبستگی.....8
سوال ۲.....8
مدل پیشنهادی.....8
method square least.....8
gradient descent.....9
سایر روش ها.....13
سوال ۳.....14
Accuracy.....14
Precision.....15
Recall.....15
F1.....15
KNN.....16
فاصله اقلیدسی.....16
فاصله منهتن.....16
SVM.....17
نقاط support vector.....17
موارد نامناسب استفاده از SVM.....19
Kernel.....20
تفاوت soft SVM با hard SVM.....22
نحوه استفاده از SVM در مسائل رگرسیون.....23

مبحث اول

سوال اول

۱. داده های از دست رفته

بهتر است اول با انواع داده های از دست رفته آشنا شویم.

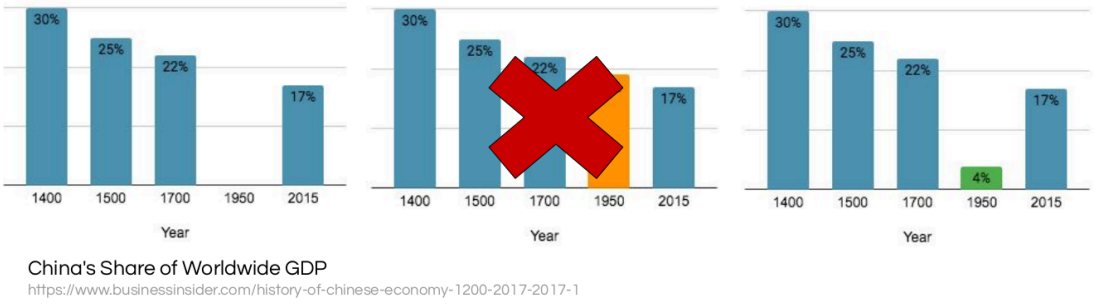


- **Missing Completely At Random – MCAR**: در این حالت احتمال از دست رفتن داده برای همه داده ها یکسان است و ارتباطی با گروه یا موارد موثری بر تحلیل ندارد و الگویی ندارد در واقع این موضوع کاملا تصادفی است. این مشکل به می تواند به دلیل خطای سیستم یا خطای انسانی. مثلا در یک کتابخانه کتابدار فراموش می کند زمان تحویل کتاب را در سامانه ثبت کند.
- **Missing At Random – MAR**: در این حلت ما می توانیم الگویی برای داده های از دست رفته ارائه کنیم. مثلا فرض کنید به ما مجموعه داده ای داده اند که حاوی سن و جنسیت است در این داده افراد به جنسیت خود پاسخ داده اند اما برخی از زنان به سن خود پاسخ نداده اند. در این حالت در صورت تخمین یا حذف داده های می تواند باعث شود نتیجه ی ما اریب شود. ما می توانیم در آخر و بعد تحلیل خود تعداد این موارد و رابطه آنها را توضیح دهیم.
- **Missing Not At Random – MNAR**: در این حالت عدم وجود داده های تصادفی نیست. یعنی گروه خاصی از جامعه به سوال خاصی پاسخ نداده اند یا داده ای ندارم از آنها و ارتباطی هم بین سایر

موارد با آن نمی توانیم پیدا کنیم. مثلا در پرسشنامه افراد فقیر از پاسخ به سوال مربوط به درآمد خودداری می کنند.

می توان از روش های زیر برای حل این مشکل استفاده کرد البته که این موارد بیشتر برای نوع MCAR و سپس MAR جواب می دهد و برای MNAR مناسب نیستند و برای MNAR تنها می باید بعد ارائه مدل خود این موضوع را متذکر شویم:

1. **حذف ردیف ها دارای داده خالی:** راحت ترین کار این است که ما ردیف های حاوی داده خالی را حذف کنیم این کار در بعضی مواقع مناسب است اما مشکلاتی دارد. مثلا می تواند باعث شود که ما حجم بسیاری از داده های خود را از دست بدهیم و این موضوع باعث می شود داده کمی برای تعلیم مدل خود داشته باشیم. مشکل دیگر این است که خالی بودن بعضی ویژگی ها برای بعضی از ردیف های می تواند به دلیل وجود ویژگی خاصی در بین آنها باشد و در واقع آنها نماینده بخش خاصی از جامعه باشد که با حذف داده ما آن ها را از دست می دهیم و این موضوع می تواند باعث کاهش دقت مدل شود. به عنوان مثال فرض کنید ما می خواهیم با مدلی افراد را بر اساس میزان درآمد خود رفتار شناسی کنیم در این داده های ممکن است مثلا افراد فقیر به سوال هایی مانند میزان خوشحالی خانواده خود جواب ندهد و ما با حذف این ردیف های عملا بخش فقیر را از داده های خود حذف کرده ایم.
2. **جایگزینی با میانگین یا میانه:** در این روش که این روش نیز نسبتا راحت است ما داده های خالی را با میانگین یا میانه و یا هر آماره ای که نشان دهنده متوسط جامعه است جایگزین می کنیم و با این کار تاثیر مقدار جایگذاری را در مدل کم می کنیم اما این موضوع می نمی تواند در همه جا مناسب باشد.
3. **تخمین آنها با استفاده از ردیف هایی که در باقی موارد مشابه هستند:** در این روش ما با الگویی برای آن ردیف پیدا می کنیم و داده های از دست رفته آن را با استفاده از آن تقریب می زنیم البته این روش در همه جا مناسب نیست. به عنوان مثال تصویر زیر که از اسلاید های درس علوم داده گرفته شده به خوبی نشان دهد مشکل این روش است:



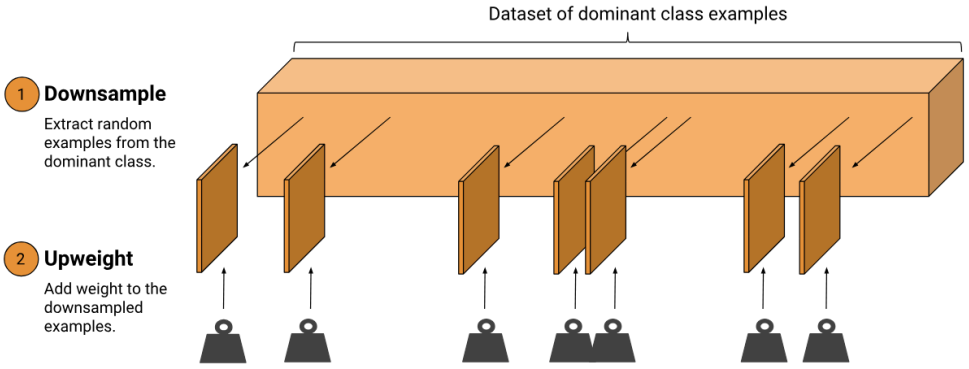
China's Share of Worldwide GDP
https://www.businessinsider.com/history-of-chinese-economy-1200-2017-2017-1

4. **استفاده از داده ها بدون ایجاد تغییر:** در این حالت ما می توانیم از مدل هایی استفاده کنیم که به داده های خالی حساس نباشد مانند درخت تصمیم این روش برای MNAR ها احتمالا مناسب تر هستند.

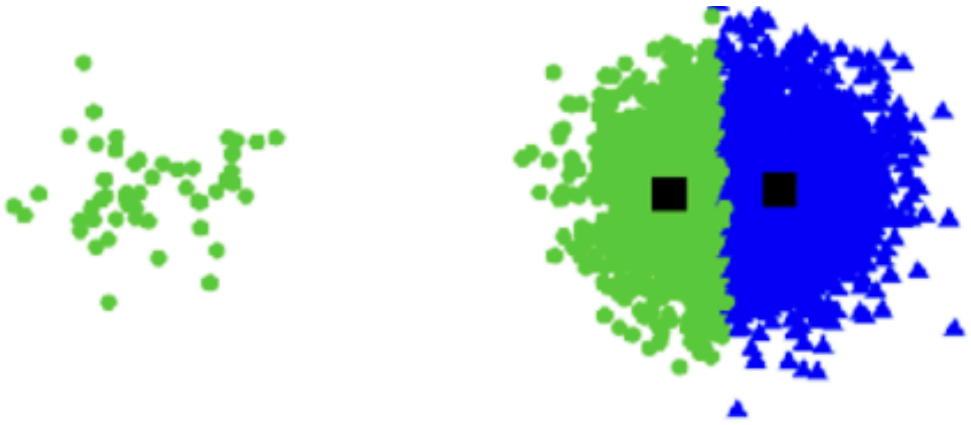
۲. نامتعادل بودن کلاس ها

برای رفع این موضوع می توان از راهکار های زیر استفاده کرد:

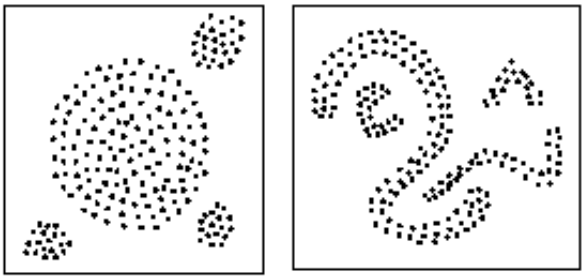
1. **انتخاب معیار سنجش دقت مناسب:** در این نوع داده ها استفاده از accuracy مناسب نیست به عنوان مثال فرض کنید ما باید تشخیص سرطان دهیم در حالت عادی افراد مبتلا به سرطان کمتر از ۱ درصد هستند بنابراین مدل ما اگر همه را غیر سرطانی تشخیص دهد دقتش به ۹۹ درصد می رسد. اما مدل ما عملا هیچ کاری انجام نمی دهد. برای همین بهتر از معیار هایی از جمله rcall یا F1 score استفاده کرد که در پروژه های قبلی آنها را شرح داده ایم.
2. **متعادل سازی داده:** برای این کار راه های مختلفی وجود دارد که به بعضی از آنها اشاره می کنیم:
   a. **تولید داده جدید:** برای این کار می توان مقداری به داده های قبلی نویز اضافه کرد یا آنها را مدل های عصبی بازتولید کرد به طوری که تعداد داده های دسته ها به یکدیگر نزدیک شوند.
   b. **Downsampling و Upweight:** در این روش ما نسبت داده ها را تغییر می دهیم مثلا اگر در مثال سرطان ما ۰.۱ درصد نمونه سرطانی داریم آن را به عنوان مثال به ۵ درصد می رسانیم. برای این کار می توان از دسته کوچک تر نمونه تولید کرد یا از دسته بزرگ تر داده حذف کنیم. در آخر هم وزن داده های دسته بزرگ تر را زیاد می کنیم.



