

# Chapter 4 Linear Model

Siheng Zhang  
zhangsiheng@cvte.com

February 7, 2021

This part corresponds to **Chapter 1,3,4 of PRML, Chapter of UML**, and mainly answers the following questions:

- 

## Contents

<b>1</b>	<b>Linear classification</b>	<b>2</b>
1.1	Extend to multiple classes . . . . .	2
1.2	Fisher's linear discriminant . . . . .	2
<b>2</b>	<b>Linear regression</b>	<b>2</b>
2.1	Generalized linear regression . . . . .	3
2.2	Regularization <i>a.k.a</i> Bayesian linear regression . . . . .	3
2.2.1	Ridge regression . . . . .	4
2.2.2	Lasso . . . . .	4

# 1 Linear classification

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

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

in which  $\mathbf{w}$  is weight vector, and  $w_0$  is bias. 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

$$\frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|} = \frac{-w_0}{\|\mathbf{w}\|} \quad (2)$$

It is usually convenient to use a more compact notation in which we introduce an additional input value  $x_0 = 1$  and then define  $\tilde{\mathbf{w}} = (w_0, \mathbf{w})$  and  $\tilde{\mathbf{x}} = (x_0, \mathbf{x})$  so that  $h(\mathbf{x}) = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}$ . For simplification, we neglect the ‘tilde’ symbol below.

## 1.1 Extend to multiple classes

- *one-versus-the-rest* For each class  $k = 1, 2, \dots, K$ , 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  $K(K-1)/2$  binary discriminant functions, one for every pair of classes (but will lead to ambiguous region).

## 1.2 Fisher’s linear discriminant

One way to view a linear classification model is in terms of dimensionality reduction. By adjusting the components of the weight vector  $\mathbf{w}$ , we can select a projection that maximizes the class separation. To begin with, consider a two-class problem in which there are  $N_1$  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

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{\mathbf{x}_n \in C_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{\mathbf{x}_n \in C_2} \mathbf{x}_n \quad (3)$$

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

$$m_2 - m_1 = \mathbf{w}^\top (\mathbf{m}_2 - \mathbf{m}_1) \quad (4)$$

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

This expression can be made arbitrarily large simply by increasing the magnitude of  $\mathbf{w}$ . To solve this problem, we could constrain  $\mathbf{w}$  to have unit length, i.e.,  $\|\mathbf{w}\|_2 = 1$ . Using a Lagrange multiplier, it turns to maximize  $\mathbf{w}^\top (\mathbf{m}_2 - \mathbf{m}_1) + \lambda(1 - \|\mathbf{w}\|_2^2)$ , which leads to  $\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1$ .

# 2 Linear regression

In linear regression model, the model is the same except that the learning target  $y$  is continuous but not discrete. And the learning goal is the sum-of-square (SSE) loss

$$\min_{\mathbf{w}} L_S(h) = \sum_{i=1}^m l(h(\mathbf{x}_i)) = \sum_{i=1}^m (h(\mathbf{x}_i) - y_i)^2 = \sum_{i=1}^m (\mathbf{w}^\top \mathbf{x}_i - y_i)^2 \quad (5)$$

Suppose the fitting error  $\epsilon_i = y_i - \mathbf{w}^\top \mathbf{x}_i$  is Gaussian noise, i.e.,  $\epsilon_i \sim \mathcal{N}(0, \beta)$ . Then the log likelihood function of the training sequence is

$$\log \mathcal{L} = -\frac{m}{2} \log 2\pi\beta - \sum_{i=1}^m \frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\beta} \quad (6)$$

Obviously, MLE is equivalent to linear regression.

*remark1:* Since linear regression is not a binary prediction task, we cannot analyse its sample complexity using the VC-dimension. One possible analysis of the sample complexity of linear regression is by relying on the "discretization trick". However, to apply the sample complexity bounds from Chapter 2 we also need that the loss function will be bounded.

## 2.1 Generalized linear regression

The model is just a linear function of the input variables, and this imposes significant limitations on it. Therefore, extended model considers **linear** combination of fixed **non-linear** functions of the input variables, of the form

$$h(\mathbf{x}) = w_0 + \sum_{j=1}^d w_j \phi_j(x) \quad (7)$$

where  $\phi_j(x)$  are known as *basis functions*. Again, denote  $\phi_0(\mathbf{x}) = 1$  so that  $h(\mathbf{x}) = \tilde{\mathbf{w}}^\top \phi(\mathbf{x})$ . For Simplification, we also neglect the 'tilde' symbol from now on.

Now, consider the closed-form solution for The gradient of the log likelihood. The gradient of the SSE loss takes the form

$$\nabla L_S(h) = \sum_{i=1}^m \{y_i - \mathbf{w}^\top \phi(\mathbf{x}_i)\} \phi^\top(\mathbf{x}_i)$$

Setting it to zero gives

$$\mathbf{w} = \Phi^\dagger \mathbf{y} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y} \quad (8)$$

Here  $\Phi$  is an  $n \times d$  matrix, whose elements are given by  $\Phi_{nj} = \Phi_j(\mathbf{x}_n)$ . And  $\Phi^\dagger$  is *Moore-Penrose pseudo-inverse*.

