Chapter *4* Linear Classification

Siheng Zhang
zhangsiheng@cvte.com

2022 年 9 月 5 日

This part corresponds to **Chapter 4 of PRML, Chapter 9 of UML**, and mainly answers the following questions:

- How to implement ERM rule on linear discriminant functions? We will see that using square error as risk, it leads to least squares solution, and using negative likelihood (for binary class, or named 'cross entropy' for multi-class), which is a surrogate of '0-1' loss, it leads to logistic regression.
- The relation between logistic regression and maximum likelihood estimation (MLE). Read also the final part of Chapter 3.
- We will introduce Fisher's method and reveal its relation to least squares.

目录

1 Linear discriminant functions

In the last chapter, we stops at the linear classification of binary classification task,

$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0 = \sum_{j=1}^{d} w_j x_j + w_0 \tag{1}$$

in which $\mathbf{w}$ is weight vector, and w_0 is bias. For convenience, we introduce an additional input value $x_0 = 1$ and drop the bias notation. The input vector is assigned to class C_1 iff. $h(\mathbf{x}) \geq 0$ and to class C_2 otherwise.

Consider two points $\mathbf{x}_1, \mathbf{x}_2$ on the decision boundary, i.e., $\mathbf{w}^\top(\mathbf{x}_1 - \mathbf{x}_2) = 0$, hence $\mathbf{w}$ is orthogonal to the decision boundary. And the distance from the origin to the decision boundary is $\mathbf{w}^\top \mathbf{x}/\|\mathbf{w}\| = -w_0/\|\mathbf{w}\|$.

> ***remark1: extend to multiple classes. one-versus-the-rest*** *For each class $k = 1, 2, \ldots, K$, each classifier judge whether an example is C_k or not. So there are K classifiers needed;* ***one-versus-one*** *An alternative is to introduce $\frac{K(K-1)}{2}$ binary discriminant functions, one for every pair of classes (but will lead to ambiguous region).*

1.1 VC dimension

The class of linear function (Eq. 1) represents a hypothesis set *half-space*, denoted as HS_d. Its VC dimension is $d + 1$. It follows that we can learn half-space using the ERM paradigm with a sample size of $\Omega(\frac{d+\log(1/\delta)}{\epsilon})$.

> ***remark2: proof.*** *Firstly, we should show that any set of $d + 1$ points in $\mathcal{R}^d$ can be shattered by HS_d. Consider the set of vectors $\mathbf{0}, \mathbf{e}_1, \ldots, \mathbf{e}_d$ where for every i the vector $\mathbf{e}_i$ is the all zeros vector except 1 in the i-th entry. This set can be shattered by HS_d by set $\mathbf{w} = [y_1 - b, \cdots, y_d - b, b]$.*
>
> *Secondly, we should show there exists a point set of $d + 2$ points in $\mathcal{R}^d$ that cannot be shattered by half-space. Denote the points as $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{d+1}, \mathbf{x}_{d+2}$. There must be some $a_i, i = 1, 2, \cdots, d + 2$ (not all of them are zero) which satisfy that $\sum_{i=1}^{d+2} a_i \mathbf{x}_i = \mathbf{0}$. Split the a_i into two sets $I = i, a_i > 0$ and $J = i, a_i < 0$, then we have*
>
> $$\sum_{i \in I} a_i \mathbf{x}_i = \sum_{i \in J} |a_i| \mathbf{x}_i$$
>
> *If the VC dimension is $d + 1$, then there must be some $\mathbf{w}$ such that $\mathbf{w}^\top \mathbf{x}_i > 0, \forall i \in I$ and $\mathbf{w}^\top \mathbf{x}_i < 0, \forall i \in J$. It follows that*
>
> $$0 < \sum_{i \in I} a_i \mathbf{w}^\top \mathbf{x}_i = \mathbf{w}^\top \sum_{i \in I} a_i \mathbf{x}_i = \mathbf{w}^\top \sum_{i \in J} |a_i| \mathbf{x}_i = \sum_{i \in J} |a_i| \mathbf{w}^\top \mathbf{x}_i < 0$$
>
> *which leads to a contradiction.*

In the follow, we introduce different solutions to halfspaces. Except for the solutions in the chapter, we can also use a perceptron algorithm to implement ERM rule, which is left in the chapter of neural networks.