3. **استفاده از مدل هایی که در مقابل این عدم توازن مقاوم هستند:** به عنوان مثال فرض کنید که ما باید داده ها را خوشه بندی کنیم اگر خوشه های نامتعادل باشند احتمالا روش k-mean مناسب نیست به تصویر زیر توجه کنید:



اما اگر ما در این حالت از مدل هایی مثل DBSCAN استفاده کنیم عملکرد بهتری می گیریم زیرا این روش به اندازه هر خوشه وابسته نیست. به عنوان مثال در تشخیص موارد زیر به خوبی عمل کند.



۳. داده های نویز

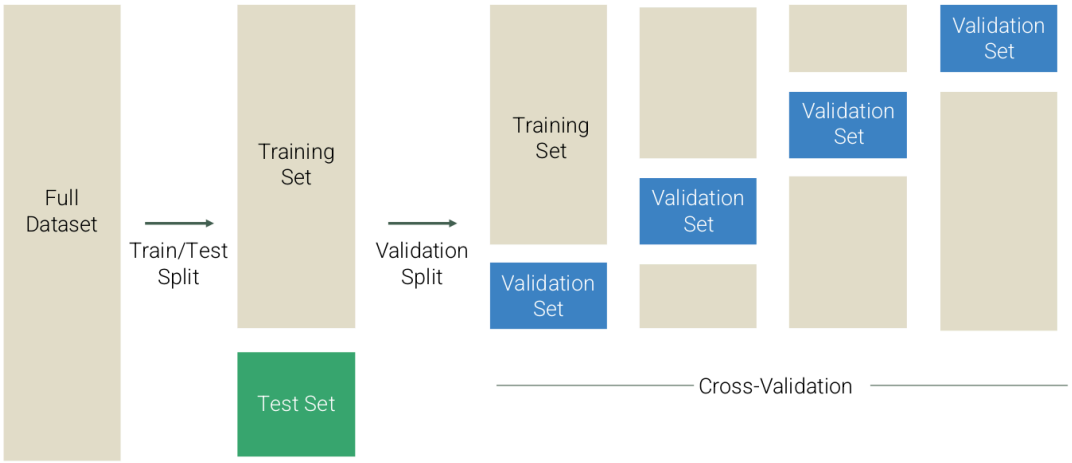
نویز های انواع مختلفی دارند بهتر از ابتدا با انواع آنها آشنا شویم:

1. **Feature Noise**: به ویژگی هایی گفته می شود که ارتباطی ما هدف ما ندارد و باعث افزایش پیچیدگی مدل، فیت شدن روی داده های آموزشی و کاهش میزان تعمیم پذیری مدل شود.
2. **Systematic Noise**: نویز های سیستماتیک شامل مواردی هستند که مثلا یک داده چند بار جمع آوری شده یا مثلا داده های به صورت اریب برچسب خورده باشند مثلا ممکن است که اگر چند نفر داده ها را برچسب بزنند بسته به نظر افراد برچسب های متفاوتی به بعضی موارد یکسان بخورد یا مثلا داده ها تاریخ در ابتدا به صورت میلادی ذخیره می شدند و این موضوع بعدا تغییر کرده ولی داده های قبلی میلادی باقی مانده اند.
3. **Random Noise**: این نویز های به صورت تصادفی هستند و به علت عواملی همچون خطای ابزار اندازه گیری ایجاد می شوند.
4. **Background noise**: به ردیف هایی از داده گفته می شوند که وجود داده های آن ضرورتی ندارد و باعث انحراف مدل می شود مثلا ما می خواهیم یک مدل برای پیشبینی مواردی در سال ۲۰۲۴ داشته باشیم برای این موضوع داده های ۲۰۲۳ و ۲۰۲۲ را جمع آوری می کنیم. در این حالت اگر ما داده های سال ۱۹۹۵ را اگر اضافه کنیم این داده های می تواند باعث گمراهی مدل ها شود بنابراین بهتر است آنها را قرار ندهیم یا حذف کنیم.

برای این موضوع راهکار های زیر پیشنهاد می شود:

1. **بررسی دقیق دلیل وجود نویز**: در بعضی مواد ممکن است داده های نویز نباشد و داده ی پرت باشند در این موارد باید بررسی کرد دلیل این موضوع را پیدا کرد و سپس برای آنها اقدام کرد.
2. **حذف داده های نویز**: راحت ترین کار حذف داده های نویز است در صورتی که این موارد زیاد نیستند می توان آنها برای حذف کرد که باعث انحراف مدل نشوند اما اگر داده های نویز زیاد هستند می تواند حذف آنها منجر به این شوند که مدل ما نتواند با داده هایی به اندازه مناسب آموزش ببیند.

3. **بازتولید داده ها(Autoencoders)**: می توان با استفاده از مدل های شبکه عصبی داده ها را بازتولید کرد و نویز آنها را کاهش داد این موضوع به خصوص در پردازش تصویر و آموزش مدل های تصویری می تواند کارآمد باشد.
4. **استفاده از مدل ها و آماره هایی که حساسیت کمتری روی داده های پرت دارند**: برای گزارش این نوع داده ها باید توجه داشته باشیم که احتمالا استفاده از مواردی همچون میانه که حساسیت کمتری به داده های پرت نسبت میانگین دارند مناسب تر باشد. همچنین استفاده از مدل هایی که حساسیت کمتری روی داده های نویز دارند می تواند مفید باشد به عنوان مثال می توان تابع خطا را به جای مربع خطا به صورت قدر مطلق خطا حساب کرد.
5. **تبدیل داده نویز به مقادیر قابل قبول**: در بعضی موارد تنها قالب داده متفاوت است و ما می توانیم به آسانی آنها را به قالب درست تبدیل کنیم به عنوان مثال در بخشی از داده های ما تاریخ به صورت میلادی و در باقی موارد شمسی است به آسانی می توان تاریخ های میلادی را به شمسی تبدیل کرد و از آنها استفاده کرد.
6. **نگهداشتن و افزایش داده های نویز!**: داده های نویز همیشه بد نیستند و این موضوع باعث می شود مدل روی داده های آموزشی فیت نشوند و فقط الگوی داده ها را یاد بگیرد در بعضی موارد که داده های ما کم هستند با استفاده از روش هایی از جمله ایجاد نویز داده های جدیدی بازتولید می کنیم که مدل بتواند با داده های بیشتر آموزش ببینید.
7. **استفاده از تبدیل فوریه**: در صورتی که داده های ما در قالب تصویر یا صوت هستند می تواند برای حذف نویز آنها از تبدیل فوریه استفاده کرد.
8. **استفاده از ابعاد اصلی (PCA)**: با استفاده از این تکنیک می توان ویژگی ها را کاهش داد و ویژگی های نویز را حذف کرد. این روش واریانس داده ها را در هر بعد می سنجد و بر طبق آن ابعاد اصلی را انتخاب می کند.
9. **Cross Validation**: در این روش ما برای بررسی مدل خود داده های آموزشی را به قسمت هایی تقسیم می کنیم یکی را برای ارزیابی استفاده می کنیم و با مابقی آموزش می دهیم. این کار را به صورت چرخشی تکرار می کنیم.



۴. وجود ویژگی هایی با همبستگی

می توان از راه های زیر استفاده کرد:

1. **نگهداشتن فقط یکی از آنها:** راحت ترین راه این است که ما تنها از یکی از آنها استفاده کنیم
2. **استفاده از ابعاد اصلی (PCA):** در این روش ما ویژگی هایی که همبستگی به یکدیگر دارند را کاهش می دهیم.
3. **داده ها را تغییر ندهیم:** در بعضی موارد اهمیت آن دسته از ویژگی ها برای ما زیاد است و می خواهیم مدل به آنها توجه بیشتری بکند در این حالت ما تغییری در آنها ایجاد نمی کنیم.
4. **استفاده از مدل هایی که استقلال ویژگی ها شرط آنها نیست:** برخی مدل ها به شرط استقلال ویژگی ها توسعه داده شده اند. به عنوان مثال مدل naive bayes شرط استقلال دارد پس استفاده از آن در این نوع داده ها ریسک بالا و احتمالا خطای بالایی داشته باشد.
5. **استفاده از Bagging و Random Forests:** در این روش ما مدل هایی با ویژگی های متفاوت آموزش می دهیم در آخر بین آنها رای گیری می کنیم. در این روش ما ویژگی های داری همبستگی را در مدل های مختلف قرار می دهیم.

