Python Machine Learning: The 2nd Book Circle

Classification Algorithm II: Logistic Regression, Bayes Logistic Regression, Linear SVM, Nonlinear Kernel SVM

Jianan Liu

Gothenburg

Nov, 24, 2016

Acknowledgement

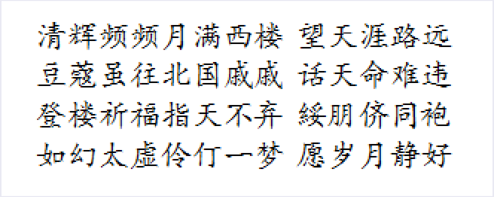
Acknowledgement

Here I would like to thank Mengbai Tao, for being a good audience. Clearly my understanding of SVM got better after explaining to him

For My Friends

This is also for two close friends, I really wish all the best for your life and hope sunshine will be available soon

Poetry



清辉频频月满西楼 望天涯路远
豆蔻虽往北国戚戚 话天命难违
登楼祈福指天不弃 綏朋侪同袍
如幻太虚伶仃一梦 愿岁月静好

Disclaimer

All opinions and statements in this presentation are mine and do not in any way represent the company

Any comment or correction of error is welcome, please contact me chisyliu@hotmail.com

The Second Book Circle of Python Machine Learning

In this presentation belongs to **algorithm** part of the book

- If you have time, please read the book first. This slide could be used as complementary resource for the book
- We will try to emphasize the key points in chapter 3
- We will try to go through every algorithm in first half of the chapter 3 of book Python Machine Learning, also show the mathematics behind each algorithm. But we only use the mathematics conclusion to explain algorithm rather than showing mathematical derivation
- All of us need to debug the python code, in order to get practice of implementing machine learning algorithm

Overview

Outline for Section 1

Overview of Chapter 3

- No single classifier works best across all possible scenarios. We always need to select model for a concrete scenario
- We step into using python library(e.g. scikitlearn)
- Remember to divide the known data set into training data set and validation data set, also important to follow the **cross validation** method to reuse the known data several times to get different training/validation data set
- **One vs Rest**, stands for iteratively dividing multi-class into two classes per iteration. By doing this, we can use binary classifier several times to get result of multi-class classifier

Overview of Chapter 3

- Chapter 3 explains more models and algorithms which have better performance over perceptron and adaline for classification
- The algorithms include logistic regression, support vector machine(maximize margin and kernel), decision tree, k nearest neighbor
- Although logistic regression is NOT hard to be understood, but the relative knowledge in this section of book regarding regularization is important
- Support vector machine(maximize margin and kernel) is hard to be understood, due to requiring of lots of knowledge in **optimization**

Data in Chapter 3

From this slide we go through the classification algorithms logistic regression, SVM in chapter 3

Training data set ($\mathbf{y}$, $\mathbf{X}$)

Suppose we have L samples, each sample has D features/dimensions(So the input $\mathbf{X}$ is L by D matrix, label $\mathbf{y}$ is L by 1 vector). If we only consider the l^{th} data sample pair, $\mathbf{x}_l$ is 1 by D vector which represents D features/dimension for l^{th} training data sample, $\mathbf{w}$ is D by 1 vector which represents weight and y_l represents the label of l^{th} data sample

Outline for Section 2

Scikit Learn Logistic Regression

- This page introduces where to find the corresponding methods in scikit learn package, please come back current page and check the link after finishing up all the contents in this section
- Logistic regression pages in scikitlearn introduces how to setup parameter of logistic regression in scikitlearn

  ▸ Link1: logistic regression

  ▸ Link2: logistic regression CV

Logistic Regression: Probability Version of Linear Classifier

We know the binary classifier needs to classify the data into either y $= +1$ or y $= 0$, which means we have y $= +1$ if $P(y = +1|\mathbf{x}, \mathbf{w}) > P(y = 0|\mathbf{x}, \mathbf{w})$, otherwise y $= 0$

- On the other word, y $= +1$ if $\ln(\frac{P(y=+1|\mathbf{x},\mathbf{w})}{P(y=0|\mathbf{x},\mathbf{w})}) > 0$, otherwise y $= 0$
- Note $P(y = +1|\mathbf{x}, \mathbf{w})$ stands for **likelihood** of model parameter $\mathbf{w}$
- Note $P(y = +1|\mathbf{x}, \mathbf{w}) = 1 - P(y = 0|\mathbf{x}, \mathbf{w})$
- The bigger $P(y = +1|\mathbf{x}, \mathbf{w})$, the more confident y $= +1$.

Logistic Regression: Probability Version of Linear Classifier

Now recall the linear classifier, we seek a "**line**" to classify the data in case the data is linear separable

Logistic Regression Model

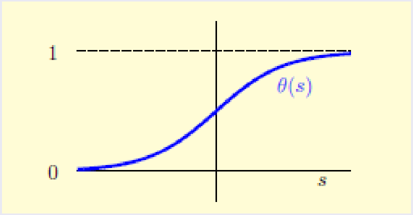
Recall linear function $\mathbf{xw}$ we used in perceptron/adaline as "**line**", here we want to use the same way to define $\ln(\frac{P(y=+1|\mathbf{x},\mathbf{w})}{P(y=0|\mathbf{x},\mathbf{w})})$

- $\ln(\frac{P(y=+1|\mathbf{x},\mathbf{w})}{P(y=0|\mathbf{x},\mathbf{w})}) = \mathbf{xw}$
- Since $P(y=0|\mathbf{x},\mathbf{w}) = 1 - P(y=+1|\mathbf{x},\mathbf{w})$, we could get expression of $P(y=+1|\mathbf{x},\mathbf{w})$, which is $P(y=+1|\mathbf{x},\mathbf{w}) = \frac{e^{(\mathbf{xw})}}{1+e^{(\mathbf{xw})}}$ by combining two equations

Logistic Regression: Probability Version of Linear Classifier

Sigmoid Function

- The mapping function which maps from **xw** to $P(y = +1|\mathbf{x}, \mathbf{w})$(Recall how we map in perceptron and adaline), $\theta(s) = \frac{e^s}{1+e^s}$, is sigmoid function



Logistic Regression: Probability Version of Linear Classifier

Now we have probability representing of $y^{(l)} = +1$,
$P(y^{(l)} = +1|\mathbf{x^{(l)}}, \mathbf{w}) = \frac{e^{(\mathbf{x^{(l)}}\mathbf{w})}}{1+e^{(\mathbf{x^{(l)}}\mathbf{w})}}$, for l^{th} training data sample

Combining Likelihood

Until now, our model is relative to one data sample. If we assume all the training data samples are independent, what is the model of joint probability of all y_l equal to the binary value it should be(same as observed labels of training samples)?

- $P(y_1, y_2, y_3, ..., y_L =$ observed label $|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_L, \mathbf{w}) = \prod_{l=1}^{L} P(y_l =$ observed label $|\mathbf{x_l}, \mathbf{w})$, where $P(y_l =$ observed label $|\mathbf{x}_l, \mathbf{w})$ is $\frac{e^{(\mathbf{x}_l\mathbf{w})}}{1+e^{(\mathbf{x}_l\mathbf{w})}}$ if $y_l = +1$, $\frac{1}{1+e^{(\mathbf{x}_l\mathbf{w})}}$ otherwise
- Note the new joint probability is still a likelihood of $\mathbf{w}$

Analytic Solution of Logistic Regression: Maximize Likelihood

By observation of the labels of training data, we want to maximize the possibility/probability of all labels are same as labels of training data(make all observed information of labels to be happened) by fixing the parameter $\mathbf{w}$ of model, so we try to maximize the joint probability(that is maximize likelihood) over $\mathbf{w}$

- $\max_{\mathbf{w}} \prod_{l=1}^{L} P(y_l = \text{observed label } | \mathbf{x}_l, \mathbf{w})$, where $P(y_l = \text{observed label } | \mathbf{x}_l, \mathbf{w})$ is $\frac{e^{(\mathbf{x}_l \mathbf{w})}}{1+e^{(\mathbf{x}_l \mathbf{w})}}$ if $y_l = +1$, $\frac{1}{1+e^{(\mathbf{x}_l \mathbf{w})}}$ otherwise

