Linear Classification

Zezhi Wang, Jingguo Lan

University of Science and Technology of China

November 5, 2021

Outline

Classification

Problem

*Infer the **categorical** dependent variable with one or more predictor variables.*

Binary classification problem is a simplest example, which dependent variable $y \in \{0, 1\}$. The predictor variables could be listed as $\boldsymbol{x} := (1, x_1, ..., x_p)$, which contain the real-valued, binary, categorical variable, etc.

Example

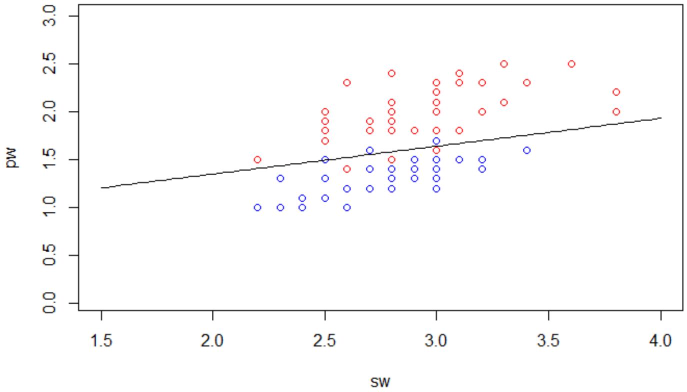


Figure: Data from iris.

Logistic Regression

Suppose $y|\boldsymbol{x} \sim Bernoulli(p(\boldsymbol{x}))$, i.e.

$$P(y = 1 \mid \boldsymbol{x}) = p(\boldsymbol{x}),$$
$$P(y = 0 \mid \boldsymbol{x}) = 1 - p(\boldsymbol{x}).$$

Denote i-th sample by $(\boldsymbol{x}_i, y_i)$, the log-likelihood function of which is:

$$\begin{aligned} L &= \sum_i \left[y_i \ln p\left(\boldsymbol{x}_i\right) + \left(1 - y_i\right) \ln\left(1 - p\left(\boldsymbol{x}_i\right)\right) \right] \\ &= \sum_i \left[y_i \ln \frac{p\left(\boldsymbol{x}_i\right)}{1 - p\left(\boldsymbol{x}_i\right)} + \ln\left(1 - p\left(\boldsymbol{x}_i\right)\right) \right]. \end{aligned}$$

Logistic Regression

Definition

For $p \in (0,1)$, logit function is defined by:

$$logit(p) = \ln \frac{p}{1-p}.$$

As its inverse, logistic function is defined by:

$$logistic(t) = \frac{1}{1+e^{-t}}.$$

In logistic regression (LR) model, we suppose there is a linear relationship between explanatory variable $\boldsymbol{x}$ and the logit of $p(\boldsymbol{x})$:

$$\ln \frac{p\left(\boldsymbol{x}_i\right)}{1-p\left(\boldsymbol{x}_i\right)} = \boldsymbol{x}_i^T \boldsymbol{\beta},$$

Link Function

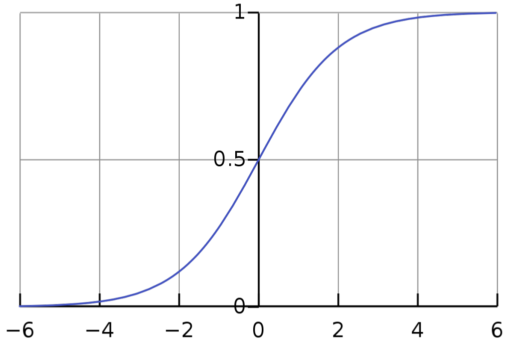


Figure: Logistic curve

In LR model, logit is called link function for it links linear estimator and expectation of dependent variable.

Likelihood

Thus, the log-likelihood function is:

$$L(\boldsymbol{\beta}) = \sum_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} \right) - \ln \left(1 + e^{\boldsymbol{x}_i^T \boldsymbol{\beta}} \right) \right].$$

At last, we use maximum likelihood estimation (MLE) method to estimate the parameter $\boldsymbol{\beta}$. Different from the least square estimation (LSE), the $\hat{\boldsymbol{\beta}}$ has no close form and is needed to be extracted through the optimization.