2 Linear programming

Linear Programming(LP) has the common form of

$$\min_{\mathbf{w}} \mathbf{u}^\top \mathbf{w}$$
$$s.t. \mathbf{A}\mathbf{w} \geq \mathbf{v}$$

In the realizable (namely, separable) case, we are looking for $\text{sign}(\mathbf{w}^\top \mathbf{x}_i) = y_i$, i.e., $y_i(\mathbf{w}^\top \mathbf{x}_i) > 0, \forall i$. Define $\gamma = \min_i y_i(\mathbf{w}^\top \mathbf{x}_i)$ and let $\mathbf{w}' = \frac{\mathbf{w}}{\gamma}$, then we have $\mathbf{w}'^\top(y_i \mathbf{x}_i) \geq 1$. With a dummy objective, it leads to LP's form.

3 Least squares

Least squares algorithm implements the ERM rule for the hypothesis class with respect to the squared loss. The weight vector can be determined by minimizing a sum-of-squares error function

$$\sum_i \|\mathsf{y}_i - \mathsf{w}^\top \mathsf{x}_i\|^2$$

Taking derivatives w.r.t. w and setting it to zero leads to

$$\mathsf{w} = \mathsf{X}^\dagger \mathsf{Y} = (\mathsf{X}^\top \mathsf{X})^{-1} \mathsf{X}^\top \mathsf{Y}$$

in which $\mathsf{X} \in \mathcal{R}^{n \times d}$ and $\mathsf{X}^\dagger$ is the *Moore-Penrose pseudo-inverse*.

Linear classification treats a problem with multiple targets to be multiple problems with single target each problem. If every target vector satisfies some linear constraint $\mathsf{a}^\top \mathsf{y} + b = 0$, then the model prediction given by least-squares solution is $\mathsf{a}^\top h(\mathsf{x}) + b = 0$. Thus if we use a one-hot encoding for K classes, the predictions made by the model will sum up to 1. However, this constraints alone is not sufficient to allow the model outputs to be interpreted as probability because they may not lie in the interval $(0, 1)$.

4 Fisher's linear discriminant

One way to view a linear classification model is in terms of *dimensionality reduction*, *i.e.*, projection from $\mathcal{R}^d$ to $\mathcal{R}$. By adjusting the components of the weight vector w, we can select a projection that maximizes the class separation. To begin with, consider a two-class problem (n_1 points of class C_1 and n_2 points of class C_2), so that the mean vectors of the two classes are given by $\mathsf{m}_k = \frac{1}{n_k} \sum_{\mathsf{x}_n \in C_k} \mathsf{x}_n, k = 1, 2$.

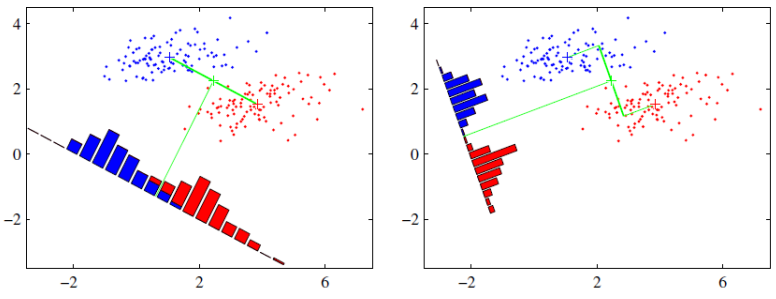
The simplest measure of the separation of the classes, when projected onto w, is the separation of the projected class means. This suggests that we might choose w so as to

$$\max_{\mathsf{w}} m_2 - m_1 = \mathsf{w}^\top (\mathsf{m}_2 - \mathsf{m}_1)$$

where $m_k = \mathsf{w}^\top \mathsf{m}_k$ is the mean of the projected data from class C_k.