It could be written as

- $\max_{\mathbf{w}} \prod_{l=1}^{L} \theta(\mathbf{x}_l \mathbf{w})^{y_l} (1 - \theta(\mathbf{x}_l \mathbf{w}))^{1-y_l}$ so for each sample l it takes either $\theta(\mathbf{x}_l \mathbf{w})$ or $(1 - \theta(\mathbf{x}_l \mathbf{w}))$ depends on y_l. And $\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} \prod_{l=1}^{L} \theta(\mathbf{x}_l \mathbf{w})^{y_l} (1 - \theta(\mathbf{x}_l \mathbf{w}))^{1-y_l}$

Analytic Solution of Logistic Regression: Maximize Likelihood

By using log function

- $\max_{\mathbf{w}} \sum_{l=1}^{L} (y_l \log(\theta(\mathbf{x}_l \mathbf{w})) + (1 - y_l)\log(1 - \theta(\mathbf{x}_l \mathbf{w})))$
- so $\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} \sum_{l=1}^{L} (y_l \log(\theta(\mathbf{x}_l \mathbf{w})) + (1 - y_l)\log(1 - \theta(\mathbf{x}_l \mathbf{w})))$

We rewrite the maximization problem as minimization problem equivalently

- $\mathbf{w}_{ML} = \arg\min_{\mathbf{w}} \sum_{l=1}^{L} -(y_l \log(\theta(\mathbf{x}_l \mathbf{w})) + (1 - y_l)\log(1 - \theta(\mathbf{x}_l \mathbf{w})))$, so the objective function to be minimized is $\mathbf{J}(\mathbf{w}) = \mathbf{w} \sum_{l=1}^{L} -(y_l \log(\theta(\mathbf{x}_l \mathbf{w})) + (1 - y_l)\log(1 - \theta(\mathbf{x}_l \mathbf{w})))$

Beware maximize likelihood equals to minimize negative log(or ln) of likelihood, this trick will be seen quite frequently in optimization and deep learning in later circles

Analytic Solution of Logistic Regression: Maximize Likelihood

Maximize Likelihood v.s. Minimize Sum Squared Error

Let's recall perceptron/adaline. There when we tried to solve the parameter of model, we defined sum squared error(SSE) over all training data samples and tried to get the minimal value of SSE, is that **same logic** as the one used for logistic regression? Yes

- In logistic regression, we exam our model from maximizing likelihood point of view by fixing a parameter, in order to maximize the possibility of predicated output from model same as observed training data label(try to make the result happened as observed training data label)
- In adaline, we exam our model from minimizing the possible error point of view by fixing a parameter, in order to minimize the error between predicated output from model and observed training data label as small as possible

Analytic Solution of Logistic Regression: Maximize Likelihood

Beware that Maximize Likelihood is identical to Minimize Sum Squared Error if the variable/weight/parameter follows Gaussian distribution(e.g. model is linear Gaussian)

Train Logistic Regression Model by Batch Gradient Descent

```
1: Initialize the all elements $w_d$ in weight $\mathbf{w}$ to 0
2: **for** t in T **do**
3:     **for** d in D **do**
4:         computer $w_d(t+1) = w_d(t) + \eta \sum_l (y_l - \theta(\mathbf{x}_l \mathbf{w})) x_{d,(l)}$
5:         d = d + 1
6:     **end for**
7:     t = t + 1
8: **end for**
```

- y_l is the label of l^{th} sample training data
- T is the number of iteration
- $\theta(s) = \frac{e^s}{1+e^s}$

Train Logistic Regression Model by Batch Gradient Descent

- It is similar to the BGD algorithm for adaline, the only difference is using $\theta(\mathbf{x}_l\mathbf{w})$ to replace of $\mathbf{x}_l\mathbf{w}$

How do we get this rule of updating $\mathbf{w}$?

- The objective function $\mathbf{J}(\mathbf{w})$ to be minimized for logistic regression is convex, without any constraints
- By using the same method, batch gradient descent, we update the parameter $\mathbf{w}$ by taking a step away from the gradient $\triangledown \mathbf{J}(\mathbf{w})$ of our cost function $\mathbf{J}(\mathbf{w})$:
  $\mathbf{w}(t+1) = \mathbf{w}(t) + (-1)\eta \triangledown \mathbf{J}(\mathbf{w})$

Train Logistic Regression Model by Batch Gradient Descent

How do we get this rule of updating w?(continue.)

- $\bigtriangledown \mathbf{J}(\mathbf{w}) = \sum_l (y_l - \theta(\mathbf{x}_l \mathbf{w})) x_{d,(l)}$ for logistic regression
- The derivation could be found in page 64 of Python Machine Learning, we don't show the detail in this slide

- In the python codes, we neither actually calculate the gradient of objective function nor write the algorithm manually
- By setting the suitable argument of function LogisticRegression from sklearn, we feed the model by using training data and get the result

Bayesian Logistic Regression: L2 Norm Penalty Factor

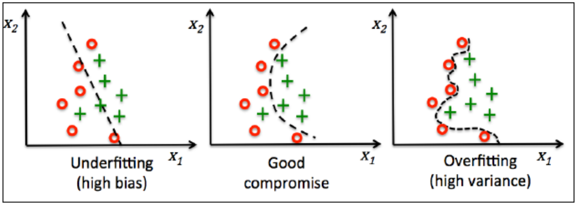
As we see, solution for logistic regression is kind of similar to solution for adaline binary classifier or LS linear regression, all of them are least square sense solution based on maximization of likelihood of observed data samples

- Recall the comparison of MMSE and LS solutions for linear regression model we mentioned on page29 first book circle. LS/ML solution has higher variance of $\mathbf{w}$ which could make **overfitting**
- We rather want MMSE/MAP solution due to low variance of $\mathbf{w}$, although it is bias
- So how could we lower variance of $\mathbf{w}$ in our model? From the other angle, saying how could we convert our solution from ML to MAP?

Bayesian Logistic Regression: L2 Norm Penalty Factor

Overfitting

- overfitting is something like you trust/rely on training data too much and overunderstand of your training data, the model with the parameter from your solution fit your training data too well to lose the ability of generic fitting of unkown data
- If number of training data samples is too small, it is hard to train the model to have ability of generic fitting of unkown data



Bayesian Logistic Regression: L2 Norm Penalty Factor

We add the L2 norm/regularization $\frac{\lambda}{2}\|\mathbf{w}\|^2 = \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$ in the following of objective function to be minimized as penalty factor

- The objective function for logistic regression becomes $\mathbf{J}(\mathbf{w}) = \mathbf{w}\sum_{l=1}^{L} -(y_l \log(\theta(\mathbf{x}_l\mathbf{w})) + (1-y_l)\log(1-\theta(\mathbf{x}_l\mathbf{w}))) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$
- Clearly seeing, the penalty factor $\frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$ becomes larger as increasing of $\mathbf{w}$, which leads the whole equation goes towards the reverse direction of minimization. By using this penalty, the model limits the variance of $\mathbf{w}$ which means it avoid overfiting somehow
- λ controls the model. In case $\lambda \longrightarrow 0$, the model becomes normal logistic regression.