Objective Function

- Let $X = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})^T \in \mathbb{R}^{n \times p}, \boldsymbol{y} = (y_1, \ldots, y_n)^T \in \mathbb{R}^n$. Moreover, $\boldsymbol{p}(\boldsymbol{\beta}) = (p_1(\boldsymbol{\beta}), \ldots, p_n(\boldsymbol{\beta}))^T \in \mathbb{R}^n$, and $V(\boldsymbol{\beta}) = \operatorname{diag}(p_1(\boldsymbol{\beta})(1 - p_1(\boldsymbol{\beta})), \ldots, p_n(\boldsymbol{\beta})(1 - p_n(\boldsymbol{\beta})))$, where $p_i(\boldsymbol{\beta}) = \frac{1}{1+e^{-\boldsymbol{x}_i^T \boldsymbol{\beta}}}$.
- Therefore,

$$-L(\boldsymbol{\beta}) = \ln(\mathbf{1} - \boldsymbol{p}(\boldsymbol{\beta})) - \boldsymbol{\beta}^T X^T \boldsymbol{y}$$

- We can calculate the gradient and hessian of L

$$g(\boldsymbol{\beta}) = \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}\boldsymbol{\beta}}(\boldsymbol{\beta}) \frac{\mathbf{1}}{\mathbf{1} - \boldsymbol{p}(\boldsymbol{\beta})} - X^T \boldsymbol{y} = X^T(\boldsymbol{p}(\boldsymbol{\beta}) - \boldsymbol{y}),$$
$$H(\boldsymbol{\beta}) = X^T V(\boldsymbol{\beta}) X.$$

- $-L$ is convex because $X^T V(\boldsymbol{\beta}) X \succ 0$, and there exists no more than one $\hat{\boldsymbol{\beta}}$.

Newton Method

We can use Newton method to optimize the objective function. At c-th iteration:

$$\begin{aligned}\boldsymbol{\beta}^{(c+1)} &= \boldsymbol{\beta}^{(c)} - H^{-1}(\boldsymbol{\beta}^{(c)})g(\boldsymbol{\beta}^{(c)}) \\ &= \boldsymbol{\beta}^{(c)} - (X^T V^{(c)} X)^{-1} X^T (\boldsymbol{p}(\boldsymbol{\beta}^{(c)}) - \boldsymbol{y}) \\ &= (X^T V^{(c)} X)^{-1} X^T V^{(c)} \boldsymbol{z}^{(c)}\end{aligned}$$

where $V^{(c)} = V(\boldsymbol{\beta}^{(c)})$, $\boldsymbol{z}^{(c)} = X\boldsymbol{\beta}^{(c)} + \left(V^{(c)}\right)^{-1}\left(\boldsymbol{y} - \boldsymbol{p}(\boldsymbol{\beta}^{(c)})\right)$.
We can iteratively update $V^{(c)}$ and $\boldsymbol{z}^{(c)}$. Hence, the Newton Method for LR is usually called as iteratively re-weighted least squares.

Approximate Method

- When data dimension p is huge, it would cost much to inverse a big matrix.
- A useful method to avoid that is to find substitutions of matrix inverse, and trade off between time and accuracy. The problem of inversing a matrix or equally solving linear equations $A\boldsymbol{x} = \boldsymbol{b}$ can be seen as quadratic optimization problem:

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}.$$

- There are some optimization methods which possess the quadratic termination property and can get a sufficiently accurate solution economically. For example, conjugate gradient method is a good choice.

An approximate Newton Method

At the c-th iteration of Newton method, we need to solve a quadratic optimization problem:

$$\begin{aligned} &\boldsymbol{\beta}^{(c+1)} = (X^T V^{(c)} X)^{-1} X^T V^{(c)} \boldsymbol{z}^{(c)} \\ \Leftrightarrow\ & A\boldsymbol{\beta}^{(c+1)} = \boldsymbol{b} \\ \Leftrightarrow\ & \boldsymbol{\beta}^{(c+1)} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}, \end{aligned}$$

with $A = X^T V^{(c)} X$, $\boldsymbol{b} = X^T V^{(c)} \boldsymbol{z}^{(c)}$. This problem can be solved in finite iterations with conjugate gradient method.

Conjugate Direction

Conjugate Direction

Let A be an $n \times n$ symmetric and positive definite matrix, $d_1, d_2, \cdots, d_m \in R^n$ be non-zero vectors, $m \le n$. If

$$d_i^T A d_j = 0, \forall i \neq j$$

the vectors $d_1, d_2, \cdots, d_m$ are called A-conjugate.

