Chapter *4* 线性模型

Siheng Zhang
zhangsiheng@cvte.com

2021 年 6 月 4 日

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

-

目录

1 Linear classification

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

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

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 $\mathbf{w}^\top \mathbf{x}/\|\mathbf{w}\| = -w_0/\|\mathbf{w}\|$.

remark1. It is usually convenient to 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.

*remark2: 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 $K(K-1)/2$ binary discriminant functions, one for every pair of classes (but will lead to ambiguous region).*

1.1 The VC dimension of half-spaces

The class of linear function (Eq. 1) represents a hypothesis set *half-space*, usually denoted as HS_d. Its VC dimension is $VCdim(HS_d) = d + 1$.

remark3: proof. Firstly, we should show that any set of d points in $\mathcal{R}^d$ can be shattered by half-space.

Secondly, we should show there exists a point set of $d+1$ points in $\mathcal{R}^d$ that cannot be shattered by half-space. Denote the points as $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{d+1}$. There must be some $a_i, i = 1, 2, \cdots, d+1$ (not all of them are zero) which satisfy that $\sum_{i=1}^{d+1} a_i \mathbf{x}_i = 0$. 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.

It follows that we can learn half-space using the ERM paradigm with a sample size of $\Omega(\frac{d+\log(1/\delta)}{\epsilon})$. Before discussing the implementation, we would like to show that the bound of sample size is too loose, and hence loss its guiding meaning in practice. Suppose that we would like to be 'very sure' to the learned hypothesis, namely $\epsilon \to 0$ and $\delta \to 1$, then the sample size tends to be infinity. In agnostic case, since we cannot learn a perfect half space, naturally more training data is better, and the estimation of samples size is trivial. In realizable case, since

In the realizable (namely, separable) case, we can implement ERM paradigm as a linear programming problem. The normal form of a linear programming problem is:

$$\min_{\mathbf{w}} \mathbf{u}^\top \mathbf{w}$$

$$s.t. \mathbf{A}\mathbf{w} \geq \mathbf{v}$$

Besides, we can also use a Perceptron algorithm, which is left in the next chapter.

However, in the agnostic (namely, non-separable) case, the implementation is computational hard. And the most popular solution is to use *surrogate loss* functions but not the 0-1 loss in realizable case. .....

1.2 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 $\mathbf{w}$, 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

$$\mathsf{m}_1 = \frac{1}{N_1}\sum_{\mathsf{x}_n \in C_1} \mathsf{x}_n, \qquad \mathsf{m}_2 = \frac{1}{N_2}\sum_{\mathsf{x}_n \in C_2} \mathsf{x}_n$$

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 $\mathsf{w}^\top(\mathsf{m}_2 - \mathsf{m}_1) + \lambda(1 - \|\mathsf{w}\|_2)$, 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. 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 $\mathsf{S}_B = (\mathsf{m}_2 - \mathsf{m}_1)(\mathsf{m}_2 - \mathsf{m}_1)^\top, \mathsf{S}_W = \sum_{\mathsf{x}_n \in C_1}(\mathsf{x}_n - \mathsf{m}_1)(\mathsf{x}_n - \mathsf{m}_1)^\top + \sum_{\mathsf{x}_n \in C_2}(\mathsf{x}_n - \mathsf{m}_2)(\mathsf{x}_n - \mathsf{m}_2)^\top$. The optimizer is (the solution is left as exercise):

$$\mathsf{w} \propto \mathsf{S}_W^{-1}(\mathsf{m}_2 - \mathsf{m}_1)$$

remark4: Fisher's linear discriminant for the case of multi-class.

1.3 Least Squares for classification

1.4 Probabilistic perspective

1.4.1 Logistic regression

1.4.2 Iterative re-weighted least squares(IRLS)

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

$$\min_{\mathsf{w}} L_S(h) = \sum_{i=1}^{m} l(h(\mathsf{x}_i)) = \sum_{i=1}^{m}(h(\mathsf{x}_i) - y_i)^2 = \sum_{i=1}^{m}(\mathsf{w}^\top \mathsf{x}_i - y_i)^2 \tag{3}$$

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

$$\log \mathcal{L} = -\frac{m}{2}\log 2\pi\beta - \sum_{i=1}^{m}\frac{(y_i - \mathsf{w}^\top \mathsf{x}_i)^2}{2\beta} \tag{4}$$

Obviously, MLE is equivalent to linear regression.

remark5. 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

$$h(\mathbf{x}) = w_0 + \sum_{j=1}^{d} w_j \phi_j(x) \tag{5}$$

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

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

Setting it to zero gives

$$\mathbf{w} = \Phi^\dagger \mathbf{y} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y} \tag{6}$$

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

remark6: 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.

remark7: 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

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

in which lr is the learning rate.

remark8: Basis function examples. 1. **Polynomial basis**, 2. **Radical basis**, 3. **Fourier basis**.

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

In closed-form solution (Eq. 6), if $n \leq d$, the SSE loss can achieve zero. It means that the model capacity is enough to 'memorize' all training examples. But it may suffer from **over-fitting**.