Bayesian Logistic Regression: L2 Norm Penalty Factor

Actually this new objective function for logistic regression with L2 regularization penalty factor, is called bayesian logistic regression

- If parameter $\mathbf{w}$ follows normal/gaussian distribution, $\mathbf{w} \sim N(0, \lambda^{-1}\mathbf{I})$, the solution of $\mathbf{w}$ for this new objective function is actual a MAP solution
- As we discussed, MAP solution has lower variance which could avoid overfitting
- The relationship between logistic regression and bayesian logistic regression will be seen again when we see linear regression problem in the later chapters of book(Comparable with LS linear regression which is LS/ML solution and ridge linear regression which is MMSE/MAP solution)
- $\mathbf{w}_{MAP}$ could also be found/searched by using BGD, similar as way to search $\mathbf{w}_{ML}$ but with different gradient

Two Types of Objective Functions: Regularization for Avoid Overfitting

We now notice there are two types of objective functions, the normal one and the one with L2 norm/regularization penalty factor, as we promised we will see the exact details of relationship between linear regression and ridge regression in the later chapter of book. But now, let me show you a bit conclusion of relationship between linear regression and ridge regression, and an example, to let's have better understanding of relationship between two different objective functions

Two Types of Objective Functions: Regularization for Avoid Overfitting

Let's recall the linear regression model to be $y_l = \mathbf{x}_l \mathbf{w}$ as the example here

- Firstly we have the norm objective/cost function as sum squared error $\mathbf{J}(\mathbf{w}) = \sum_l (y_l - \mathbf{x}_l \mathbf{w})^2$
- Then when we setup objective/cost function by adding up the L2 regularization penalty factor on top of SSE, which is called ridge regression, $\mathbf{J}(\mathbf{w}) = \sum_l (y_l - \mathbf{x}_l \mathbf{w})^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}$

Two Types of Objective Functions: Regularization for Avoid Overfitting

- From optimization point of view, the solution for normal objective function is least square solution $\mathbf{w}_{LS} = (\mathbf{X}^T\mathbf{X})^{(-1)}(\mathbf{X}^T\mathbf{y})$ and solution for L2 regularization penalty factor based objective function becomes MMSE solution $\mathbf{w}_{MMSE} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{(-1)}(\mathbf{X}^T\mathbf{y})$
- From statistical point of view, normal objective function (stands for LS criteria) is equivalent to maximize likelihood(max. $P(X|w)$) while L2 regularization penalty factor based objective function (stands for MMSE criteria) is equivalent to maximize a posterior(max. $P(w|X)$) for the linear regression model $\mathbf{y} = \mathbf{X}\mathbf{w}$ with gaussian noisy.

Two Types of Objective Functions: Regularization for Avoid Overfitting

- The L2 regularization penalty factor of parameter w is the prior information. Usually we want MAP solution cause MAP works for the ensemble data set(all possible data) but ML works for the train samples(it means solution of normal objective function can work pretty well for your training data set but it might not work well for the coming unknown data)
- From matrix theory point of view, the LS solution can fall into situation where the number of training data is smaller than dimension of training data that lead to the matrix equations $\mathbf{y} = \mathbf{X}\mathbf{w}$ has no solution. By adding the L2 regularization penalty factor will give MMSE solution which changes the rank of $\mathbf{X}$, then made the non invertible matrix $\mathbf{X}^T\mathbf{X}$ becomes invertible matrix $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$. It solves problem when you dont have enough training data

Two Types of Objective Functions: Regularization for Avoid Overfitting

- As we mentioned in the first book circle, for our linear regression model, ML/LS solution is unbias $\mathbb{E}(\mathbf{w}_{LS}) = \mathbf{w}_{LS}$, but the variance for w can be extremely big which leads to overfitting(Althoght LS has the best performance in variance among possible unbias estimators), while MAP/MMSE solution is bias but the variance is smaller than LS. It means solution for L2 regularization penalty factor based objective function could avoid overfitting
- If we have extremely large number of training data for LS solution(approach infinity) and stationary process: $\mathbf{w}_{LS}$ is equivalent to $\mathbf{w}_{MMSE}$

Two Types of Objective Functions: Example

Let's see an example which might explain all the conclusion we listed in previous slides(**Although I don't think it is a suitable example in correctness of showing Posterior, anyhow we can get some feeling on relationship between likelihood and posterior**)

- we have a dice, we do 100 rolling of it and the result is 10 times "1", 20 times "2", 20 times "3", 5 times "4", 5 times "5", 40 times "6"
- Where the results of 100 roll is our training data(observed samples)
- If I ask what is probability you see of 6 which is $40/100 = 0.4$. And this is something like likelihood, 0.4 is something like ML(is comparable of LS solution in this example) solution which stands for norm objective function

Two Types of Objective Functions: Example

- But all of us know the probability of 6 is 1/6, this is something like posterior, 1/6 could be understood as a MAP (which is comparable of MMSE solution in this example) solution which stands for L2 regularization penalty factor based objective function
- We clearly find out the ML/LS solution could has very well explanation of what you observed(of training data) but it wont work for the real world(ensemble data set), MAP/MMSE is the truth of ensemble data
- If you roll dice more than 10000000 times, you for sure get almost all 1/6 for every "1", "2", "3", "4", "5", ... It means $\mathbf{w}_{LS}$ is equivalent to $\mathbf{w}_{MMSE}$ when the number of training data grow to 10000000

Outline for Section 3

Support Vector Machine

From here we start to learn another type of classifier, Support Vector Machine. Prof. Bernhard Boser in UC Berkeley, together with his wife Dr. Isabelle Guyon and Dr. Vladimir Vapnik from Bell Lab invented the SVMs close to their current form with kernel trick 25 years ago, which made SVM classifier can be applied for nonlinear separable data

- The funny thing is, Prof. Bernhard Boser's career was, and is now still within the analog and digital circuit, and even MEMS filed, which are highly relative to semiconductor industry rather than scientific computation, mathematics engineering, machine learning, signal processing or financial engineering, although SVM was dominated technology in machine learning and highly impacted all fields I mentioned during the end of last century.

Support Vector Machine

- His co-author, Dr. Vladimir Vapnik from Bell Lab actually proposed SVM in 1960s but it only worked for linear separable data
- My own comments here, Bell Lab is a really innovative place which created most of fundamental technologies in our information/networked society today
- Transistors, laser, information theory, FEC channel code, SVM, Unix, C, C++, Massive MIMO,...
- Wiki page of SVM is well written
  ▸ Link
- **SVM is (one of) the best classifier in both theory and practice**

Scikit learn SVM

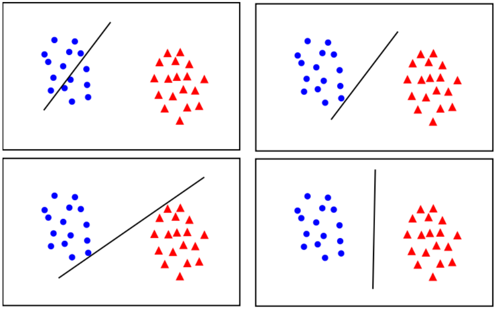
- This page introduces where to find the corresponding methods in scikit learn package, please come back current page and check the link after finishing up all the contents in this section
- SVM page in scikitlearn introduces how to setup parameter of SVM in scikitlearn

  ▸ Link: SVM

Maximum Hard Margin Classifier

Now let us look at the first idea of SVM: Maximum hard margin classifier

- Previously, we saw how perceptron, adaline and logistic regression are used to find out 'a line' to divide data into two parts when data set is linear-separable
- We also noticed 'the line' could be found(converged) in lots of places for the same data set



Maximum Hard Margin Classifier

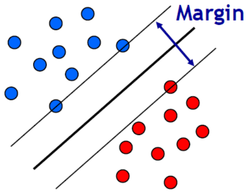
- In the figure shown last slide, we obviously don't want the first result
- But, previous introduced classifiers could get results as either figure 2 or 3. Which one is better?
- It seems the last result looks the best, why? Which classifier could have this result?

Result in figure 4 is the best

- Result in figure 4 is actually the best we prefer. Cause intuitively we would like to design the classifier which has maximal 'safety room(margin)' for unkown future data
- The bigger 'safety room(margin)' we have, the less chance unknown coming data in this gap((margin)) place will be classified into wrong side

Maximum Hard Margin Classifier

We specify the result in figure 4, what we prefer is actual something as this



- We try to find the maximum margin which classifies all observed training data correctly when the data is linear-separable
- The 'line' in the middle of margin is the classifier, this is called **maximum hard margin SVM**
- Then the task is, how to define classifier in figure 4 and find out the 'line' in case of data is linear-separable?