سوال ۲

مدل پیشنهادی

با توجه به این که باید یک مقدار عددی پیوسته را پیشبینی کنیم باید از hyperplane و regression استفاده کنیم.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p$$

در اینجا ما تا تتای ۲ داریم و یکی از ایکس ها ساعت مطالعه و دیگری تعداد آزمون است.

method square least

در این روش تابع بهینه سازی ما به صورت زیر است:

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
$$= \frac{1}{n}||\mathbb{Y} - \hat{\mathbb{Y}}||_2^2$$

سپس تخمین خود را به صورت ماتریسی باز می کنیم:

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ 1 & x_{31} & x_{32} & \dots & x_{3p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_p \end{bmatrix}$$

$$\hat{\mathbb{Y}} = \mathbb{X}\theta$$

که در واقع در هر ستون x شامل مقادیر یک فیچر خاص برای هر داده است. پس در نتیجه ماتریس ما در این مسئله ۳ تا ستون دارد زیرا ۲ تا ویژگی تعداد آزمون و ساعت مطالعه را داریم.

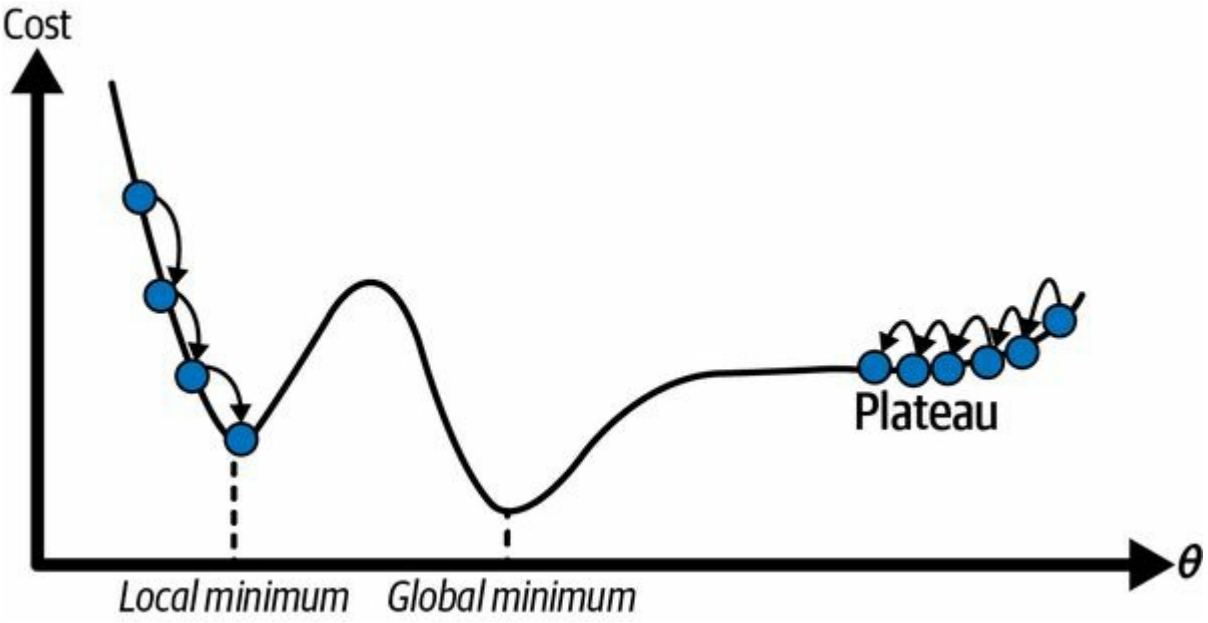
در نتیجه تابع به صورت زیر می شود:

$$R(\theta) = \frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2$$

حال که تابع را تعریف کردیم باید برای به دست آوردن مقدار بهینه هر پارامتر طبق دانشی که از بهینه سازی داریم از تابع بر حسب پارامتر مربوطه مشتق جزئی بگیریم و مقدار ۰ آن یعنی اکسترمم آن را محاسبه کنیم. با این کار ما بهینه ترین پارامتر ها را پیدا کردیم.

gradient descent

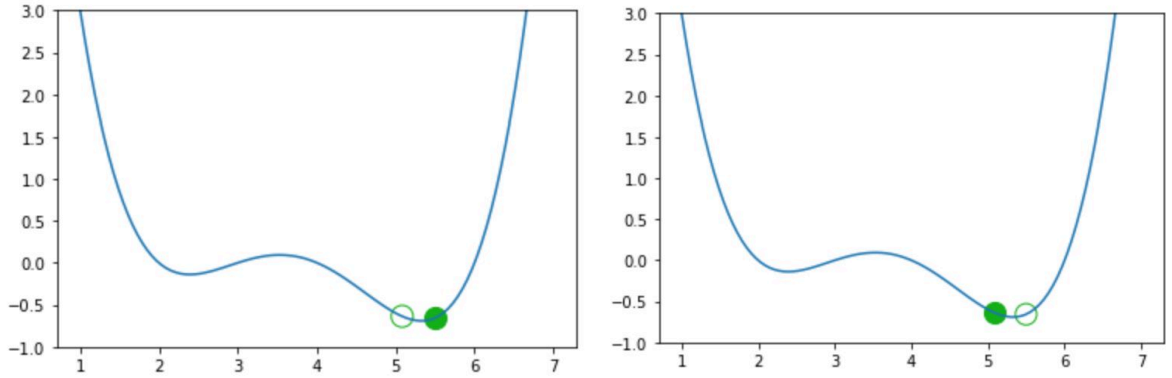
این روش در واقع نوعی جست و جو در فضای پیوسته است ما در تابع خود پیمایش می کنم و نقطه اکسترمم مطلوب خود را پیدا می کنیم. این روش لزوما بهترین پاسخ را به ما نمی دهد و ممکن است در اکسترمم های محلی گیر بیفتند با اجرای چند برای الگوریتم میتوان این موضوع را بهبود داد. تصویر زیر گیر کردن الگوریتم در اکسترمم های محلی را نشان می دهد.



برای این کار ابتدا از نقطه ی شانسی ای شروع می کنیم. فرض می کنیم نقطه مطلوب ما مینیمم تابع است در فضای تک پارامتری مقداری جلوتر و عقب تر از نقطه فعلی را در تابع بررسی می کنیم طبیعی است که باید به سمتی برویم که بیشتر است نکته ای که باید در نظر داشته باشیم این است که چه مقدار حرکت کنیم. برای این که میزان حرکت را مشخص کنیم به صورت عددی مشتق تابع را حساب می کنیم و به همان مقدار حرکت می کنیم. پس تابع به شکل زیر می شود:

$$x^{(t+1)} = x^{(t)} - \frac{d}{dx} f\left(x^{(t)}\right)$$

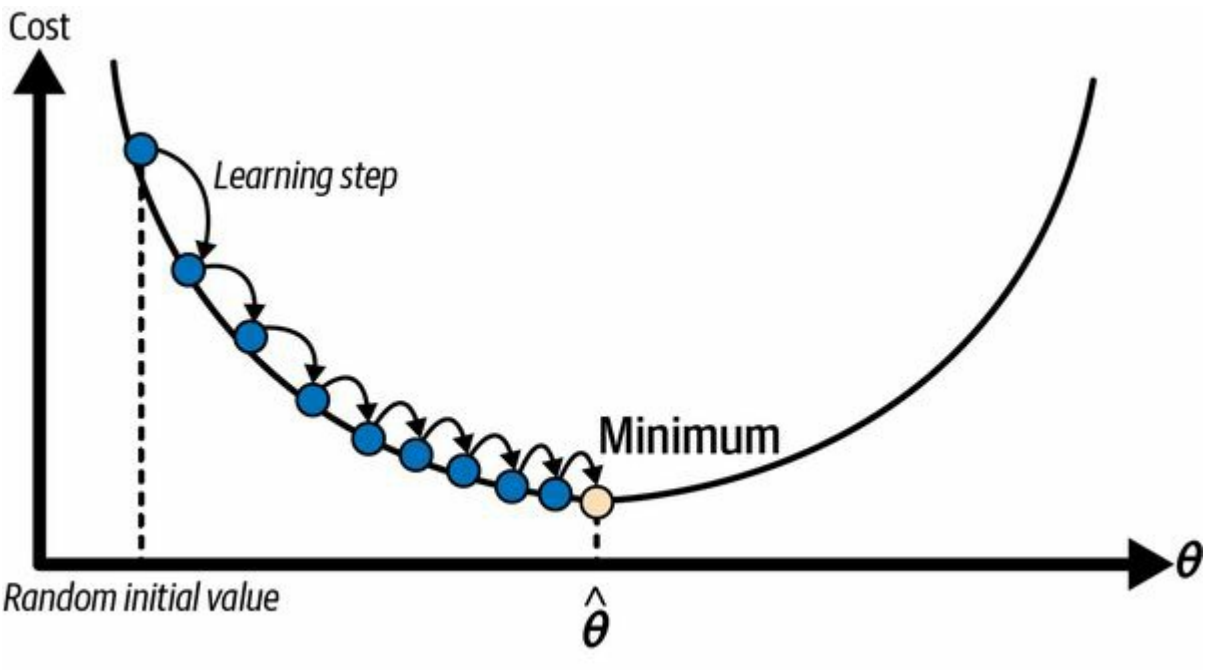
حال مشکلی وجود دارد این است که وقتی به اکسترمم نزدیک می شویم حول آن نوسان می کنیم به دو تصویر زیر نگاه کنید ما به طور پی در پی بین این دو نقطه جابه جا می شویم.