Consider the expected loss with squared error,

$$\begin{aligned}
\mathbb{E}(l) &= \int\int l(h(\mathbf{x}))p(\mathbf{x}, y)\mathrm{d}\mathbf{x}\mathrm{d}y \\
&= \int\int (h(\mathbf{x}) - y)^2 p(\mathbf{x}, y)\mathrm{d}\mathbf{x}\mathrm{d}y \quad \text{Note that, setting its derivatives to zero leads to } h(\mathbf{x}) = \int yp(\mathbf{x}, y)\mathrm{d}y/p(\mathbf{x}) = \mathbb{E}(y|\mathbf{x}) \\
&= \int\int \left\{ h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x}) + \mathbb{E}(y|\mathbf{x}) - y \right\}^2 p(\mathbf{x}, y)\mathrm{d}\mathbf{x}\mathrm{d}y \\
&= \int\int \left\{ \left[h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x}) \right]^2 + \left[\mathbb{E}(y|\mathbf{x}) - y \right]^2 + 2 \left[h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x}) \right] \left[\mathbb{E}(y|\mathbf{x}) - y \right] \right\} p(\mathbf{x}, y)\mathrm{d}\mathbf{x}\mathrm{d}y \\
&= \int \left\{ h(\mathbf{x}) - \mathbb{E}(y|\mathbf{x}) \right\}^2 p(\mathbf{x})\mathrm{d}\mathbf{x} + \int\int \left\{ \mathbb{E}(y|\mathbf{x}) - y \right\}^2 p(\mathbf{x}, y)\mathrm{d}\mathbf{x}\mathrm{d}y
\end{aligned} \tag{7}$$

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. The first term depends on our choice for the function $h(\mathbf{x})$. Because it is non-negative, its optimal value 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.

Consider the integrand of the first term. For a particular data set $\mathcal{D}$, the expectation is not depend on the data set, but the function $h(\cdot)$ does. So it takes the form

$$
\begin{aligned}
&\{h(\mathbf{x};\mathcal{D}) - \mathbb{E}(y|\mathbf{x})\}^2 \\
&= \{h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D})) + \mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D})) - \mathbb{E}(y|\mathbf{x})\}^2 \\
&= \{h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D}))\}^2 + \{\mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D})) - \mathbb{E}(y|\mathbf{x})\}^2 + 2\{h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D}))\}\{\mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D})) - \mathbb{E}(y|\mathbf{x})\}
\end{aligned}
$$

Now take expectation of it with respect to $\mathcal{D}$, the third term will vanish, giving

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[\{h(\mathbf{x};\mathcal{D}) - \mathbb{E}(y|\mathbf{x})\}^2\right] &= \mathbb{E}_{\mathcal{D}}\left[\{\mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D})) - \mathbb{E}(y|\mathbf{x})\}^2\right] + \mathbb{E}_{\mathcal{D}}\left[\{h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D}))\}^2\right] \\
&= \underbrace{\{\mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D})) - \mathbb{E}(y|\mathbf{x})\}^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{h(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}(h(\mathbf{x};\mathcal{D}))\}^2\right]}_{\text{variance}}
\end{aligned} \tag{8}
$$

The first term, called the squared bias, represents the extent to which the average prediction over all data sets differs from the desired regression function. The second term, called the variance, measures the extent to which the solutions for individual data sets vary around their average, and hence this measures the extent to which the regressor is sensitive to the particular choice of data set.

As discussed before, there is a trade-off between bias and variance, with very flexible models having low bias and high variance, and relatively rigid models having high bias and low variance. 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:

$$
p(\mathbf{w}|S) \propto p(\mathbf{w})p(S|\mathbf{w}) \propto \exp\left(-\frac{\alpha}{2}\mathbf{w}^\top\mathbf{w}\right) \cdot \prod_{i=1}^{N} \exp\left(-\frac{(y_i - \mathbf{w}\mathbf{x}_i)^2}{2\beta}\right) \tag{9}
$$

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 $\mathbf{w}$,

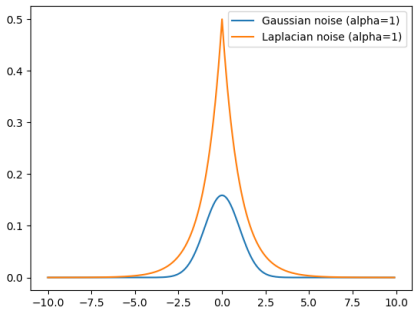
$$
\min_{\mathbf{w}} \sum_{i=1}^{m} (\mathbf{w}\phi_i(\mathbf{x}) - y_i)^2 + \lambda \|\mathbf{w}\|_1
$$

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:

$$
p(\mathbf{w}|S) \propto p(\mathbf{w})p(S|\mathbf{w}) \propto \exp\left(-\frac{\|\mathbf{w}\|_1}{\alpha}\right) \cdot \prod_{i=1}^{N} \exp\left(-\frac{(y_i - \mathbf{w}\mathbf{x}_i)^2}{2\beta}\right) \tag{10}
$$

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

remark5: Below is the comparison of Laplace distribution and Gaussian distribution with $\alpha = 1$.



2.3 Predictive distribution

2.4 Equivalent kernel