*remark2, multiple-outputs:* A more general case is multiple outputs, i.e.,  $\mathbf{y}_i \in \mathcal{R}^k, k > 1$ . However, the solution to multiple-outputs regression problem decouples between the different target variables so we do not discuss it here.

*remark3, on-line learning:* Batch techniques involve processing the entire training set in one go, can be computationally costly for large data sets. For linear regression, stochastic gradient descent algorithm updates parameter using

$$\mathbf{w}^{t+1} = \mathbf{w}^t - lr * \nabla L_{(\mathbf{x}_t, y_t)}(h) = \mathbf{w}^t + lr * (y_t - (\mathbf{w}^t)^\top \phi(\mathbf{x}_t)) \phi(\mathbf{x}_t)$$

in which  $lr$  is the learning rate.

*remark4, basis:* There are three popular basis function: 1. **Polynomial basis**, 2. **Radical basis**, 3. **Fourier basis**.

## 2.2 Regularization *a.k.a* Bayesian linear regression

Recall the closed-form solution (Eq. 8) for linear regression problem, if  $n \leq d$ , the SSE loss can achieve zero. It means that the model capacity is enough for the training dataset. However, it may suffer from over-fitting problem. Consider the learning dataset generated from  $y$ .

.....

Further, consider the expected loss from the view of bias-variance trade-off. Using squared error, the expected loss is given by

$$\mathbb{E}(l) = \int \int l(h(\mathbf{x})) p(\mathbf{x}, y) d\mathbf{x} dy = \int \int (h(\mathbf{x}) - y)^2 p(\mathbf{x}, y) d\mathbf{x} dy$$

Setting its derivatives to zero leads to  $h(\mathbf{x}) = \int y p(\mathbf{x}, y) dy / p(\mathbf{x}) = \mathbb{E}(y|\mathbf{x})$ . Take it back to the expected loss, we can expand it as

$$\begin{aligned} \mathbb{E}(l) &= \int \int \{h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x}) + \mathbb{E}(y|\mathbf{x}) - y\}^2 p(\mathbf{x}, y) d\mathbf{x} dy \\ &= \int \int \left\{ [h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x})]^2 + [\mathbb{E}(y|\mathbf{x}) - y]^2 + 2[h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x})][\mathbb{E}(y|\mathbf{x}) - y] \right\} p(\mathbf{x}, y) d\mathbf{x} dy \\ &= \int \int \{h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \int \int \{\mathbb{E}(y|\mathbf{x}) - y\}^2 p(\mathbf{x}) d\mathbf{x} \end{aligned} \quad (9)$$

The first term depends on our choice for the function  $h(\mathbf{x})$ . Because it is non-negative, the smallest that we can hope to make this term is zero. If we had an unlimited supply of data (and unlimited computational resources), we could in principle find the regression function  $h(\mathbf{x})$  to any desired degree of accuracy. However, in practice we have only a finite number of data points. The second term, which is independent of  $h(\mathbf{x})$ , arises from the intrinsic noise on the data and represents the minimum achievable value of the expected loss.

.....

To address on this problem, Bayesian linear regression set a prior assumption of  $\mathbf{w}$ , and view the learning procedure to maximizing its posterior. Two of the most popular case is discussed below.

### 2.2.1 Ridge regression

Ridge regression addresses on over-fitting by penalizing the  $l_2$ -norm of weight vector  $\mathbf{w}$ ,

$$\min_{\mathbf{w}} \sum_{i=1}^m (\mathbf{w}\phi_i(\mathbf{x}) - y_i)^2 + \lambda \|\mathbf{w}\|_2^2$$

If we assume a Gaussian prior for the weight vector,  $\mathbf{w} \sim \mathcal{N}(0, \alpha^{-1}\mathbf{I})$ , then the posterior of the training sequence is:

$$p(\mathbf{w}|S) \propto p(\mathbf{w})p(S|\mathbf{w}) \propto \exp\left(-\frac{\alpha}{2}\mathbf{w}^\top \mathbf{w}\right) \cdot \prod_{i=1}^N \exp\left(-\frac{(y_i - \mathbf{w}\mathbf{x}_i)^2}{2\beta}\right) \quad (10)$$

Maximizing the log posterior function is equivalent to the ridge regression.

### 2.2.2 Lasso

Lasso addresses on over-fitting by penalizing the  $l_1$ -norm of weight vector  $\mathbf{w}$ ,

$$\min_{\mathbf{w}} \sum_{i=1}^m (\mathbf{w}\phi_i(\mathbf{x}) - y_i)^2 + \lambda \|\mathbf{w}\|_1$$

If we assume a Laplace prior for the weight vector,  $p(\mathbf{w}) = \frac{1}{2\alpha} \exp\left(-\frac{\|\mathbf{w}\|_1}{\alpha}\right)$ , then the posterior of the training sequence is:

$$p(\mathbf{w}|S) \propto p(\mathbf{w})p(S|\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|_1}{\alpha}\right) \cdot \prod_{i=1}^N \exp\left(-\frac{(y_i - \mathbf{w}\mathbf{x}_i)^2}{2\beta}\right) \quad (11)$$

Maximizing the log posterior function is equivalent to the Lasso model.

remark2:

