Chapter *4* Linear Model

Siheng Zhang
zhangsiheng@cvte.com

2022 年 8 月 29 日

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

-

目录

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

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

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

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

Obviously, MLE is equivalent to linear regression.

remark8. *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.*

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

$$h(\mathsf{x}) = w_0 + \sum_{j=1}^{d} w_j \phi_j(x) \tag{2}$$

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

Least squares also gives the result by $\mathsf{w} = \Phi^\dagger \mathsf{y} = (\Phi^\top\Phi)^{-1}\Phi^\top \mathsf{y}$. Φ is an $n * d$ matrix, whose elements are given by $\Phi_{nj} = \phi_j(\mathsf{x}_n)$.

remark9: multiple-outputs. *The solution to multiple-outputs regression problem decouples between the different target variables so we do not need to discuss it here.*
remark10: 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* $\mathsf{w}^{t+1} = \mathsf{w}^t - lr * \nabla L_{(\mathsf{x}_t, y_t)}(h) = \mathsf{w}^t + lr * (y_i - (\mathsf{w}^t)^\top \phi(\mathsf{x}_t))\phi(\mathsf{x}_t)$*, in which* lr *is the learning rate.*
remark11: Basis function examples. 1. Polynomial basis, 2. Radical basis, 3. Fourier basis.

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

In closed-form solution, if $n \le d$, the SSE loss can achieve zero. It means that the model is able to 'memorize' all training examples, but the parameters vary significantly if the data points changes (even just a bit). So if there exists noisy data point, the model fits the noise well (zero training error) and will show bad performance to unseen data. This phenomenon is called **over-fitting**. Formally, consider the expected loss with squared error,

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

The second term, which is independent of $h(\mathsf{x})$, 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 $h(\mathsf{x})$. 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 $h(\mathsf{x})$ to any desired degree of accuracy. But indeed, we can just estimate $h(\mathsf{x})$ from a particular data set $\mathcal{D}$, so we turn to study the following:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}\left[\{h(\mathsf{x};\mathcal{D})-\mathbb{E}(y|\mathsf{x})\}^2\right] &= \mathbb{E}_{\mathcal{D}}\left[\{h(\mathsf{x};\mathcal{D})-\mathbb{E}_{\mathcal{D}}(h(\mathsf{x};\mathcal{D}))+\mathbb{E}_{\mathcal{D}}(h(\mathsf{x};\mathcal{D}))-\mathbb{E}(y|\mathsf{x})\}^2\right] \\
&= \mathbb{E}_{\mathcal{D}}\left[\{\mathbb{E}_{\mathcal{D}}(h(\mathsf{x};\mathcal{D}))-\mathbb{E}(y|\mathsf{x})\}^2\right]+\mathbb{E}_{\mathcal{D}}\left[\{h(\mathsf{x};\mathcal{D})-\mathbb{E}_{\mathcal{D}}(h(\mathsf{x};\mathcal{D}))\}^2\right] \\
&= \underbrace{\{\mathbb{E}_{\mathcal{D}}(h(\mathsf{x};\mathcal{D}))-\mathbb{E}(y|\mathsf{x})\}^2}_{\text{bias}^2}+\underbrace{\mathbb{E}_{\mathcal{D}}\left[\{h(\mathsf{x};\mathcal{D})-\mathbb{E}_{\mathcal{D}}(h(\mathsf{x};\mathcal{D}))\}^2\right]}_{\text{variance}}
\end{aligned} \tag{3}$$

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: 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 w, and view the learning procedure to maximizing its posterior. Two of the most popular case is discussed below.

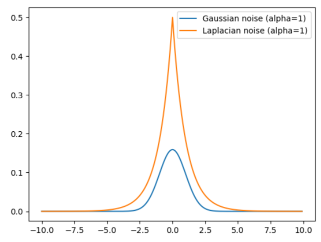
2.0.1 Ridge regression

Ridge regression addresses on over-fitting by penalizing the l_2-norm of weight vector,

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

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

$$p(\mathsf{w}|S) \propto p(\mathsf{w})p(S|\mathsf{w}) \propto \exp\left(-\frac{\alpha}{2}\mathsf{w}^\top\mathsf{w}\right)\cdot\prod_{i=1}^{N}\exp\left(-\frac{(y_i-\mathsf{w}\mathsf{x}_i)^2}{2\beta}\right) \tag{4}$$



Laplace distribution and Gaussian distribution with $\alpha = 1$.

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

2.0.2 Lasso

Lasso addresses on over-fitting by penalizing the l_1-norm of weight vector,

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

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

$$p(\mathsf{w}|S) \propto p(\mathsf{w})p(S|\mathsf{w}) \propto \exp\left(-\frac{\|\mathsf{w}\|_1}{\alpha}\right)\cdot\prod_{i=1}^{N}\exp\left(-\frac{(y_i-\mathsf{w}\mathsf{x}_i)^2}{2\beta}\right) \tag{5}$$

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

3 Predictive distribution

4 Equivalent kernel