This expression can be made arbitrarily large simply by increasing the magnitude of w. To solve this problem, we could constrain w to have unit length, i.e., $\|\mathsf{w}\|_2 = 1$. Using a Lagrange multiplier, it turns to maximize $\mathsf{w}^\top (\mathsf{m}_2 - \mathsf{m}_1) + \lambda(\|\mathsf{w}\|_2 - 1)$, which leads to $\mathsf{w} \propto \mathsf{m}_2 - \mathsf{m}_1$.

However, some outliers, which lays far from its class and close to the other class, may be mis-classified after projection, even though the dataset is linearly separable (see figure 4). This indicates that the objective above still needs to be improved. In fact, besides maximizing the separation margin, the projection is expected to reduce the inner-class variance. Denote the variance as $s_k^2 = \sum_{\mathsf{x}_n \in C_k} (\mathsf{w}^\top \mathsf{x}_n - m_k)^2$, the objective is given by



$$\max_{\mathsf{w}} J(\mathsf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathsf{w}^\top \mathsf{S}_B \mathsf{w}}{\mathsf{w}^\top \mathsf{S}_W \mathsf{w}} \tag{2}$$

in which $\mathsf{S}_B = (\mathsf{m}_2 - \mathsf{m}_1)(\mathsf{m}_2 - \mathsf{m}_1)^\top$, $\mathsf{S}_W = \sum_{k=1}^2 \sum_{\mathsf{x}_n \in C_k} (\mathsf{x}_n - \mathsf{m}_k)(\mathsf{x}_n - \mathsf{m}_k)^\top$. Setting the derivative to zero leads

to $(\mathbf{w}^\top \mathbf{S}_B \mathbf{w})\mathbf{S}_W \mathbf{w} = (\mathbf{w}^\top \mathbf{S}_W \mathbf{w})\mathbf{S}_B \mathbf{w}$. Since we just need the direction, we can neglect scalar factors. Besides, $\mathbf{S}_B \mathbf{w}$ is always in the direction of $\mathbf{m}_2 - \mathbf{m}_1$. So the optimizer is $\mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$.

> *remark3: Isotropic.* If $\mathbf{S}_W$ is unit matrix (i.e., the data distribution of each class is isotropic), then the solution degenerates to $\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1$.

4.1 Relation to least squares

The least-squares approach expect the **model predictions as close as possible to target values**. By contrast, the Fisher criterion was derived by **maximizing class separation in the output space**. It is interesting to see the relationship between these two approaches. In particular, we shall show that, **for the two-class problem, the Fisher criterion can be obtained as a special case of least squares**.

Denote n to be the total number of training samples, n_1 be the number of class 1 and n_2 be the number of class 2. For class 1, we take the label as n/n_1, for class 2, we take the label as $-n/n_2$, then we have $\sum_i y_i = n_1 * n/n_1 - n2 * n/n_2 = 0$.

Here, we consider the derivatives of sum of square error with regard to weight and bias,

$$\sum_i (y_i - \mathbf{w}^\top \mathbf{x}_i - w_0) = 0 \tag{3}$$

$$\sum_i (y_i - \mathbf{w}^\top \mathbf{x}_i - w_0)\mathbf{x}_i = 0 \tag{4}$$

Solving the Eq.3 leads to $w_0 = -\mathbf{w}^\top \mathbf{m}$, in which $\mathbf{m}$ is the mean of data. Bring it to Eq.4, we obtain that

$$\sum_i \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m})\mathbf{x}_i = \sum_i y_i \mathbf{x}_i$$

The right-side can be written as

$$\sum_i y_i \mathbf{x}_i = \sum_{\mathbf{x}_i \in C_1} y_i \mathbf{x}_i + \sum_{\mathbf{x}_i \in C_2} y_i \mathbf{x}_i = \frac{n}{n_1} n_1 \mathbf{m}_1 - \frac{n}{n_2} n_2 \mathbf{m}_2 = n(\mathbf{m}_1 - \mathbf{m}_2)$$

And the left-side can be written as

$$\begin{aligned}
&\sum_i \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m})\mathbf{x}_i = \sum_i \mathbf{w}^\top \mathbf{x}_i \mathbf{x}_i - \mathbf{w}^\top \mathbf{m} \sum_i \mathbf{x}_i \\
&= \sum_{\mathbf{x}_i \in C_1} \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m}_1 + \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1 + \mathbf{m}_1) + \sum_{\mathbf{x}_i \in C_2} \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m}_2 + \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2 + \mathbf{m}_2) - n\mathbf{w}^\top \mathbf{m}\mathbf{m} \\
&= \sum_{\mathbf{x}_i \in C_1} \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1) + \sum_{\mathbf{x}_i \in C_1} \mathbf{w}^\top \mathbf{m}_1 \mathbf{m}_1 + \sum_{\mathbf{x}_i \in C_2} \mathbf{w}^\top (\mathbf{x}_i - \mathbf{m}_2)(\mathbf{x}_i - \mathbf{m}_2) + \sum_{\mathbf{x}_i \in C_2} \mathbf{w}^\top \mathbf{m}_2 \mathbf{m}_2 - n\mathbf{w}^\top \mathbf{m}\mathbf{m} \\
&= S_W \mathbf{w} + n_1 \mathbf{m}_1 \mathbf{m}_1^\top \mathbf{w} + n_2 \mathbf{m}_2 \mathbf{m}_2^\top \mathbf{w} - n\mathbf{m}\mathbf{m}^\top \mathbf{w}
\end{aligned}$$

Also note that $\mathbf{m} = \frac{n_1}{n}\mathbf{m}_1 + \frac{n_2}{n}\mathbf{m}_2$, Eq.4 finally leads to:

$$(S_w + \frac{n_1 n_2}{n} S_B)\mathbf{w} = n(\mathbf{m}_1 - \mathbf{m}_2)$$

Note that $S_B \mathbf{w}$ is along the direction of $(\mathbf{m}_1 - \mathbf{m}_2)$, and neglect the magnitude again, we can obtain $\mathbf{w} \propto S_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$. The result is same as Fisher's method and we get the bias in addition.

4.2 For multi-class case

Now, denote the weight matrix as $\mathbf{W} \in \mathcal{R}^{d\times K}$, in which K is the number of classes. By the same criteria, that is, encourage the between-class covariance to be large and the within-class covariance to be small. We can maximize the following target:

$$\max \mathrm{Tr}\left((\mathbf{W}\mathsf{S}_W\mathbf{W}^\top)^{-1}(\mathbf{W}\mathsf{S}_B\mathbf{W}^\top)\right)$$

As a result, $\mathbf{W}$ is determined by those eigenvectors of $\mathsf{S}_W^{-1}\mathsf{S}_B$ that correspond to the K largest eigenvalues. Note that this also raises a restriction that $K < d$.

5 Logistic regression

From a probabilistic perspective, define *log odds* a to represent the log of the ratio of probabilities:

$$a = \log \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})}$$

Then the posterior can be expressed by '*sigmoid*' function $\sigma(a) = 1/(1+\exp(-a))$. We have shown in the last chapter that Gaussian distribution assumption leads to linear discriminant of this form. Besides, for any distribution in exponential family, this also holds. Denote $a = \mathbf{w}^\top\mathbf{x}$, logistic regression uses

$$p = 1/(1+\exp(-\mathbf{w}^\top\mathbf{x}))$$

for prediction. The ERM rule associated with logistic regression is ($y_i \in -1, 1$)

$$\arg\min_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m} \log(1+\exp(-y_i\mathbf{w}^\top\mathbf{x}_i))$$

or I prefer to write it as ($y_i \in 0, 1$)

$$\arg\min_{\mathbf{w}} \mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} (y_i \log p_i + (1-y_i)\log(1-p_i)) \tag{5}$$