Conjugate Gradient Method

We can sum up the above step by the following algorithm:

Algorithm 1 Conjugate gradient method for quadratic function

```
Require: $A, \boldsymbol{b}, \boldsymbol{x}^{(0)}, \epsilon$, IterMax
Ensure: $\hat{\boldsymbol{x}}$ such that $A\hat{\boldsymbol{x}} = \boldsymbol{b}$
  $\boldsymbol{g}^{(0)} = A\boldsymbol{x}^{(0)} - \boldsymbol{b}$
  $\boldsymbol{d}^{(0)} = -\boldsymbol{g}^{(0)}$
  $k = 0$
  while $\|\boldsymbol{g}^{(k)}\| > \epsilon$ and $k \leq$ IterMax do
    $s^{(k)} = \frac{{\boldsymbol{g}^{(k)}}^T \boldsymbol{g}^{(k)}}{{\boldsymbol{d}^{(k)}}^T A\boldsymbol{d}^{(k)}}$
    $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + s^{(k)}\boldsymbol{d}^{(k)}$
    $\boldsymbol{g}^{(k+1)} = A\boldsymbol{x}^{(k+1)} - \boldsymbol{b}$
    $\boldsymbol{d}^{(k+1)} = -\boldsymbol{g}^{(k+1)} + \frac{\|\boldsymbol{g}^{(k+1)}\|^2}{\|\boldsymbol{g}^{(k)}\|^2}\boldsymbol{d}^{(k)}$
    $k \leftarrow k + 1$
  end while
```

Linear Discriminant

- Predict: use the sigmoid function to predict the probability.

$$P(y = 1 \mid \boldsymbol{x}) = p(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}^T \hat{\boldsymbol{\beta}}}}$$

- Discriminant: Classify the object into the $y = 1$ population if $\boldsymbol{x}^T \hat{\boldsymbol{\beta}} \geq 0$.

Linear Discriminant Analysis

- In the binary classification setting, we define the log-odd of model as $\log\left(\frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})}\right)$.
- It's worth noting that the log-odds of Linear Discriminant Analysis (LDA) and LR are both linear functions of $\boldsymbol{x}$, therefore, the shape of the decision boundaries of them are both line or plane.

Linear Discriminant Analysis

Requested assumptions of LDA:

- Assume the marginal of $\boldsymbol{x}$ following:

$$p_{\boldsymbol{x}}(t) = \sum_k P(\boldsymbol{x} = t \mid Y = k) P(Y = k).$$

- Multivariate normality of the condition distribution of explanatory variables. (A difficult assumption to satisfy in practice)
- Complete equality of all of the underlying covariance matrices.

Advantage of LDA

- Under normality and equality of covariance assumptions, LDA is faster than LR, it's more economical and efficient in this case.
- The mean and variance of the data are sufficient statistics if normality condition holds, and it's natural to propose the dimension reduction for $\boldsymbol{x}$ by sufficient reduction technique.

Disadvantage of LDA

- These assumptions are too strong, especially explanatory variables are dichotomous zero-or-one variables.
- When the assumption does not hold, LDA estimator coefficients will not be consistent, we can't predict accurately even data is infinite.

Robustness

The LR would not be sensitive to the event with a fitting probability being close to 0 or 1, while all the samples would be equally contributed to the estimation of mean and covariance in LDA.

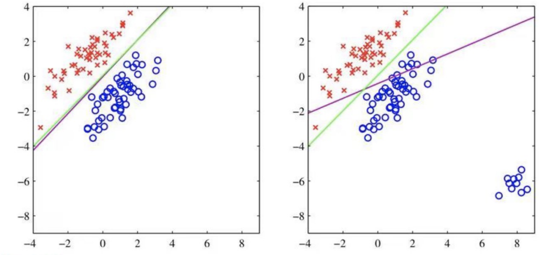
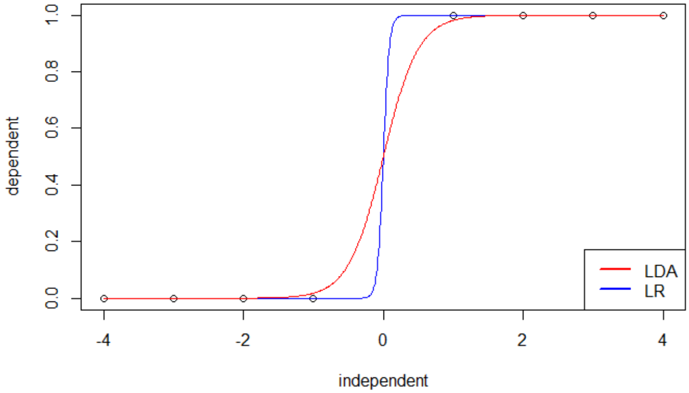