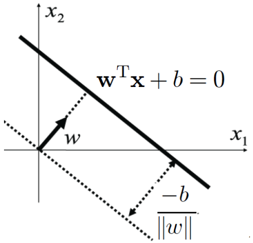
Maximum Hard Margin Classifier

We previously mentioned 'line' to divide observed training data into two types, but the 'line' is a line for 2D space(which means number of dimension/features of training data $\mathbf{x}$ is only 1). For generic N D space(number of dimension/features of training data $\mathbf{x}$ is larger than 1) the 'line' is actual a hyperplane

- Recall our perceptron or adaline, how do we define the hyperplane? $y = \mathbf{xw}$, which could be written as $y = \mathbf{xw} + w_0$ if we define first dimension/feature of observed data x_0 is 1(we normalized all features by divided first feature)
- Now we still want to use this hyperplane to divide observed training data into either 1 or -1 but with maximizing the 'gap'(margin) between two types

Maximum Hard Margin Classifier

Firstly look at what is the exact value of distance between hyperplane $\mathbf{xw} + b = 0$ and hyperplane in parallel goes across the origin($\mathbf{xw} = 0$)
Note I used the figure from Bastian Leibe's machine learning course, the $\mathbf{w}^T\mathbf{x} + b = 0$ in figure is same as $\mathbf{xw} + b = 0$ in our presentation



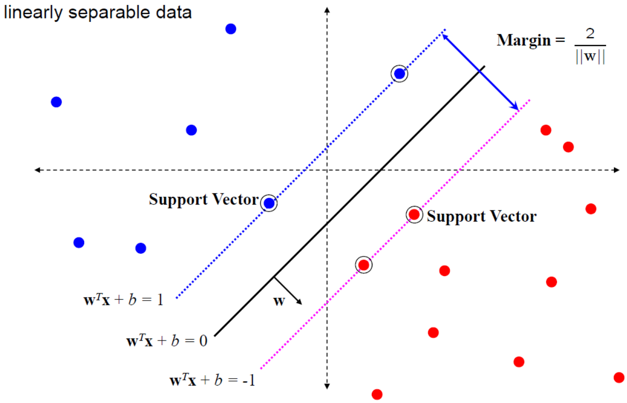
- The distance between them is $\frac{b}{||\mathbf{w}||}$(easy to be calculated)

Maximum Hard Margin Classifier

Now we could think about how to represent the margin between two hyperplanes which represent $\mathbf{xw} + w_0 = -1$ and $\mathbf{xw} + w_0 = 1$ respectively, in the distance sense

- From last figure we see the distance between $\mathbf{xw} + b = 0$ and $\mathbf{xw} = 0$ is $\frac{b}{||\mathbf{w}||}$
- So what is the distance between $\mathbf{xw} + b = 1$ and $\mathbf{xw} + b = -1$? In the other word. the distance between $\mathbf{xw} + b - 1 = 0$ and $\mathbf{xw} + b + 1 = 0$
- Yes, it is $\frac{2}{||\mathbf{w}||}$
- The figure next page from Andrew Zisserman's machine learning course will show the distance clearly

Maximum Hard Margin Classifier


linearly separable data
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$
$\mathbf{w}$

Maximum Hard Margin Classifier

Now we already got the equation to represent the margin which we with to maximize, together with some conditions. So we could define our problem mathematically

- $\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|}$
  subject to(s.t.) $\mathbf{xw} + b \geq 1$ if $y = 1$; $\mathbf{xw} + b \leq -1$ if $y = -1$
- Equivalently we rewrite our problem as a minimization

hard margin SVM as minimization optimization

- $\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$
  s.t. $y_l(\mathbf{x}_l\mathbf{w} + b) \geq 1$ for all l $= 1, 2, \ldots$ L

This is a quadratic optimization problem subject to linear constraints, which is typical convex problem. It means there is a unique minimum

Maximum Soft Margin Classifier

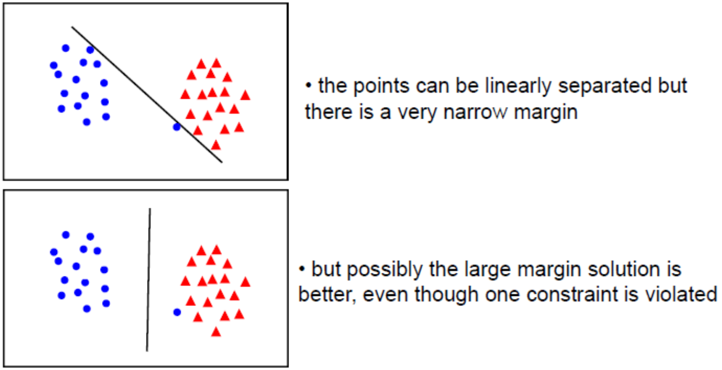
Previously we already defined hard margin SVM mathematically, the optimization problem represent a classifier which tries to classify exactly all observed training data into either 1 or -1 as they should be, without allowing any of these observed training data to be wrongly classified

Question

- Any problem with this hard margin criteria?

Maximum Soft Margin Classifier

Let's look at this figure from Andrew Zisserman's machine learning course


• the points can be linearly separated but there is a very narrow margin
• but possibly the large margin solution is better, even though one constraint is violated

- Which subplot is better classifier?

Maximum Soft Margin Classifier

- The linear classifier in first sub figure classifies the observed training data perfectly(which is hard margin SVM), but the maximum margin it could find is extremely narrow due a little bit circle type data is closer to triangle type
- The linear classifier in second sub figure classifies the observed training data NOT perfectly, but the maximum margin it could find is much wider(could be better for the generic situation which might better fits the unknow coming data)due the linear classifier allows a bit error(small amount of wrongly classified observed training data)
- The linear classifier in second sub figure is the idea of soft margin SVM

Maximum Soft Margin Classifier

- **For generic linear separable data set(which can be even little bit non-linear separable, e.g. small amount of data are hard to be separated by line completely correct), the linear classifier with bit error, like maximize soft margin SVM, has better performance on unknow coming data(more generic situation) than linear classifier classifies every data hardly correct, e.g. maximize hard margin SVM**
- This is similar to the reason why we have regularization by using penalty factor
- How to define soft margin mathematically?

Maximum Soft Margin Classifier

soft margin SVM as minimization optimization

- $\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_l \xi_l$
  s.t. $y_l(\mathbf{x}_l\mathbf{w} + b) \geq 1 - \xi_l$ for all l = 1, 2, ... L; $\xi_l \geq 0$

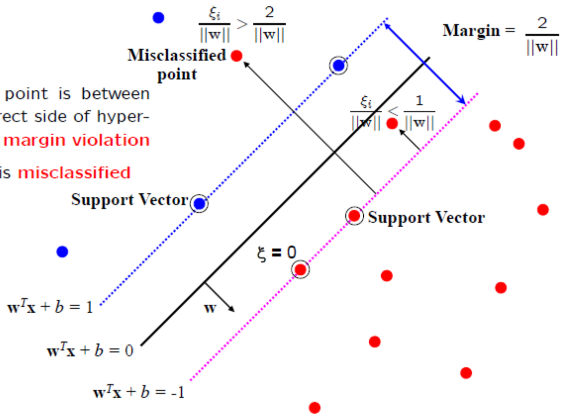
- where C is regularization parameter(a predefined fixed value), ξ_l is slack variable(It is a variable needs to be solved!)
- The objective function has '$C\sum_l \xi_l$' is because we want the sum of 'violation' as small as possible. The value of ξ_l could NOT be arbitrarily large, otherwise any of the hyperplane can be the hyperplane which forms the border of margin
- **If $0 \leq \xi_l \leq 1$, it represents classifier allows the training data to be in the margin but still on the correct side**
- **If $\xi_l > 1$, it represents classifier allows the training data to be on the wrong side**

Maximum Soft Margin Classifier

We still use figure from Andrew Zisserman's machine learning course to make the discussion last slide clearly

$\xi_i \geq 0$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a margin violation
- for $\xi > 1$ point is misclassified


$\frac{\xi_i}{||\mathbf{w}||} > \frac{2}{||\mathbf{w}||}$
Misclassified point
Margin = $\frac{2}{||\mathbf{w}||}$
$\frac{\xi_i}{||\mathbf{w}||} < \frac{1}{||\mathbf{w}||}$
Support Vector
Support Vector
$\xi = 0$
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Maximum Soft Margin Classifier

How about parameter C?