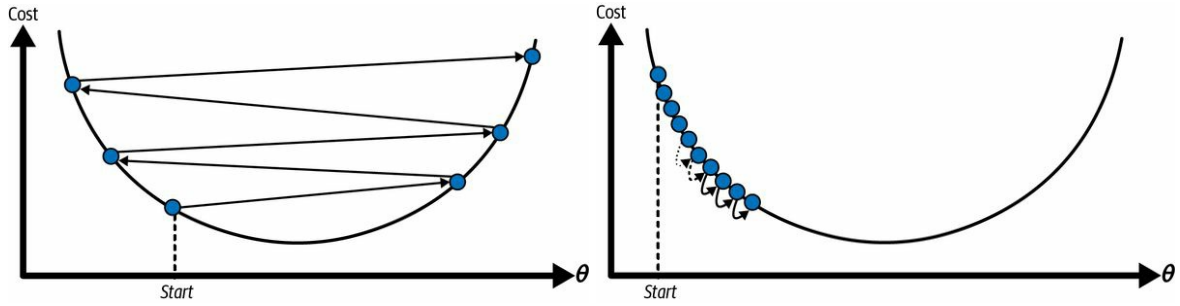
برای حل این مشکل ما یک ضریب یادگیری قرار می دهیم تا این مشکل را برطرف کنیم.

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \frac{d}{d\theta} L\left(\theta^{(t)}\right)$$

این ضریب در مرور زمان کاهش می یابد و باعث همگرایی الگوریتم می شود همچنین در تنظیم تابع به ما انعطاف بیشتری می دهد. به تصویر زیر توجه کنید.



باید توجه کنیم که اگر این ضریب مقدار بسیار کم یا زیادی بگیرد مشکل ساز می شود به تصویر زیر این موضوع را نشان می دهد. در تصویر راست این ضریب کم است و در تصویر چپ زیاد.

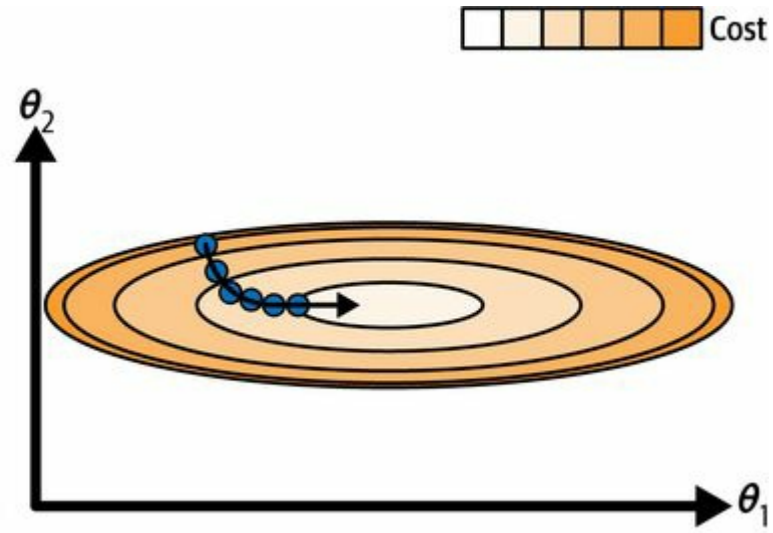


مورد دیگری که باید در نظر بگیریم این است که اگر مقادیر را نرمالایز کنیم و اسکیل آنها را تنظیم کنیم مدل سریع تر عمل می کند.

حال الگوریتم را برای ابعاد بالا تر تعمیم می دهیم. کافیست که عبارت را به صورت ماتریسی بازنویسی کنیم و بر حسب پارامتر ها مشتق جزئی بگیریم.

$$\begin{bmatrix} \theta_0^{(t+1)} \\ \theta_1^{(t+1)} \\ \vdots \end{bmatrix} = \begin{bmatrix} \theta_0^{(t)} \\ \theta_1^{(t)} \\ \vdots \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial L}{\partial \theta_0} \\ \frac{\partial L}{\partial \theta_1} \\ \vdots \end{bmatrix}$$

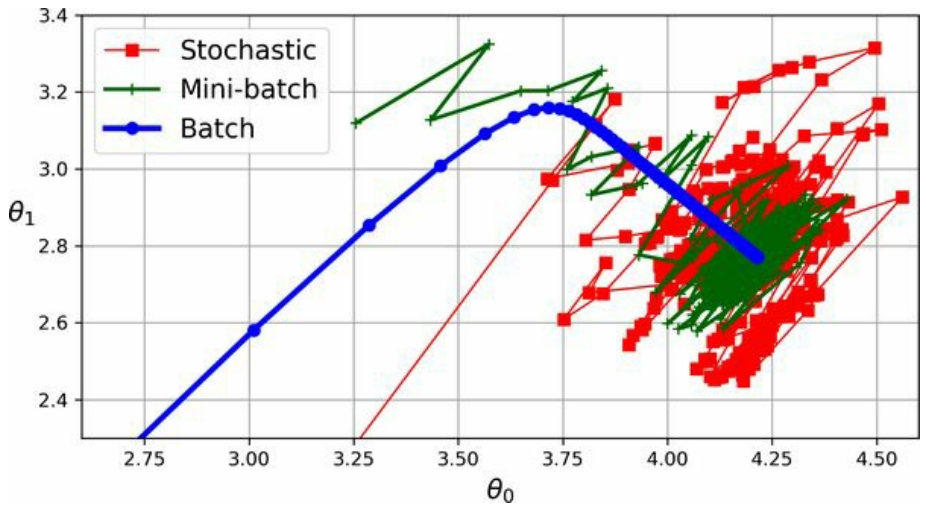
تصویر زیر عملکرد این روش را در تابعی با ۲ پارامتر نشان می دهد.



نکته ای که وجود دار این است که ما برای هر مرحله تابع خود را چطور تشکیل دهیم روش های مختلفی وجود دارد.

- **Batch**: در این روش ما تابع خود را با تمام داده های خود تشکیل می دهیم.
- **Stochastic**: در این روش ما تابع خود را تنها با یک داده تشکیل می دهیم.
- **Mini-batch**: این روش مابین دو روش قبلی است و ما با انتخاب مجموعه ای از کل داده ها تابع خود را تشکیل می دهیم.

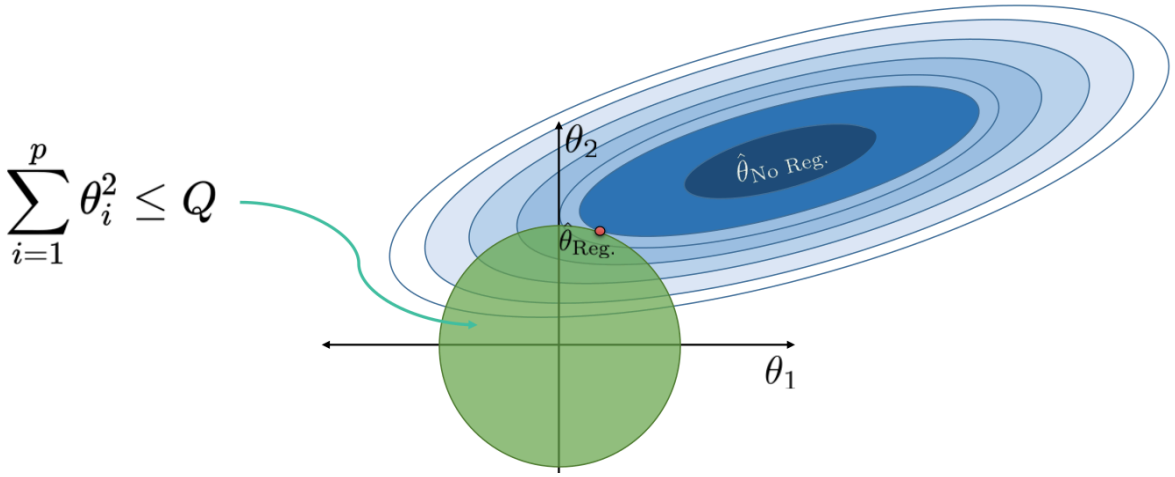
دو تصویر زیر مقایسه این روش ها را نشان می دهد. m نشان دهنده تعداد ردیف های داده است و n تعداد فیچر ها را نشان می دهد.



Algorithm	Large *m*	Out-of-core support	Large *n*	Hyperparams
Normal equation	Fast	No	Slow	0
SVD	Fast	No	Slow	0
Batch GD	Slow	No	Fast	2
Stochastic GD	Fast	Yes	Fast	≥2
Mini-batch GD	Fast	Yes	Fast	≥2

سایر روش ها

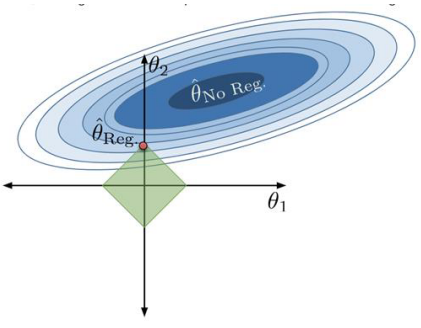
Ridge regression: در این روش ما فضای جست و جوی خود را محدود نگه می داریم که جست و جو سریع تر انجام شود. در تصویر زیر دایره نشان دهنده محدود جست و جوی است.



که تابع آن به صورت زیر نوشته می شود.