For multi-class case, we define $a_i = \log(p(\mathbf{x}|C_i)p(C_i))$, then the posterior is

$$p(C_k|x) = \frac{\exp a_k}{\sum_j \exp a_j}$$

which is known as the 'softmax' function. And the ERM rule with '*one-hot*' encoding target (that is, $y_i \in \{0,1\}^K, \sum_{k=1}^K y_{ik} = 1$), a.k.a cross-entropy, is as below:

$$\arg\min_{\mathbf{w}} -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_{ik}\log\frac{\exp a_{ik}}{\sum_j \exp a_{ij}}$$

remark4: property of sigmoid function.

$$\sigma(-a) = 1/(1+\exp(a)) = \exp(-a)/(1+\exp(-a)) = 1-\sigma(a)$$
$$\frac{\mathrm{d}\sigma(a)}{\mathrm{d}a} = \exp(-a)/(1+\exp(-a))^2 = \sigma(a)(1-\sigma(a))$$

remark5: 'soft'-max. *Consider the maximum operator* $\max\{a_1, \ldots, a_k\}$. *Note that exponential operator quickly enlarge the gap of two positive scalar, we can approximate the* ***non-derivable*** *maximum operator by 'log-sum-exp'*

$$\max\{a_1, \ldots, a_k\} \approx \log \sum_j \exp a_j$$

In multi-class classification, we would like the posterior of true class is that maximize among that of all classes, *i.e.*, $a_i \geq \max\{a_1, \ldots, a_k\}$. Using the 'log-sum-exp' with exponential, it requires to maximize the following scalar for the true class C_i:

$$\frac{\exp a_i}{\sum_j \exp a_j}$$

That is what '*soft*'-max means.

How to solve logistic regression? Taking derivatives of its ERM goal leads to

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \sum_{i=1}^{m} (y_i(1-p_i) - (1-y_i)p_i) \frac{\partial p_i}{\partial \mathbf{w}} = \sum_{i=1}^{m} (y_i - p_i)\mathbf{x}$$

Hence, we can use gradient descent to solve $\mathbf{w}$. Gradient descent takes the following procedure:

Algorithm 1: GD procedure

Input: learning rate η
Output: $\mathbf{w}^*$

```
init: w_o, t = 0 while True do
    update w_{t+1} = w_t − η ∂L/∂w
    if ||w_{t+1} − w_t||_2 < ε then
        break;
    end
end
```

remark6: ***Newton-Raphson updating rule cannot be applied.*** *Some people may challenge gradient descent for it takes multiple iterations and requires manually selection of learning step (though it can be selected in a large range). They may recall that Newton-Raphson updating rule uses the Hessian matrix to determine the best learning step, i.e.,*

$$\mathbf{w} = \mathbf{w} - \mathsf{H}^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{w}}$$

However, the Hessian matrix in logistic regression is

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}^2} = -\sum_{i=1}^{m} p_i(1-p_i)\mathbf{x}\mathbf{x}^\top$$

which is not constant. In fact, it depends on $\mathbf{w}$ *through the value of* p_i*. Hence, this update cannot be applied directly. In the next chapter, we will show that, for linear regression, this update rule does work. It can find the global minima with only one step updating.*

6 Summary

We have shown two ways to implement ERM rule for linear classification,

- using square error as risk, it leads to least squares solution
- using negative likelihood (for binary class, or named 'cross entropy' for multi-class), which is a surrogate of '0-1' loss, it leads to logistic regression.

We have also revealed the relation between Fisher's method and least squares, as well as the relation between logistic regression and MLE.

Since we have seen that linear discriminant hypothesis space is flexible and may suffer from overfitting, we will come up with some techniques to address it, like regularization, MAP instead of MLE, and support vector machines (SVM), which introduces '*margin*' concept as another criterion. Indeed, these techniques are tightly connected at their intrinsic.

Besides, we have shown the gradient descent algorithm, which is the base of modern updating rules. We will go deeper in the chapter of neural network.

7 Exercises and solutions