- C is a predefined parameter which we setup
- C actually controls the balance between the first part and second part in the objective function
- the first part, $\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$ stands for 'try to find the maximal margin'; the second part, $\min_{\mathbf{w}} C\sum_l \xi_l$ stands for 'minimize the wrongly classified training data'
- **In case C is too large, e.g. C $\rightarrow \inf$, in order to make $C\sum_l \xi_l$ as small as possible the ξ_l for each l needs to be $\rightarrow 0$. which means it becomes maximize hard margin SVM**
- **In case C is too small, e.g. C $\rightarrow 0$, ξ_l for each l will becomes extremely large which will make any hyperplane could be selected and margin extremely big**

Idea to Solve Maximum Margin Classifier: From Constraint Optimization to Unconstrained Optimization

Now we have the mathematical representation of both maximize hard and soft margin SVM. We call both of them as **linear SVM**

- Recall our perceptron, adaline or logistic regression model, we have unconstrained optimization problem there and we know the objective functions are convex, so we could solve them by using gradient descent search
- But now, both maximize hard and soft margin SVM are constrained optimization problem. What could we do for constrained optimization problem? We go to next subsection
- Note: The form of original linear SVM is actual a **Quadratic Programming** problem which could be solved by standard solver

Trick to Convert from Constraint Optimization to Unconstrained Optimization for Maximum Soft Margin Classifier

-

Gradient Descent Searching Algorithm for Maximum Soft Margin Classifier

-

A General Method to Convert to Unconstrained Optimization: Lagrange Multiplier

The answer to the question in last slide: We convert a constraint optimization problem to an unconstrained optimization problem. Let's start with maximum hard margin SVM to see how to do this

- In **optimization theory**, there is a method called **Method of Lagrange Multiplier** which convert a constraint optimization problem to an unconstrained optimization problem by adding constraint parts into objective function
- Let's see what is method of Lagrange multiplier in general

A General Method to Convert from to Unconstrained Optimization: Lagrange Multiplier

Method of Lagrange Multiplier

- For a general constraint optimization problem
  $\min_x f(x)$
  s.t.
  $h_i(x) \leq 0$, for all i = 1, 2, ... M;
  $l_j(x) = 0$, for all j = 1, 2, ... R;
- We could define **Lagrange Function** of original problem as $L(x, u, v) = f(x) + \sum_{i=1}^{M} u_i h_i(x) + \sum_{j=1}^{R} v_j l_j(x)$ by introducing **Lagrange Multipliers** u_i and v_j, where $\mathbf{u_i \geq 0}$
- Now we already got a $L(x, u, v)$ as an unconstrained objective function but has all the constraints plugged in it

Note: If all the constraints have been fulfilled , i.e. $l_j(x) = 0$, $u_i \geq 0$ and $h_i(x) \leq 0$, we can easily get $L(x, u, v) \leq f(x)$

A General Method to Convert from to Unconstrained Optimization: Lagrange Multiplier

Which means maximum of $L(x, u, v)$ over u and v is $f(x)$

- Let us try using 'max $L(x, u, v)$ over u and v' to replace $f(x)$

$\max_{u,v} L(x, u, v)$

- $\max_{u,v} L(x, u, v) = \max_{u,v}(f(x) + \sum_{i=1}^{M} u_i h_i(x) + \sum_{j=1}^{R} v_j l_j(x))$
- In order to make $\max_{u,v} L(x, u, v)$ equals to $f(x)$, we need to let $\sum_{i=1}^{M} u_i h_i(x)$ and $\sum_{j=1}^{R} v_j l_j(x))$ be as zero
- Cause $u_i \geq 0$, $\sum_{i=1}^{M} u_i h_i(x)$ could become a large positive value rather than 0 when maximize $L(x, u, v)$ in case of $h_i(x) > 0$
- $\sum_{j=1}^{R} v_j l_j(x))$ could become a large positive value rather than 0 when maximize $L(x, u, v)$ in case of $l_j(x) \neq 0$(due we do NOT have any constraint on v_j, v_j could be any number)

A General Method to Convert from to Unconstrained Optimization: Lagrange Multiplier

Which means when we max $L(x, u, v)$ over u and v, all the original constraints have to be fulfilled $\Longrightarrow$ All constraints have been hidden into the 'maximize the Lagrange function'

- Go further with this conclusion, we have $\min_x \max_{u,v} L(x, u, v)$ **which equals to the original constraint optimization problem**
  $\min_x f(x)$
  s.t.
  $h_i(x) \leq 0$, for all i = 1, 2, ... M;
  $l_j(x) = 0$, for all j = 1, 2, ... R;
- Till now, we find a way to convert original constraints optimization problem to unconstrained optimization problem

How to Solve Maximum Hard Margin Classifier: From Constraint Optimization to Unconstrained Optimization

Now let us go back to our hard margin SVM model. What is the equivalent problem which is represented by $\min_x \max_{u,v} L(x, u, v)$?

- First let us check the Lagrange function of hard margin SVM $L(b, \mathbf{w}, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{l=1}^{L} \alpha_l (1 - y_l(\mathbf{x}_l \mathbf{w} + b))$ with $\alpha_l \geq 0$
- Then we have our equivalent unconstrained problem of original hard margin SVM model, is $\min_{b,\mathbf{w}} \max_{\alpha, \alpha_l \geq 0} L(b, \mathbf{w}, \alpha) =$ $\min_{b,\mathbf{w}} \max_{\alpha, \alpha_l \geq 0} (\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{l=1}^{L} \alpha_l (1 - y_l(\mathbf{x}_l \mathbf{w} + b)))$

equivalent unconstrained problem of original hard margin SVM

- $\min_{b,\mathbf{w}} \max_{\alpha, \alpha_l \geq 0} (\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{l=1}^{L} \alpha_l (1 - y_l(\mathbf{x}_l \mathbf{w} + b)))$

How to Solve Maximum Hard Margin Classifier: From Constraint Optimization to Unconstrained Optimization

Until now, we already have the 'equivalent unconstrained format of our original constraint hard margin SVM'

- According to optimization theory, the 'equivalent unconstrained problem of our original constraint hard margin SVM' is convex due that $\frac{1}{2}\|\mathbf{w}\|^2$ is convex and $1 - y_l(\mathbf{x}_l\mathbf{w} + b)$ is affine
- Which means we have an unconstrained convex problem to be minimized, we have already known gradient descent search algorithm could be used
- However, we will see the other way to deal with the original constraint hard margin SVM by using Lagrange function in the next subsection

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

According to **optimization theory**, we define **Lagrange dual function** $g(u, v) = \max_{u,v} \min_x L(x, u, v)$

- So we have original constraint problem $\underbrace{f(x)}_{primal} =$
  $$\min_x \max_{u,v} L(x, u, v) \geq \underbrace{\max_{u,v} \min_{x} L(x, u, v) \equiv g(u, v)}_{dual}$$
- Dual problem $g(u, v)$ is always a convex function regardless the original $f(x)$ is or is NOT convex
- For any of general constraint optimization problem dual problem $g(u, v)$ gives a lower bound on $f(x)^*$ where $f(x)^*$ denotes the optimal value of original constraint problem(Primal problem), it means $g(u, v)^* \leq f(x)^*$ where $g(x)^*$ denotes the optimal value of dual

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

- Lower bound is NOT minimum of primal problem but it is something $\leq$ the minimum, this $\leq$ property is called weak duality
- Dual problem is a maximization problem over u and v but not x, we convert the minimization problem over x(primal) to maximization problem over u and v(dual)
- However, if the primal problem satisfies KKT condition(or is a convex problem), the dual problem will provide the maximum as exact same point as the minimum of primal, which means $g(u, v)^* = f(x)^*$. It is called strong duality(In this situation the duality gap $f(x)^* - g(u, v)^*$ will be zero)

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

- It means KKT is sufficiency, if our Lagrange dual problem satisfies KKT condition then dual problem will provide strong duality(usually if the objective function of primal problem is convex with linear constraints, strong duality holds)
- Besides, KKT is also necessity. It means if strong duality between primal and dual exits, KKT holds
- **So the conclusion is, we could solve dual problem instead of solving primal problem(the optimum will be same $g(u, v)^* = f(x)^*$) if strong duality holds**

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

- So question is, what is KKT condition?
- Let's recall general original constraint optimization problem
  $\min_x f(x)$
  s.t.
  $h_i(x) \leq 0$, for all i = 1, 2, ... M;
  $l_j(x) = 0$, for all j = 1, 2, ... R;
- We treat it to be as primal problem, the KKT condition for this general form primal problem is as next page

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

Karush-Kuhn-Tucker condition

- Primal feasibility: $h_i(x) \leq 0$, for all i = 1, 2, ... M; and $l_j(x) = 0$, for all j = 1, 2, ... R;
- Dual feasibility: $u_i \geq 0$ for all i = 1, 2, ... M;
- Complementary slackness: $h_i(x)u_i = 0$ for all i = 1, 2, ... M;

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

Now let's go back to our primal hard margin SVM problem

- As we know objective function of our primal problem(original hard margin SVM) is convex and the constraint is linear, the strong duality holds for hard margin SVM
- We could solve the dual instead to get the optimum of primal cause strong duality holds for hard margin SVM
- According to our previous analysis, equivalent of primal is $\min_{b,\mathbf{w}} \max_{\alpha,\alpha_l \geq 0} L(b,\mathbf{w},\alpha) =$ $\min_{b,\mathbf{w}} \max_{\alpha,\alpha_l \geq 0} (\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{l=1}^{L} \alpha_l(1 - y_l(\mathbf{x}_l\mathbf{w} + b)))$

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

- Then dual is

dual of primal hard margin SVM

- $g(\alpha, \mathbf{w}, b) = \max_{\alpha, \alpha_l \geq 0} \min_{b, \mathbf{w}} (\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{l=1}^{L} \alpha_l (1 - y_l(\mathbf{x}_l \mathbf{w} + b)))$

- First let's see what actually should be inside the inner part of our dual $\min_{b,\mathbf{w}} (\frac{1}{2}\|\mathbf{w}\|^2 + \sum_{l=1}^{L} \alpha_l (1 - y_l(\mathbf{x}_l \mathbf{w} + b)))$
- Cause we know $L(b, \mathbf{w}, \alpha)$ is convex for $\mathbf{w}$ and b, so we have $\frac{\partial L(b,\mathbf{w},\alpha)}{\partial b} = 0$ and $\frac{\partial L(b,\mathbf{w},\alpha)}{\partial w_d} = 0$ for $\min_{b,\mathbf{w}} L(b, \mathbf{w}, \alpha)$, where w_d is d^{th} element of vector $\mathbf{w}$(weight of each feature)

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

- So we have
  $\frac{\partial L(b,\mathbf{w},\alpha)}{\partial b} = 0 = -\sum_{l=1}^{L} \alpha_l y_l \Longrightarrow \sum_{l=1}^{L} \alpha_l y_l = 0$
- And we have
  $\frac{\partial L(b,\mathbf{w},\alpha)}{\partial w_d} = 0 = w_d - \sum_{l=1}^{L} \alpha_l y_l x_{l,d} \Longrightarrow \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l$
- So the dual $g(\alpha, \mathbf{w}, b)$ could be rewritten as the form without b and $\mathbf{w}$ by plugging the result we had from last two bullets:

$$\max_{\alpha_l \geq 0, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l} \left(-\frac{1}{2} \left\| \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l \right\|^2 + \sum_{l=1}^{L} \alpha_l\right)$$

- Note this new dual problem is only relative to variable α_l in the representation

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

dual of hard margin SVM with only variable α_l

- $g(\alpha, \mathbf{w}, b) = g(\alpha) =$

$$\max_{\alpha_l \geq 0, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l} \left(-\frac{1}{2} \| \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l \|^2 + \sum_{l=1}^{L} \alpha_l\right)$$

- Until now, we transfered the primal problem into another quadratic problem which has only relative to new variable α_l
- What is the physical meaning of this new form of dual problem? And more importantly, how could we map the result of optimum of new form of dual problem back to primal(what is relationship between α_l and **w** and b)?

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

Let's look at the first question on the last page. What is the physical meaning of this new form of dual problem?

- Recall KKT condition we introduced previously, KKT condition is both sufficiency and necessity for strong duality. We already had strong duality for our primal hard margin SVM, so the optimal solution of primal hard margin SVM needs to fulfill KKT condition

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

So we have KKT holds for hard margin SVM

KKT condition for hard margin SVM

- Primal feasibility: $1 - y_I(\mathbf{x}_I\mathbf{w} + b) \leq 0$
- Dual feasibility: $\alpha_I \geq 0$
- Complementary slackness: $\alpha_I(1 - y_I(\mathbf{x}_I\mathbf{w} + b)) = 0$

- Clearly from KKT condition, we know in order to satisfy $\alpha_I(1 - y_I(\mathbf{x}_I\mathbf{w} + b)) = 0$ with variable $\alpha_I \geq 0$, the only possible primal feasibility is $1 - y_I(\mathbf{x}_I\mathbf{w} + b) = 0$

Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

- It means only the data satisfy $y_l(\mathbf{x}_l\mathbf{w} + b) = 1$ will be the feasible solution of our hard margin SVM
- What does this mean physically?
- Only the data satisfy $y_l(\mathbf{x}_l\mathbf{w} + b) = 1$ will be the feasible solution of our hard margin SVM $\Longrightarrow$ Only the data/vector on the boundary will effect the margin of SVM, these vector on the boundary(satisfy $y_l(\mathbf{x}_l\mathbf{w} + b) = 1$) are so called 'support vector'
- First question answered:)

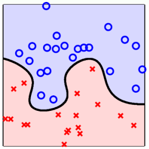
Maximum Hard Margin Classifier in Dual: Primal and Dual Problem in Optimization

Here we look at the second question, how could we map the result of optimum of new form of dual problem back to primal(what is relationship between α_l and $\mathbf{w}$ and b)?

- In case we already solved α_l (e.g. by using some standard SW solver for quadratic programming, to solve the quadratic problem 'dual of hard margin SVM with only variable α_l')
- We can then get $\mathbf{w}$ due that $\mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l$
- From KKT condition, we already got data satisfy $y_l(\mathbf{x}_l \mathbf{w} + b) = 1$. By this, we could calculate b by given $\mathbf{w}$ and α_l
- Second question answered:)

Maximum Hard Margin Classifier in Dual with Kernel Trick

- Until now, we have both maximize hard and soft margin SVM. Both maximize hard and soft margin SVM are linear SVM which tries to find a 'line'(hyperplane) to separate the data in case data is linear separable or a little bit nonlinear separable
- What if our data is totally nonlinear separable like this figure?

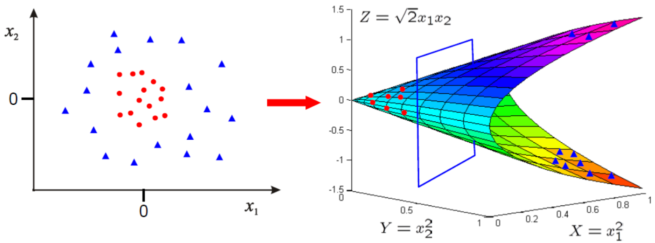


- Inspired by dual problem of SVM(e.g. our dual problem of maximize hard margin SVM), we could see how to apply kernel trick to deal with nonlinear separable data set

Maximum Hard Margin Classifier in Dual with Kernel Trick

- In order to make our nonlinear data set 'linear separable', we consider the way mapping data from low dimension to higher dimension. Let's look at the example from Andrew Zisserman's machine learning course