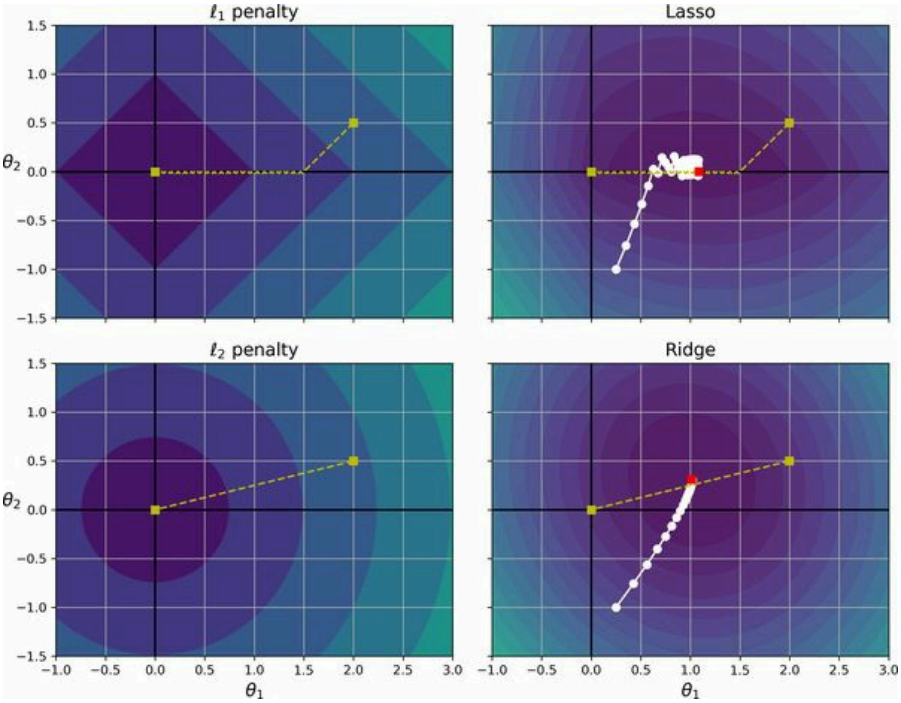
$$\underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - (\theta_0 + \theta_1\phi_{i,1} + \ldots + \theta_p\phi_{i,p})\right)^2}_{\text{Keep MSE on the data low...}} + \underbrace{\lambda\sum_{i=1}^{p}\theta_i^2}_{\text{...while also keeping the size of parameters small}}$$

Lasso Regression: این روش مشابه روش قبلی است با این تفاوت که به جای توان ۲ از قدر مطلق در تابع جریمه استفاده می شود این موضوع باعث میشود که نمودار آن حالت لوزی شکل پیدا کند و این موضوع سبب می شود که مدل سعی کند تعداد فیچر های مورد استفاده خود را کاهش دهد.



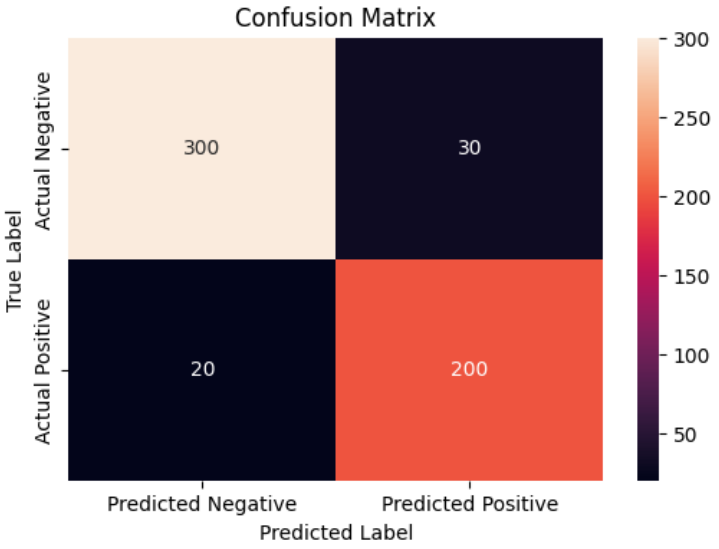
$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - (\theta_0 + \theta_1\phi_{i,1} + \ldots + \theta_p\phi_{i,p})\right)^2 + \lambda\sum_{i=1}^{p}|\theta_i|$$

در تصویر زیر هم می تواند مقایسه عملکرد دو مدل مذکور را مشاهده کنید.



سوال ۳

نتایج ارزیابی به صورت زیر است:



Metric	Value
Recall	0.937500
Precision	0.909091
Accuracy	0.909091
F1 Score	0.923077

موارد مذکور به صورت زیر حساب می شوند.

Accuracy

ای معیار نسبت موارد درست تشخیص داده شده به کل را نشان می دهد.

Precision

نسبت موارد درستی با تشخیص درست به کل مواردی که درست تشخیص دادیم(لزوما درست تشخیص داده نشده اند). (T به معنای درست تشخیص داده شده و F به معنی غلط تشخیص داده شده و P به معنی این است که ورودی واقعا مثبت است و N یعنی ورودی واقعا منفی است.)

$$\frac{TP}{TP+FP}$$

Recall

نسبت مواردی که درست تشخیص داده ایم نسبت به مواردی که درست هستند.(چه تشخیص داده باشیم چه نداده باشیم)

$$\frac{TP}{TP+FN}$$

F1

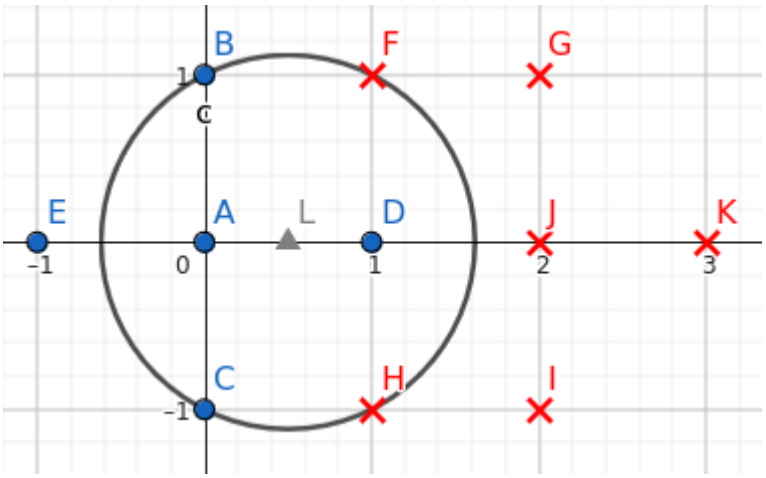
میانگین گیری هارمونیک از ۲ معیار بالا

$$2 \times \frac{Precision \times Recall}{Precision+Recall}$$

KNN

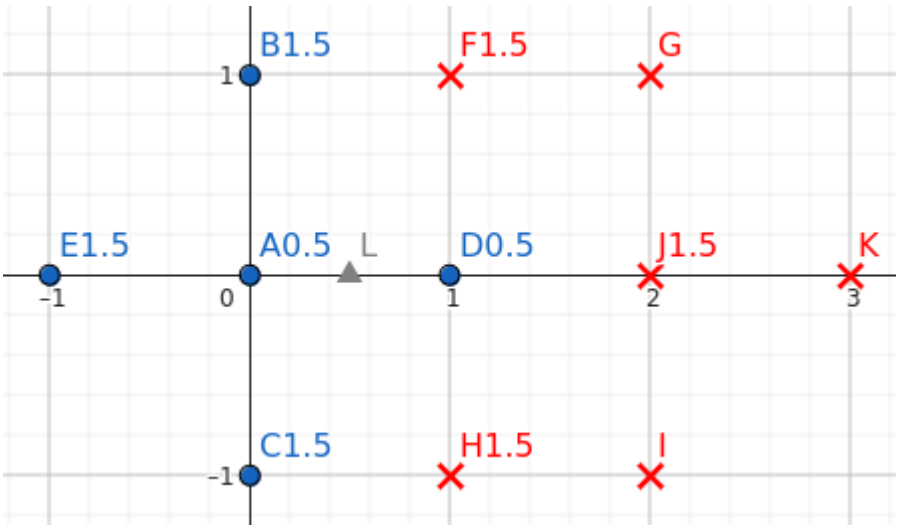
فاصله اقلیدسی

همانطور که مشاهده می شود به کلاس دایره نسبت داده می شود.