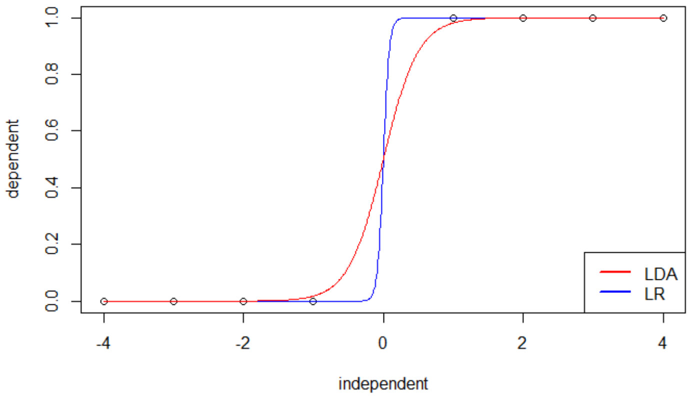
Figure: Fitting by LR (Green) and other method.

Identifiability



In the middle region, there is no unique solution and any curve is as good as any other curve in this region. LDA tends to mask the troublesome cases by not providing danger signals.

Warnings of LR



For LR, there may be two warnings:
Warning: algorithm did not converge.
Warning: fitted probabilities numerically 0 or 1 occurred.

Imbalance Data

Data that has an unequal class proportion can be considered as imbalanced data.

Commonly, the label y of data $(\boldsymbol{x}, y)$ in the majority class is set to $\{y = 1\}$, and it also denoted by event or positive. Other data with $\{y = 0\}$ is denoted by non-event or negative.

Unluckily, when trained on imbalanced data, LR model is biased towards the majority class.

Reasons

There are many reasons which can cause the imbalanced data problem.

- The amount of data is small.
- Sampling is unreasonable.
- The cost of observing event is much more than non-event.
- The event occurs rarely, such as fraud, infections by uncommon diseases, and delinquency in the credit card.

Difficulty

The reasons of difficulty when data is imbalanced:

- The data is biased, the fitted model performs better in the majority class.
- The classification problem is too complex, and the positive samples are too few to describe events' features.
- Premature convergence cause under-fitting.

Sampling

- **Random undersampling**
  Randomly remove samples from the majority class. Random undersampling can lead to information loss in the majority class.
- **Random oversampling**
  Randomly replicate the minority class samples. Random oversampling may lead to overfitting at the duplicated observations in the minority class.

SMOTE

- For informed oversampling, a widely used method is SMOTE (Synthetic Minority Over-Sampling technique).
- For continuous variables, we generate new events which are located on the line between an event and one of its k nearest event neighbors, where the k nearest neighbors are considered within the minority class observations.
- SMOTE generates the same number of synthetic samples for each original minority class observation, without consideration of increasing the occurrence of overlapping between classes.

Weighting

Find the MLE through a weighted log-likelihood function:

$$\sum_i w_{y_i} \left[\ln(1 - p_i(\boldsymbol{\beta})) - \boldsymbol{\beta}^T \boldsymbol{x_i} y_i \right],$$

where w_0 is the weight for non-event, and w_1 is the weight for event. The weight could be estimated from data or chosen based on prior information.

Weighting

Denote that the prior distribution $\boldsymbol{x}$ and y are $\pi(y)$ and $\pi(\boldsymbol{x})$, and the true conditional ditribution is $f(y|\boldsymbol{x})$. However what we can observe is the empirical distributions denoted by $f_s(y)$, $f_s(y)$ $f_s(y|\boldsymbol{x})$.