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \to \mathbb{R}^3$$



Maximum Hard Margin Classifier in Dual with Kernel Trick

- According to the figure shown last page, we clearly see the how to map nonlinear separable 2D data set(x_1, x_2) to 3D linear separable data set$(x_1^2, x_2^2, Z = \sqrt{2}x_1x_2)$ by applying mapping function Φ
- Φ is the feature map function which has $\mathbf{x}$ in $\mathbb{R}^D \longrightarrow \Phi(\mathbf{x})$ in $\mathbb{R}^Q$. Where Q stands for higher dimension than original D feature dimension. This mapping let new data set in higher dimension becomes linear separable
- Recall 'dual of hard margin SVM with only variable α_l'

$$\max_{\alpha_l \geq 0, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l} \left(-\frac{1}{2}\left\|\sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l\right\|^2 + \sum_{l=1}^{L} \alpha_l\right)$$

Maximum Hard Margin Classifier in Dual with Kernel Trick

- Where 'dual of hard margin SVM with only variable α_l' could be written like

$$\max_{\alpha_l \geq 0, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l} \left(-\frac{1}{2} \left\| \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l \right\|^2 + \sum_{l=1}^{L} \alpha_l \right)$$

$$= \max_{\alpha_l \geq 0, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l} \left(-\frac{1}{2} \sum_{l=1,k=1}^{L,L} \alpha_l \alpha_k y_l y_k \mathbf{x}_l \mathbf{x}_k^T + \sum_{l=1}^{L} \alpha_l \right)$$

Maximum Hard Margin Classifier in Dual with Kernel Trick

- Now we want to apply the feature mapping function Φ to map the data set $\mathbf{x}$ into higher dimension, it gives

$$\max_{\alpha_l \geq 0, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l} \left(-\frac{1}{2} \sum_{l=1,k=1}^{L,L} \alpha_l \alpha_k y_l y_k \Phi(\mathbf{x}_l) \Phi(\mathbf{x}_k)^T + \sum_{l=1}^{L} \alpha_l\right)$$

Maximum Hard Margin Classifier in Dual with Kernel Trick

- Now let's recall example in the previous figure from Andrew Zisserman's machine learning course. There we define feature mapping function Φ to map 2D data (x_1, x_2) to 3D data set$(x_1^2, x_2^2, Z = \sqrt{2}x_1x_2)$, number of elements in new data set: 3
- If we have more than 2 features for our data set in lower dimension, say 3 features, how many elements we will have for the new mapped data set in higher dimension? At most 19
- How about 4 features of out data set? 5?, ...
- Now we notice problem about this new method of mapping to higher dimension is, it is impossible to calculate the result of mapping $\Phi(\mathbf{x}_l)$ or $\Phi(\mathbf{x}_k)^T$ in higher dimension if the number of features of original data set is too large

Maximum Hard Margin Classifier in Dual with Kernel Trick

What should we do to avoid calculating $\Phi(\mathbf{x}_l)$ or $\Phi(\mathbf{x}_k)^T$?

- Before answering this question, let's first have a look at another example of defining Φ, 2nd order polynomial transform
- $\Phi_2(\mathbf{x}) = (1, x_1, x_2, ..., x_D, x_1^2, x_1x_2, x_1x_3, ..., x_1x_D, x_2x_1, x_2^2, x_2x_3, ..., x_2x_D, x_Dx_1, x_Dx_2, x_Dx_3, ..., x_D^2)$
- By using this $\Phi_2(\mathbf{x})$, the multiplying of $\Phi(\mathbf{x_l})\Phi(\mathbf{x_k})^T$ in our dual of hard margin SVM with only variable α_l and mapping function becomes $\Phi_2(\mathbf{x_l})\Phi_2(\mathbf{x_k})^T$

Maximum Hard Margin Classifier in Dual with Kernel Trick

- $$\Phi_2(\mathbf{x_l})\Phi_2(\mathbf{x_k})^T = 1 + \sum_{d=1}^{D} x_{l,d}x_{k,d} + \sum_{d=1}^{D}\sum_{d'=1}^{D} x_{l,d}x_{l,d'}x_{k,d}x_{k,d'}$$
  $$= 1 + \sum_{d=1}^{D} x_{l,d}x_{k,d} + \sum_{d=1}^{D} x_{l,d}x_{k,d} \sum_{d'=1}^{D} x_{l,d'}x_{k,d'}$$
  $$= 1 + \mathbf{x}_l\mathbf{x}_k^T + \mathbf{x}_l\mathbf{x}_k^T(\mathbf{x}_l\mathbf{x}_k^T) = K_{\Phi_2}(\mathbf{x}_l\mathbf{x}_k^T)$$
- Notice $\mathbf{x}_l\mathbf{x}_k^T$ is the inner product of vector $\mathbf{x}$, so $\mathbf{x}_l\mathbf{x}_k^T$ is actual a number rather than a matrix
- What does it matter for?

Maximum Hard Margin Classifier in Dual with Kernel Trick

- We want to avoid calculation of $\Phi(\mathbf{x_l})$ and $\Phi(\mathbf{x_k})^T$ separately
- In this 2nd order polynomial Kernel example case. Cause dual of hard margin SVM with only variable α_l and mapping function always needs result of $\Phi(\mathbf{x_l})\Phi(\mathbf{x_k})^T$ rather than calculating of $\Phi(\mathbf{x_l})$ and $\Phi(\mathbf{x_k})^T$ separately. So we could always use 2nd order polynomial Kernel K_{Φ_2} to calculate $K_{\Phi_2}(\mathbf{x}_l\mathbf{x}_k^T)$ rather than $\Phi(\mathbf{x_l})$ and $\Phi(\mathbf{x_k})^T$, the $K_{\Phi_2}(\mathbf{x}_l\mathbf{x}_k^T) = \Phi_2(\mathbf{x_l})\Phi_2(\mathbf{x_k})^T$
- More generally, we try to use function satisfies $K_{\Phi}(\mathbf{x}_l, \mathbf{x}_k^T) = \Phi(\mathbf{x_l})\Phi(\mathbf{x_k})^T$ (who could avoid calculation of $\Phi(\mathbf{x_l})$ and $\Phi(\mathbf{x_k})^T$ by calculation $K_{\Phi}(\mathbf{x}_l, \mathbf{x}_k^T)$ in lighter way) to be the Kernel function

Maximum Hard Margin Classifier in Dual with Kernel Trick

What are the commonly used Kernel function?

General Polynomial Kernel

- $K_\Phi(\mathbf{x}_l, \mathbf{x}_k^T) == \Phi(\mathbf{x_l})\Phi(\mathbf{x_k})^T = (\zeta + \gamma(\mathbf{x}_l\mathbf{x}_k^T)), \zeta \geq 0, \gamma > 0$

Gaussian/Radial Basis Function (RBF) Kernel

- $K_\Phi(\mathbf{x}_l, \mathbf{x}_k^T) = \Phi(\mathbf{x_l})\Phi(\mathbf{x_k})^T = exp(-\gamma\|\mathbf{x}_l - \mathbf{x}_k\|^2), \gamma > 0$

Maximum Hard Margin Classifier in Dual with Kernel Trick

- On page 77 of Python Machine Learning book, Gaussian/Radial Basis Function (RBF) Kernel has been introduced
- The choice of parameter γ will matter the result. It is similar to selection of parameter C in soft margin SVM
- **In case γ is too large, e.g. $\gamma \to \inf$, RBF kernel maps the any original to very high dimension space which makes any nonlinear separable data set linear separable, it leads to overfitting**
- **In case γ is too small, e.g. $\gamma \to 0$, the kernel function actually doesn't do the mapping to higher dimension, so it won't work well for nonlinear separable data setting**

Maximum Hard Margin Classifier in Dual with Kernel Trick

- We finally come to nonlinear maximize hard margin SVM, in the 'dual of hard margin SVM with only variable α_l and kernel' by selecting particular kernel function $K_{\Phi}(\mathbf{x}_l, \mathbf{x}_k^T)$, which can separate the nonlinear separable data

nonlinear hard margin SVM in dual form with only variable α_l and kernel

$$\max_{\alpha_l \geq 0, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l} \left(-\frac{1}{2} \sum_{l=1,k=1}^{L,L} \alpha_l \alpha_k y_l y_k K_{\Phi}(\mathbf{x}_l, \mathbf{x}_k^T) + \sum_{l=1}^{L} \alpha_l\right)$$

Revisit Maximum Soft Margin Classifier: Dual and Kernel

We already had the procedure that gets the 'dual of hard margin SVM with only variable α_l and kernel' for nonlinear separable data from 'original primal constraint hard margin SVM' for linear separable data. Currently we are trying to repeat the same procedure briefly for maximize soft margin SVM

- Recall primal constraint maximize soft margin SVM

primal constraint maximize soft margin SVM

- $\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_l \xi_l$
  s.t. $y_l(\mathbf{x}_l\mathbf{w} + b) \geq 1 - \xi_l$ for all l $=$ 1, 2, ... L; $\xi_l \geq 0$

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- Note we penalize the objective function with sum of margin violation: $\sum_l \xi_l$, the margin violation: ξ_l is linear constraints which leads the primal constraint maximize soft margin SVM is still a quadratic programming problem(as well as convex problem)
- We repeat the same way for hard margin SVM, firstly get the Lagrange function and equivalent unconstrained problem
- Note we have one more constraint $-\xi_l \leq 0$ for our equivalent unconstrained problem

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- The Lagrange function of soft margin SVM is $L(b, \mathbf{w}, \xi, \alpha, \beta) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{l=1}^{L}\xi_l + \sum_{l=1}^{L}\alpha_l(1 - \xi_l - y_l(\mathbf{x}_l\mathbf{w} + b)) + \sum_{l=1}^{L}\beta_l(-\xi_l)$ with $\alpha_l \geq 0$ and $\beta_l \geq 0$
- Then we have our equivalent unconstrained problem of original soft margin SVM model, is $\min_{b,\mathbf{w},\xi}\max_{\alpha_l\geq 0,\beta_l\geq 0} L(b, \mathbf{w}, \xi, \alpha, \beta)$

equivalent unconstrained problem of original soft margin SVM

- $\min_{b,\mathbf{w},\xi}\max_{\alpha_l\geq 0,\beta_l\geq 0}(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{l=1}^{L}\xi_l + \sum_{l=1}^{L}\alpha_l(1 - \xi_l - y_l(\mathbf{x}_l\mathbf{w} + b)) + \sum_{l=1}^{L}\beta_l(-\xi_l))$

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- Same as hard margin SVM. Due that the primal problem of soft margin SVM is convex problem as well, our duality gap is 0. It means solving dual problem is identical to solving of primal problem. So that we go for dual form of primal maximize soft margin SVM

dual problem of original soft margin SVM

- $g(\alpha, \beta, b, \mathbf{w}, \xi) \max_{\alpha_l \geq 0, \beta_l \geq 0} \min_{b, \mathbf{w}, \xi} (\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{l=1}^{L} \xi_l + \sum_{l=1}^{L} \alpha_l (1 - \xi_l - y_l (\mathbf{x}_l \mathbf{w} + b)) + \sum_{l=1}^{L} \beta_l (-\xi_l))$

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- Same as hard margin SVM. Due that the primal problem of soft margin SVM is convex/affine problem over $b, \mathbf{w}, \xi$ as well, so we could simplify the dual form according to $\frac{\partial L(b,\mathbf{w},\alpha)}{\partial b} = 0$, $\frac{\partial L(b,\mathbf{w},\alpha)}{\partial w_d} = 0$ and $\frac{\partial L(b,\mathbf{w},\alpha)}{\partial \xi_l} = 0$
- By using all these conditions, we could have the dual of soft margin SVM with only variable α_l as third step(The exact derivation we skip for now, anyone is interested in could refer to Lecture 4: Soft-Margin Support Vector Machine of Machine Learning Technique Course by Hsuan-Tien Lin)

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- So the dual of soft margin SVM with only variable α_l as below

dual of soft margin SVM with only variable α_l

- $g(\alpha, \beta, \mathbf{w}, b, \xi) = g(\alpha, \xi) =$

$$\max_{0 \le \alpha_l \le C, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l, \xi_l = C - \alpha_l} \left(-\frac{1}{2} \| \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l \|^2 + \sum_{l=1}^{L} \alpha_l\right)$$

- It is also quadratic programming problem

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- Later in the forth step, we still do same as for hard margin SVM. Due to strong duality holds, KKT condition needs to be fulfilled. So we can get the exact value of $\mathbf{w}, b, \xi$ from solved α, β(In case they are solved by standard solver for QP) by KKT condition
- The exact derivation we skip for now, anyone is interested in could refer to Lecture 4: Soft-Margin Support Vector Machine of Machine Learning Technique Course by Hsuan-Tien Lin, or derivate yourself by following the same way for hard margin SVM

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- Last step, we still do same as for hard margin SVM. Apply a kernel function on 'dual of soft margin SVM with only variable α_l' to make the the soft margin SVM works for nonlinear separable data set as well

nonlinear soft margin SVM dual format with only variable α_l and kernel

- $g(\alpha, \beta, \mathbf{w}, b, \xi) = g(\alpha, \xi) =$

$$\max_{0 \leq \alpha_l \leq C, \sum_{l=1}^{L} \alpha_l y_l = 0, \mathbf{w} = \sum_{l=1}^{L} \alpha_l y_l \mathbf{x}_l, \xi_l = C - \alpha_l} \left(-\frac{1}{2} \sum_{l=1,k=1}^{L,L} \alpha_l \alpha_k y_l y_k K_\Phi(\mathbf{x}_l, \mathbf{x}_k^T) + \sum_{l=1}^{L} \alpha_l\right)$$

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- By choosing Gaussian/Radial Basis Function (RBF) Kernel, $K_{\Phi}(\mathbf{x}_l, \mathbf{x}_k^T) = \Phi(\mathbf{x_l})\Phi(\mathbf{x_k})^T = exp(-\gamma\|\mathbf{x}_l - \mathbf{x}_k\|^2), \gamma > 0$, the soft margin SVM with Gaussian kernel will be modeled as in page 77 of Python Machine Learning book
- Let's revisit the parameter C and γ to give detailed discussion again for selecting of both parameters for soft margin SVM with Gaussian kernel

Revisit Maximum Soft Margin Classifier: Dual and Kernel

- **In case C is too large, e.g. C** $\rightarrow \inf$**, in order to make** $C\sum_l \xi_l$ **as small as possible the** ξ_l **for each l needs to be** $\rightarrow 0$**. which means it becomes maximize hard margin SVM**
- **In case C is too small, e.g. C** $\rightarrow 0$**,** ξ_l **for each l will becomes extremely large which will make any hyperplane could be selected and margin extremely big**
- **In case** γ **is too large, e.g.** $\gamma \rightarrow \inf$**, RBF kernel maps the any original to very high dimension space which makes any nonlinear separable data set linear separable, it leads to overfitting**
- **In case** γ **is too small, e.g.** $\gamma \rightarrow 0$**, the kernel function actually doesn't do the mapping to higher dimension, so it won't work well for nonlinear separable data setting**

Sequential Minimal Optimization Algorithm for SVM

So far we already had the dual format with only variable α_l and kernel for both hard/soft margin SVM which could work for nonlinear separable data well. But besides treat the model as QP problem to solve it by standard solver, any way else to solve the problem?
Yes, Sequential Minimal Optimization Algorithm(SMO) is one of the best algorithm to solve SVM

- SMO is NOT part of Python Machine Learning book, so we skip it for now and come back to discuss it later

Outline for Section 4

Conclusion

- Too lazy to give conclusion for 2nd circle
- It took me a month to finish this presentation:(

References

- Aarti Singh and Barnabas Poczos, Machine Learning Course, Carnegie Mellon University
- Andrew Zisserman, Machine Learning Course, Oxford University
- Hsuan-Tien Lin, Machine Learning Foundation Course, National Taiwan University
- Hsuan-Tien Lin, Machine Learning Technique Course, National Taiwan University
- Geoffrey J. Gordon and Ryan Tibshirani, Optmization Course, Carnegie Mellon University
- Bastian Leibe, Machine Learning Course, RWTH Aachen
- Sebastian Raschka, Python Machine Learning
- Zhihua Zhou, Machine Learning(Chinese Version)
- Hang Li, Statistical Learning(Chinese Version)

Question?