فاصله منهتن

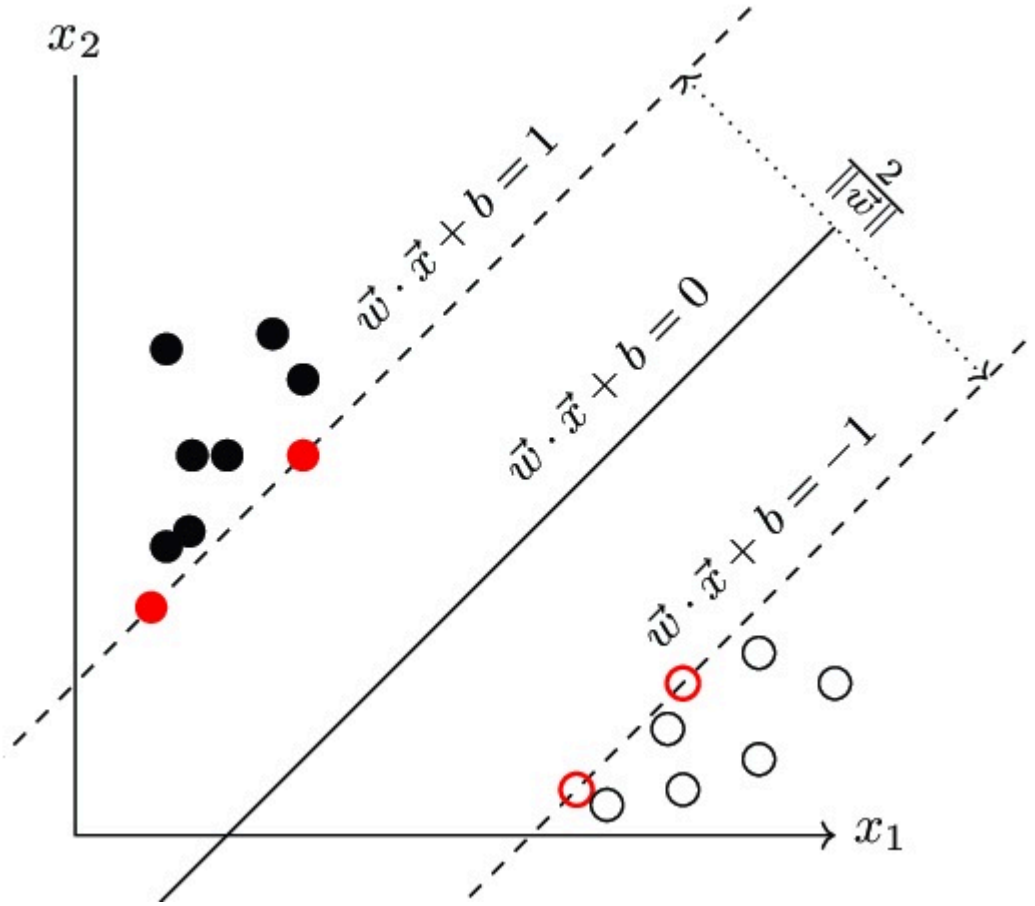
فاصله منهتن محاسبه شد و مجدد دایره انتخاب شد.



SVM

نقاط support vector

(برگرفته از اسلاید های درس یادگیری ماشین دکتر توسلی پور)
در SVM ما باید بهترین خط را پیدا کنیم.



خطی بهترین است که بیشتر فاصله را از ۲ دسته داشته باشد یعنی خط $\mathbf{w}.\mathbf{x}+b=0$. واضح است و اثبات می شود که برای این که این خط بهترین خط باشد باید $\frac{2}{||\vec{\omega}||}$ ماکزیمم شود. پس طبق شکل و موارد گفته شده مسئله بهینه سازی خود را به شکل زیر تعریف می کنیم.

$$\min_{\omega,\, b} \; ||\vec{\omega}||_2^2$$

$$s.t.\; y_i(x_i^T\omega + b) \geq 1$$

حال برای حل مسئله بهینه سازی خود تابع lagrangian را می نویسیم. این تابع برای حل مسائل بهینه سازی با شرایط محدود کاربرد دارد و فرم آن به صورت زیر است.

minimize $f(\mathbf{x})$
subject to
$g_i(\mathbf{x}) \leq 0,$
$h_j(\mathbf{x}) = 0.$

$$\mathcal{L}(\mathbf{x}, \mu, \lambda) = f(\mathbf{x}) + \mu^\top \mathbf{g}(\mathbf{x}) + \lambda^\top \mathbf{h}(\mathbf{x})$$

تابع g را به صورت زیر تعریف می کنیم.

$$g(\mu,\ \lambda) := \min_{x}\ L(x, \mu, \lambda)$$

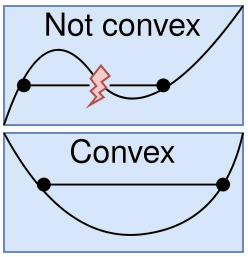
و مسئله بهینه سازی خود را به صورت زیر تعریف می کنیم.

$$\min_{\mu,\lambda}\ g(\mu,\ \lambda)$$

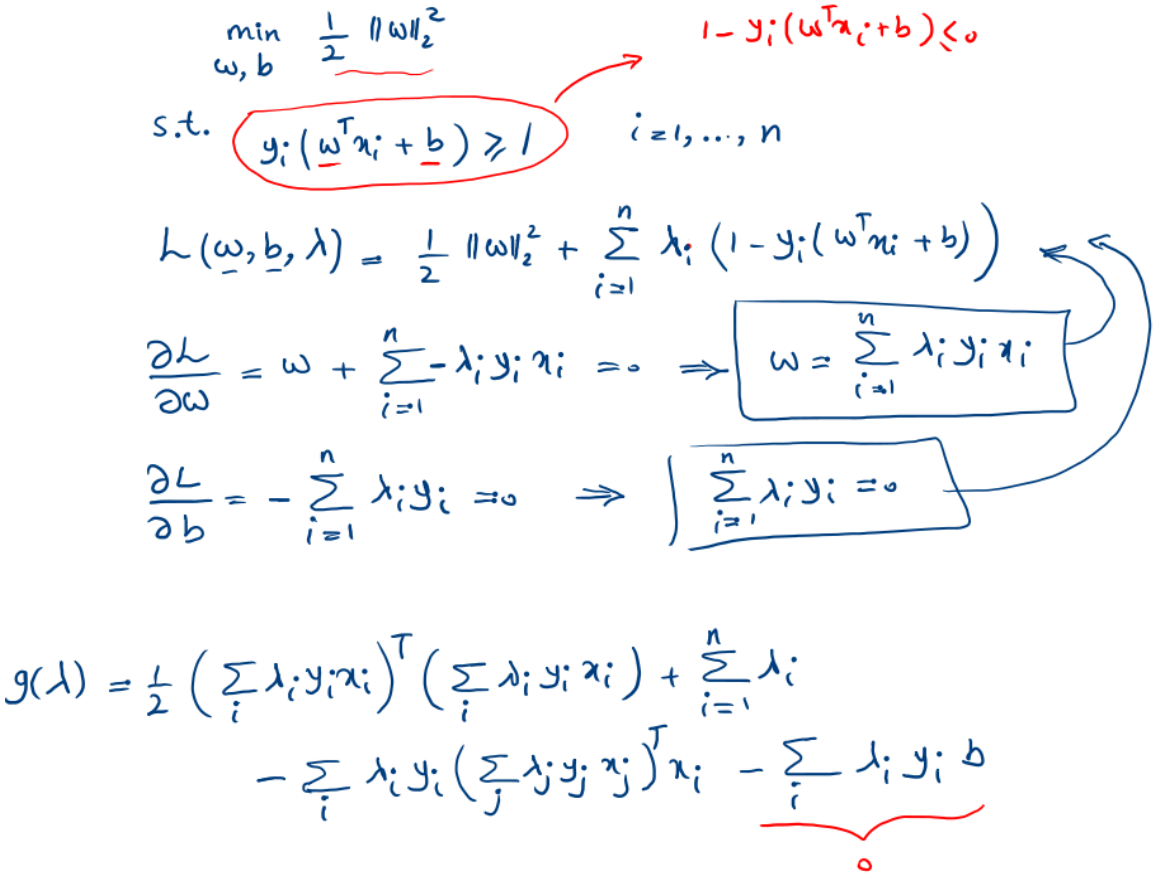
$$s.t.\ \lambda_i \geq 0$$

طبق قضیه strong duality اگر شرایط زیر برقرار باشد این مقدار بهینه برابر با مقدار بهینه تابع اصلی ما می شود.

- Convex
- $\exists x,\ g_i(x) < 0$



حال که به این موضوع پی بردیم شروع به حل مسئله می کنیم. ابتدا از تابع خود مشتق جزئی می گیریم و مقادیر را به دست می آوریم سپس آنها را جایگذاری می کنیم.



$$g(\lambda) = \frac{1}{2}\left(\sum_i \lambda_i y_i x_i\right)^T \left(\sum_i \lambda_i y_i x_i\right) + \sum_{i=1}^{n} \lambda_i$$
$$- \sum_i \lambda_i y_i \left(\sum_j \lambda_j y_j x_j\right)^T x_i - \sum_i \lambda_i y_i b$$
$$0$$

$$= \frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j x_i^T x_j - \sum_i\sum_j \lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_i \lambda_i$$

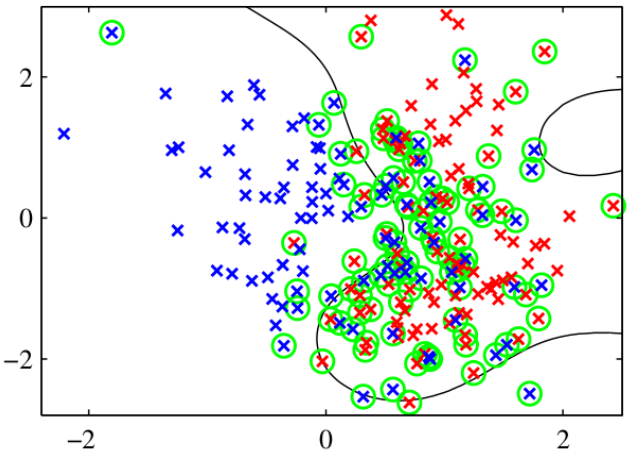
$$\Rightarrow g(\lambda) = -\frac{1}{2}\sum_i\sum_j \lambda_i\lambda_j y_i y_j x_i^T x_j + \sum_i \lambda_i \qquad \lambda_i \geq 0$$

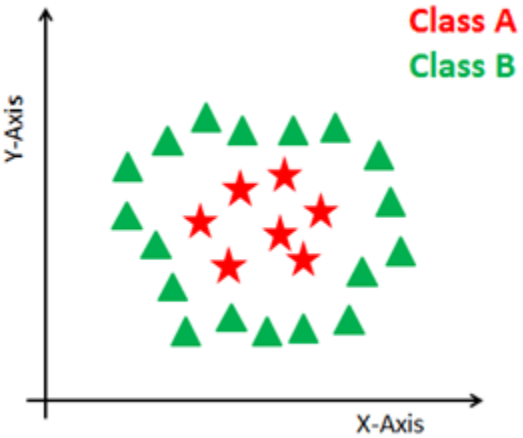
در نهایت عبارت به دست آمده به آسانی با روش های عددی قابل حل است آن را حل می کنیم. و به یک بردار از لامبدا ها می رسیم. **در این بردار به نقاط متناظر لامبدا هایی که صفر نیستند، support vector گفته می شود. این نقاط در واقع همان نقاطی هستند که در تصویر روی خط چین ها قرار گرفته اند.** این نقاط، نقاط موثر در مسئله بهینه سازی هستند و باقی نقاط که از خط دورتر هستند تاثیری در محاسبه SVM ندارند و چون لامبدا متناظر آنها صفر است اصلا در محاسبه وارد نمی شوند.

$$\lambda^* = \begin{bmatrix} \lambda_1^* \\ \vdots \\ \lambda_n^* \end{bmatrix}$$

موارد نامناسب استفاده از SVM

در حالت اولیه SVM تنها برای داده هایی مناسب است که بتوان به صورت خطی آنها را جدا کرد. اما مثلا حالت هایی مانند تصاویر زیر را در نظر بگیرید.