If suppose that $f(\boldsymbol{x}|y) = f_s(\boldsymbol{x}|y)$, we can find the likelihood in the population is:

$$\begin{aligned} f(y|\boldsymbol{x}) &= \frac{f(\boldsymbol{x}|y)\pi(y)}{\pi(\boldsymbol{x})} = \frac{f_s(\boldsymbol{x}|y)\pi(y)}{\pi(\boldsymbol{x})} \\ &= \frac{f_s(y|\boldsymbol{x})f_s(\boldsymbol{x})\pi(y)}{f_s(y)\pi(\boldsymbol{x})} \propto \frac{\pi(y)}{f_s(y)} f_s(y|\boldsymbol{x}) \\ &= w_y f_s(y|\boldsymbol{x}). \end{aligned}$$

where $w_1 = \frac{\pi(y=1)}{f_s(y=1)} := \frac{\tau}{\bar{y}}$, $w_0 = \frac{\pi(y=0)}{f_s(y=0)} := \frac{1-\tau}{1-\bar{y}}$.

Weighting

- It is recommended that set $w_1 = \frac{\tau}{\bar{y}}, w_0 = \frac{1-\tau}{1-\bar{y}}$, where $\bar{y}$ is the proportion of the events in the sample set, and τ is prior information about the proportion of event in the population.
- Without prior information, we can set $\tau = 0.5$ which is a default value in many software. It's a good choice that use cross validation method to choose τ.

Now, assuming that $\hat{p}$ and $\tilde{p}$ are the corrected or uncorrected probabilities for the minority class, then

$$\hat{p} = \frac{w_1 \tilde{p}}{w_1 \tilde{p} + w_0(1 - \tilde{p})}$$

The odd ratio would be

$$O = \frac{\hat{p}}{1 - \hat{p}} = \frac{w_1 \tilde{p}}{w_0(1 - \tilde{p})}$$

and the log odds is

$$\ln(O) = \ln\left(\frac{w_1}{w_0}\right) + \ln(\tilde{p}) - \ln(1 - \tilde{p})$$

which implies that

$$\mathbf{x}\hat{\boldsymbol{\beta}} = \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\bar{y}}{1-\bar{y}}\right)\right] + \mathbf{x}\tilde{\boldsymbol{\beta}}$$

Weighting

Prior correction is therefore easy to apply as it involves only correcting the intercept, $\boldsymbol{\beta}_0$, such that

$$\tilde{\boldsymbol{\beta}}_0 = \hat{\boldsymbol{\beta}}_0 - \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\bar{y}}{1-\bar{y}}\right)\right].$$

Prior correction requires knowledge of the fraction of events in the population, τ. The advantage of prior correction is its simplicity. However, the main disadvantage of this correction is that if the model is misspecified, then estimates on both $\hat{\boldsymbol{\beta}}_0$ and $\hat{\boldsymbol{\beta}}$ are less robust.

Introduction to SVM

Similar to logistic regression, we want to separate points into two classes.

- We want to separate them by a line or a hyperplane.
- There may be many lines that separate the data.

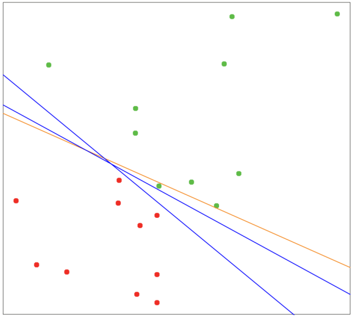


Figure: Separate points by hyperplane

Perceptrons

Perceptrons (Rosenblatt, 1958): Compute a linear combination of the input features and return the sign.[1]

- A response $y_i = 1$ is misclassified, then $\boldsymbol{w}^T\boldsymbol{x}_i + b < 0$, and the opposite for a misclassified response with $y_i = -1$. The goal is to minimize

$$D(\boldsymbol{w}, b) = -\sum_{i\in\mathbb{M}} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)$$

  where $\mathbb{M}$ indexes the set of misclassified points.

- The parameters w are updated via (stochastic gradient descent)

$$\begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix} \leftarrow \begin{pmatrix} \boldsymbol{w} \\ b \end{pmatrix} + \eta \begin{pmatrix} y_i\boldsymbol{x}_i \\ y_i \end{pmatrix}$$

- If the classes are linearly separable, it can be shown that the algorithm converges to a separating hyperplane in a finite number of steps.

Perceptrons

There are a number of problems with this algorithm:

- When the data are separable, there are many solutions, and which one is found depends on the starting values.
- The “finite” number of steps can be very large. The smaller the gap,the longer the time to find it.
- When the data are not separable, the algorithm will not converge, and cycles develop. The cycles can be long and therefore hard to detect.