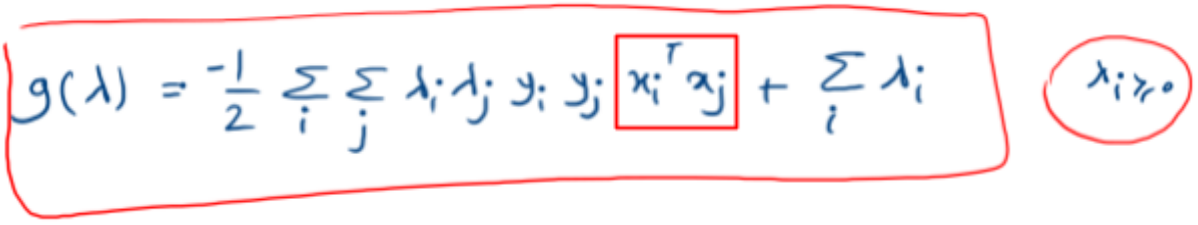




در این حالت مدل نمی تواند داده های را دسته بندی کند اما می تواند با استفاده از یک تابع تبدیل فیچر ها را به گونه ای تغییر داد که خطی شوند. مثلا در داده های بالا راست اگر ما داده ها را به مختصات قطبی ببریم خطی می شوند.در حالت کلی اگر با یک تبدیل برداری ویژگی های خود را به یک فضای خطی ببریم می توانیم از مدل استفاده کنیم.

Kernel

در بخش قبل گفته شد که ما اگر بتوانیم یک فضای خطی جدید تعریف کنیم و ویژگی های خود را با یک تبدیل خطی به آن فضا ببریم می توانیم از مدل برای داده هایی که در حالت اولیه غیر خطی هستند استفاده کنیم. اما مشکلی که وجود دارد این است که ما حجم بسیاری ویژگی و بعد داریم و تعریف تبدیلی که فضای خطی ما را به یک فضای خطی مناسب برای مدل تغییر دهد، سخت و در بسیاری موارد ممکن نیست. در بعضی موارد هم ابعاد در فضای مقصد بسیار زیاد یا بینهایت هستند مثلا فضایی مثل فوریه را در نظر بگیرید که باعث می شود نتوانیم مقادیر را به راحتی در سیستم تولید و ذخیره کنیم. بهتر است یک بار دیگر به عبارتی که قبل حل عددی مسئله و به دست آوردن لامبدا ها رسیده بودیم نگاهی بیندازیم.



همانطور که مشاهده می شود ما در نهایت فقط از ضرب داخلی ۲ بردار استفاده می کنیم. بنابراین تنها کافیست ما ضرب داخلی خود را تعریف کنیم و دیگر لازم نیست کل فضای خطی و تبدیلات آن را تعریف کنیم. مشکلی که به وجود می آید این است که ما یک ضرب داخلی تعریف کردیم چطور صحت آن را بسنجیم. Mercer Theorem به ما می گوید که اگر ضرب داخلی ما در عبارت زیر به ازای هر تابع g صدق کند شرایط مطلوب را دارد.

$$\iint g(x)K(x,y)g(y)\,dx\,dy \geq 0$$

تریک استفاده از کرنل هم دقیقا همین موضوع را می گوید یعنی این که ما برای تغییر فضای داده ی خود و خطی کردن آن تنها کافیست که ضرب داخلی صحیحی برای داده هایمان تعریف کنیم.

$$k(x,x') = \boldsymbol{\phi}(x)^{\mathrm{T}}\boldsymbol{\phi}(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x')$$

شاید باز هم تعریف ضرب داخلی مقداری پیچیدگی داشته باشد بنابراین در بسیاری موارد می تواند از کرنل های مشهور استفاده کرد.

Examples of Kernel Functions

- **Polynomial** kernel with **degree** d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d$$

- **Gaussian kernel** with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$$

  - Closely related to **radial basis function neural networks**
  - The feature space is **infinite-dimensional** (it still be written as a dot product in a new feature space k(**x**, $\mathbf{x}_0$) = Φ(**x**) ∗ Φ($\mathbf{x}_0$), only with an **infinite number of dimensions**)

- **Sigmoid** with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa\mathbf{x}^T\mathbf{y} + \theta)$$

  - It does not satisfy the **Mercer condition** on all κ and θ

A. Dehaqani, UT 35

همچنین می تواند با روش های مختلفی آنها را ادغام کرد از جمله آنها می توان به مواردی زیر که در Pattern Recognition and Machine Learning - Bishop - page 296 آمده اشاره کرد.

Techniques for Constructing New Kernels.

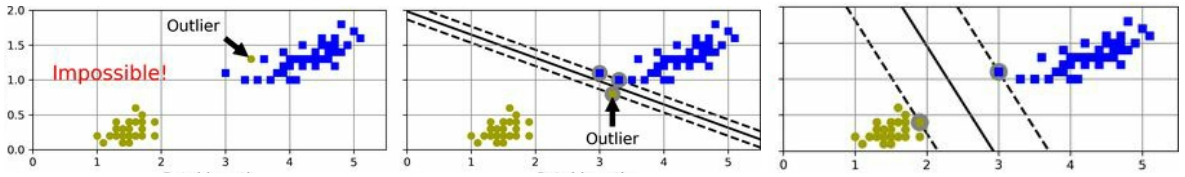
Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= ck_1(\mathbf{x}, \mathbf{x}') && (6.13)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') && (6.14)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.15)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.16)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (6.17)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') && (6.18)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' && (6.20)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.21)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.22)
\end{aligned}$$

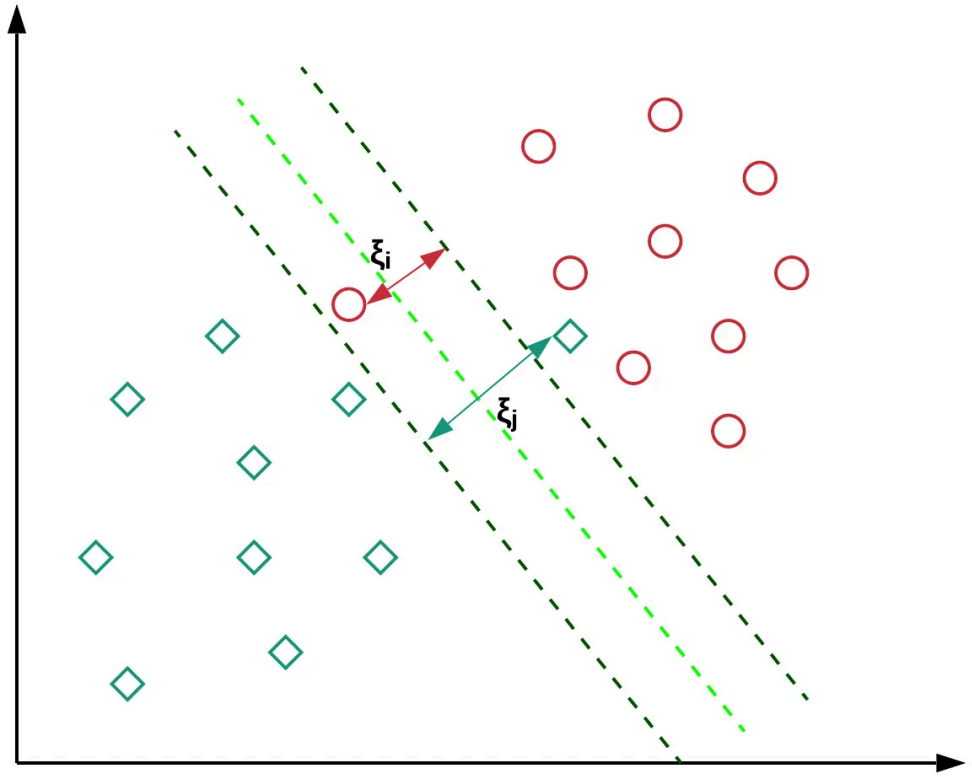
where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

تفاوت soft SVM با hard SVM

اگر ما فرض کنیم که داده ها به صورت خطی جدایی پذیر هستند و مدل خود را طبق آن پیاده کنیم hard SVM داریم و اگر همچین فرضی درست نباشد مسئله بی جواب است. توضیحاتی که در بخش اول توضیح داده شد مربوط به این نوع SVM است.