Optimal hyperplane

An optimal hyperplane is here defined as the linear decision function with maximal margin between the vectors of the two classes. To construct such optimal hyperplanes one only has to take into acconnt a small amount of the training data, the so called support vectors, which determine this margin. [2]

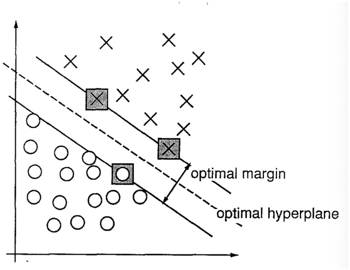


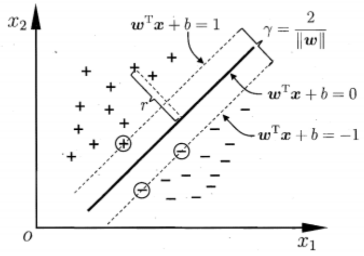
Figure: Optimal hyperplane

Notation

Our training data consists of n pairs $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$, with $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. Define a hyperplane by

$$\{\boldsymbol{x} \in \mathbb{R}^p | \boldsymbol{w}^T \boldsymbol{x} + b = 0\}$$

Geometrical margin: To solve the identification problem, we can assume $\boldsymbol{w}^T\boldsymbol{x} + b = 1$ or -1 for the support vectors $\boldsymbol{x}$ with label 1 or -1.



x_2
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 1$
$\gamma = \frac{2}{\|\boldsymbol{w}\|}$
r
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0$
$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = -1$
o
x_1

Figure: Geometrical margin

Optimal Problem

The problem we care can define as

$$\max_{\boldsymbol{w}} \frac{1}{||\boldsymbol{w}||_2} \quad s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \quad i = 1, 2, ..., n$$

which is equal to

$$\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2 \quad s.t. \quad y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \quad i = 1, 2, ..., n$$

KKT Condition

Consider our optimization problem:

$$\begin{cases} \text{Minimizing } f(\boldsymbol{x}) \\ \text{subject to } g_i(\boldsymbol{x}) \geq 0, h_j(\boldsymbol{x}) = 0 \text{ for } i = 1 : m, j = 1 : l \end{cases}$$

Then the Lagrange primal function becomes

$$L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\mu}^T g(\boldsymbol{x}) - \boldsymbol{\lambda}^T h(\boldsymbol{x})$$

If $\boldsymbol{x}^*$ is the local minimize of the optimization problem, it follows:

- Stationarity

$$\nabla f(\boldsymbol{x}^*) = \sum_{i=1}^{m} \mu_i \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^{l} \lambda_j \nabla h_j(\boldsymbol{x}^*)$$

- Primal feasibility

$$g_i(\boldsymbol{x}^*) \geq 0, \text{ for } i = 1, \cdots, m$$
$$h_j(\boldsymbol{x}^*) = 0, \text{ for } j = 1, \cdots, l$$

KKT Condition

- Dual feasibility

$$\mu_i \geq 0, \text{ for } i = 1, \cdots, m$$

- Complementary slackness

$$\mu_i g_i(\boldsymbol{x}^*) = 0, \ \forall i$$

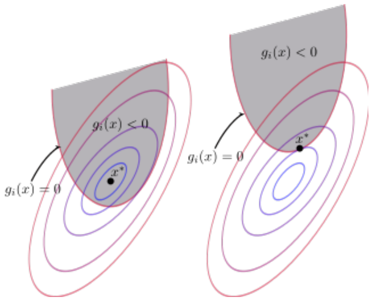


Figure: KKT

Lagrange function

Let $\boldsymbol{\alpha}$ be the dual variables, corresponding to Lagrange function is

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i (y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1)$$

Let derivative to 0,

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

$$\nabla_b L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = -\sum_{i=1}^{n} \alpha_i y_i$$

and hence

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

$$0 = \sum_{i=1}^{n} \alpha_i y_i$$

Dual Problem

So,

$$
\begin{aligned}
L(\boldsymbol{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j - \sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j - b\sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \\
&= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j
\end{aligned}
$$

The dual problem

$$
\begin{array}{ll}
\max_{\boldsymbol{\alpha}} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\
\text{s.t.} & \alpha_i \geq 0, i = 1, \ldots, n \\
 & \sum_{i=1}^{n} \alpha_i y_i = 0
\end{array}
$$

Boundary condition

As a result, in this problem, the KKT condition includes

$$\alpha_i(y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1) = 0 \quad \forall i$$

We may see from that:

- If $\alpha_i > 0$, then $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1$, implying that $\boldsymbol{x}_i$ is on the boundary.
- If $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) > 1$, $\boldsymbol{x}_i$ is not on the boundary, then $\alpha_i = 0$.
- Adding a new point $(\boldsymbol{x}_i, y_i)$ doesn't change the hyperplane if $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) > 1$.

Linear Non-Separable Case

- If the data is not linearly separable, there will be no feasible solution in which $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$ for all i. We therefore introduce slack variables $\xi_i \geq 0$ and replace the hard constraints with the soft margin constraints that $y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$.

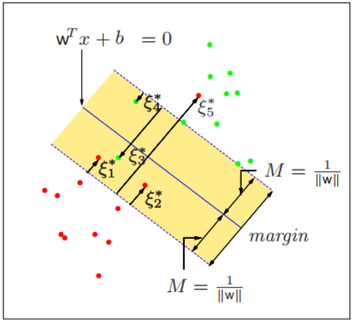


Figure: Linear Non-Separable Case

Linear Non-Separable Case

The new objective becomes

$$\min_{\boldsymbol{w},b,\xi} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

s.t. $\xi_i \geq 0,\ y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$. The cost parameter C is a constant to control the slackness. The corresponding Lagrangian for the soft margin classifier becomes

$$\begin{aligned} L(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\mu}) =& \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ &+ \sum_{i=1}^{n}\alpha_i(1 - \xi_i - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)) - \sum_{i=1}^{n}\mu_i\xi_i \end{aligned}$$

Dual Problem

Setting the derivative to 0,we have

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$$

$$0 = \sum_{i=1}^{n} \alpha_i y_i$$

$$C = \alpha_i + \mu_i$$

The dual problem becomes

$$\begin{array}{cl} \max_{\alpha} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{s.t.} & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n \end{array}$$

Connection with logistic regression

- We can rewrite the objective in SVM as follows[3]:

$$L(\boldsymbol{w}) = \sum_{i=1}^{n} \xi_{L1}(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)) + C||\boldsymbol{w}||^2$$

  where $\xi_{L1}(x) = \max(0, 1 - x)$. Similarly, we can define the L^2 version loss as $\xi_{L2}(x) = \max(0, 1 - x)^2$, which is forced to be differentiable at zero.

- By contrast, (penalized) logistic regression optimizes:

$$L(\boldsymbol{w}) = \sum_{i=1}^{n} \xi_{LR}(y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)) + C||\boldsymbol{w}||^2$$

  where $\xi_{LR}(x) = \log(1 + e^{-x})$.

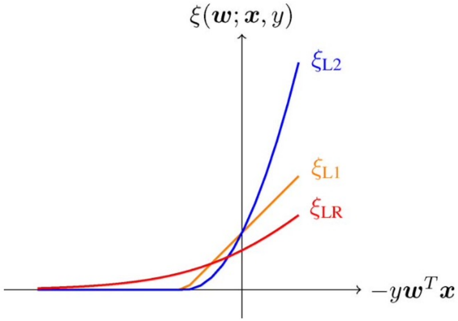


Figure: Curves of different loss functions

Recent Advanced Algorithm

In general, recent development suggests that there are several kinds of algorithms to solve the above linear classification problem with penalty term. We select two of them for solving the optimization problems with non-differentiable or differentiable losses under L^2 regularization.

Pegasos (A Stochastic gradient Method)

Consider L_2 Regularization and L_1 loss, i.e.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i)$$

- Pegasos takes the following subgradient direction of $f(\boldsymbol{w})$

  $$\nabla_i f(\boldsymbol{w}) = \boldsymbol{w} - Cy_i\boldsymbol{x}_i$$

  where $i \in B^+(\boldsymbol{w}) := \{i|1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i > 0\}$.
- Update $\boldsymbol{w}$ by $\boldsymbol{w} = \boldsymbol{w} - \eta\nabla_i f(\boldsymbol{w})$ for each $i \in B^+(\boldsymbol{w})$, where η is the learning rate.

Pegasos (A Stochastic Gradient Method)

- Since the optimal solution is proven to be in a ball set

$$\{\boldsymbol{w} \mid ||\boldsymbol{w}||_2 \leq \sqrt{Cn}\},$$

Pegasos introduces a projection step.

Algorithm 2 Pegasos

```
Require: $\boldsymbol{w}$ such that $||\boldsymbol{w}||_2 \leq \sqrt{Cn}$
  for $i = 1, 2, 3...$ do
    (a) Update the learning rate $\eta$.
    (b) Compute subgradient $\nabla_i f(\boldsymbol{w})$.
          If $i \in B^+(\boldsymbol{w})$, $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_i f(\boldsymbol{w})$.
    (c) Project $\boldsymbol{w}$ by $\min(1, \frac{\sqrt{Cn}}{||\boldsymbol{w}||_2}\boldsymbol{w})$.
  end for
```

- Our initial presentation and implementation of Pegasos included a projection step, while here we include it as an optional step.
- However, the newly revised analysis does not require such a projection and establishes almost the same guarantees for the basic (without projection) Pegasos algorithm. We did not notice major differences between the projected and unprojected variants in our experiments.

- While it's similarity to Perceptron algorithm, the uniqueness is guaranteed by the convexity.
- It is well known that SGD methods have slow convergence. However, they are suitable for large data because of accessing only one instance at a time. For data with many features, recent studies show that SGD is effective.
- They allow more flexible settings such as using more than one training instance at a time. At each step, a small random subset $B^+(w)$ is used instead of the full set.
- Since the convergence of algorithm is on basis on stochastic gradient descent, its running time guarantees are probabilistic. Hence, it highlights interesting tradeoffs between running time and data.

Trust Region Newton Method (TRON)

Consider L_2 Regularization and L_2 loss, e.g.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i)^2$$

- At each iteration, given an iterate w, a trust region interval Δ, and a quadratic model

$$q(\boldsymbol{d}) = \nabla f(\boldsymbol{w})^T\boldsymbol{d} + \frac{1}{2}\boldsymbol{d}^T\nabla^2 f(\boldsymbol{w})\boldsymbol{d}$$

  as an approximation of $f(\boldsymbol{w}+\boldsymbol{d}) - f(\boldsymbol{w})$.
- TRON solves the following subproblem in trust region method:

$$\min_{\boldsymbol{d}} q(\boldsymbol{d}) \quad \text{subject to} \quad ||\boldsymbol{d}|| \leq \Delta.$$

Trust Region Newton Method

By checking the ratio

$$\sigma = \frac{f(\boldsymbol{w}+\boldsymbol{d}) - f(\boldsymbol{w})}{q(\boldsymbol{d})}$$

of actual function reduction to estimated function reduction, TRON decides if w should be updated and then adjusts Δ. A large enough Δ indicates that the quadratic model $q(\boldsymbol{d})$ is close to $f(\boldsymbol{w}+\boldsymbol{d}) - f(\boldsymbol{w})$, so TRON updates w to be $\boldsymbol{w}+\boldsymbol{d}$ and slightly enlarges the trust region interval Δ for the next iteration. Otherwise, the current iterate w is unchanged and the trust region interval Δ shrinks by multiplying a factor less than one.

Trust Region Newton Method

Algorithm 3 Trust Region Newton Method

```
Require: $\boldsymbol{w}, \sigma_0, \Delta$
  for $k = 1, 2, 3...$ do
    (a) Find an approximate solution $\boldsymbol{d}$ of subproblem by the conjugate gradient method.
    (b) Check the ratio $\sigma$
    if $\sigma > \sigma_0$ then
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{d}$
    end if
    (c) Adjust $\Delta$ according to $\sigma$
  end for
```

Because of using high-order information (Newton directions), TRON gives fast quadratic local convergence.

References I

J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

J. C. Platt, “Probabilities for sv machines,” *Advances in Large Margin Classifiers*, p. 61, 2000.

G.-X. Yuan, C.-H. Ho, and C.-J. Lin, “Recent Advances of Large-Scale Linear Classification,” *Proceedings of the IEEE*, vol. 100, pp. 2584–2603, Sept. 2012.

References II

- S. J. Press and S. Wilson, “Choosing Between Logistic Regression and Discriminant Analysis,” p. 8.
- M. Maalouf, “Logistic regression in data analysis: an overview,” *International Journal of Data Analysis Techniques and Strategies*, vol. 3, no. 3, p. 281, 2011.
- G. King and L. Zeng, “Logistic Regression in Rare Events Data,” p. 28, 2001.
- Y. Li, “Addressing class imbalance for logistic regression,” 2020.