حال به توضیح soft SVM می پردازیم. داده های زیر را در نظر بگیرید.



در بسیاری از موارد داده های ما با یک خط به صورت کامل جدا نمی شوند و مانند شکل داده های نویز وجود دارد برای حل این مشکل ما از soft SVM استفاده می کنیم. برعکس hard این روش می تواند داده هایی مانند تصویر بالا را جدا کند. ایده این روش این است که داده ها می توانند از SVM تبعیت نکنند اما ما برای آنها جریمه در نظر می گیریم و در مسئله بهینه سازی خود سعی می کنیم این مقدار را نیز مینیمم کنیم.

$$\min_{\omega,\, b} \; ||\vec{\omega}||_2^2 + c \sum_{i=1}^{n} e_i$$

$$s.t.\; y_i(x_i^T \omega + b) \geq 1 - e_i \qquad e_i \geq 0$$

e در واقع همان جریمه ی ما است که میزان فاصله از خطوط مرزی هر دسته در نظر گرفته می شود. (در شکل با سای نمایش داده شده) و در آخر بعد از جمع آنها در یک ثابت ضرب می کنیم. حال مشابه قسمت اول تابع لاگرانژ رو تشکیل می دهیم و مقدار بهینه را برای آن حساب می کنیم. نتیجه مانند قبل است فقط یک شرط مربوط به ثابت عددی به آن اضافه می شود.

$$\max_{\lambda} \; -\frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i^T x_j - \sum_i \lambda_i$$

$$\sum_{i=1}^{n} \lambda_i y_i = 0, \quad 0 \le \lambda_i \le C$$

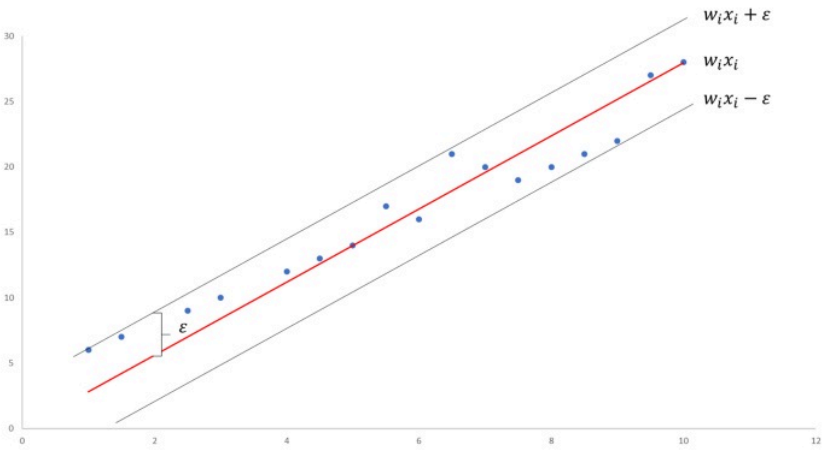
در نهایت اگر داده های به صورت خطی با بازه خوبی قابل تفکیک هستند روش hard مناسب است اما اگر داده ها مقداری با یکدیگر در هم آمیخته هستند یا به علت یک داده پرت SVM به شکل نامناسبی کوچک شده استفاده از روش soft بهتر است.

نحوه استفاده از SVM در مسائل رگرسیون

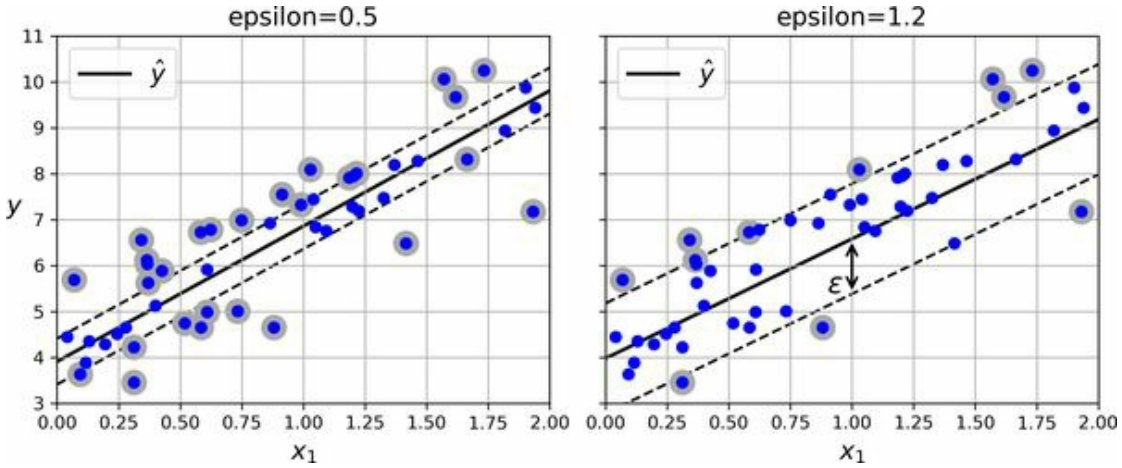
در این حالت ما مسئله را به این صورت تعریف می کنیم که خطی پیدا کنیم که بیشترین داده ها را در باند (خط چین های بالا و پایین) قرار دهد.

$$MIN \; \frac{1}{2}||w||^2$$

$$|y_i - w_i x_i| \le \varepsilon$$



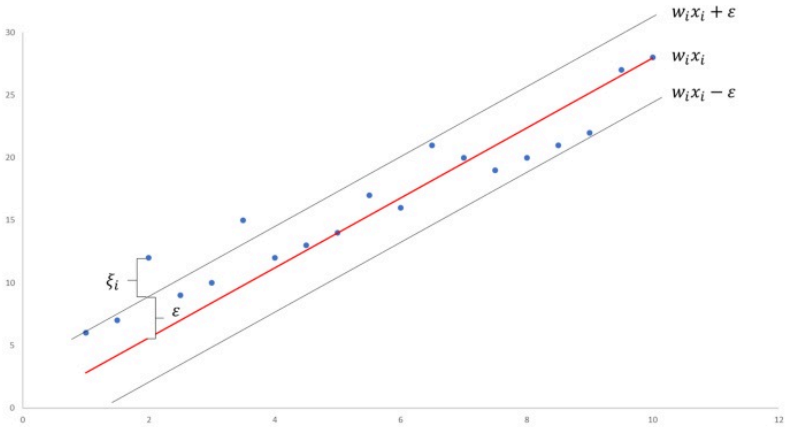
که اپسیلون توسط ما تعیین می شود تاثیر مقادیر مختلف اپسیلون را می تواند در تصویر زیر ببینید.



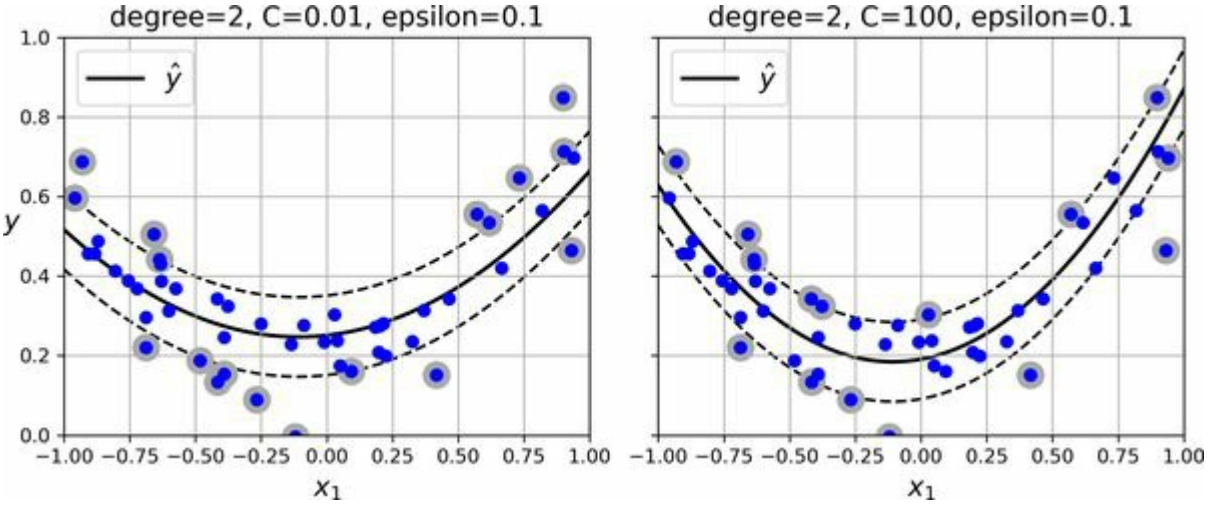
در نهایت کافیست مانند موارد قبلی این مسئله بهینه سازی را حل کنیم. همچنین مشابه قسمت های قبل که دسته بندی بود اینجا نیز چیزی مشابه soft تعریف می شود که به صورت زیر تعریف می شود.

$$MIN\ \frac{1}{2}\left|\left|\boldsymbol{w}\right|\right|^2 + C\sum_{i=1}^{n}|\xi_i|$$

$$|y_i - w_i x_i| \leq \varepsilon + |\xi_i|$$



همچنین مجددا می توانیم از کرنل در این قسمت نیز استفاده کنیم که نمونه ای از آن را مشاهده می کنید.



در مثال از کرنل polynomial درجه ۲ استفاده شده و همچنین soft SVM با ثابت ۱۰۰ و ۰.۰۱ و همچنین بازه باند که را ۰.۱ در نظر گرفته.