Machine Learning Working Notes

Huang Xiao[*]
Department of Computer Science
Technical University of Munich
Munich, Germany 85748

Abstract

Machine learning is a fast pacing discipline in many working fields, especially it is now regarded as the most impacting subject in artificial intelligence. In this working notes, I summarize some important notes during my study of machine learning. For the completeness, references are included for readers who are reading this article. Note that this working note is only distributed and shared with author's acknowledge and confirmation. It is not intended as a publishable research paper or tutorial.

1 Linear Models

1.1 Regression

It seems everything starts to grow from linear model, whatever regression or classification, linear models expand to almost many other learning models we face during the research. So starting from a very simple linear regression problem, given training set $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^n$ with n sample, where $\boldsymbol{y}$ are the responses as numerical values. A linear regression model finds a linear weight vector, which minimizes a certain type of empirical error. This is obviously defined in perspective of statistical learning theory.

$$\arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \|w^T x_i + b - y_i\|^2 \tag{1}$$

where empirical error introduced by individual sample is equally weighted by $1/n$. It is seen that we are using a straight line to fit a possibly any shaped function, obviously the summation or mean error can be large due to the noise or intrinsic nonlinearity of function. However, a typical misunderstanding of linear model for beginners is that the term linear refers to $\boldsymbol{w}$ instead of $\boldsymbol{x}$. That is to say, we are expecting a linear model on parameters $\boldsymbol{w}$, but the feature vectors $\boldsymbol{x}$ can actually be any shape. Therefore, in literature we mostly see a feature mapping of input $\boldsymbol{x}$ as $\phi(\boldsymbol{x})$, and it does not break the linear property of the model. Therefore, we have our liner model as,

$$\arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \|w^T \phi(x_i) + b - y_i\|^2 \tag{2}$$

A typical feature mapping is polynomial feature mapping. Suppose we have a 2-dimensional input data sample (x_1, x_2), we define a feature mapping as follows,

$$(x_1, x_2) \to \left(x_1, x_2, x_1^2 + x_2^2\right)$$

, where a 2-d plane is transformed as a paraboloid in 3-d space. Substituting back into previous linear function, it becomes a polynomial line fitting problem, but it is still linear in $\boldsymbol{w}$.

[*]Visiting homepage: feuerchop.github.io

To solve the least square problem defined in Eq. 2 to obtain an optimal parameter estimation $\boldsymbol{w}$, we take the gradient with respect to the loss and set it to zero. Note that the intercept b can be folded in vector $\boldsymbol{w}$ by adding additional entry 1 in the end, for simplicity, we ignore it in our formulation. The least square solution is,

$$\boldsymbol{w}^* = \left(\Phi(\boldsymbol{X})^T \Phi(\boldsymbol{X})\right)^{-1} \Phi(\boldsymbol{X})^T \boldsymbol{Y} \tag{3}$$

As long as the $\Phi(\boldsymbol{X})^T \Phi(\boldsymbol{X})$ is not singular, there exists analytical solution. We will call $\Phi(\boldsymbol{X})$ design matrix, which takes each row as a feature mapping on $\boldsymbol{x}$. We will see in short that the inner product of feature mapping can be explicitly defined by a certain kernel function, which established a very important chapter of learning theory, *i.e.Kernel Methods*. To predict the response for a new sample $\boldsymbol{x}^*$, we have,

$$\boldsymbol{y}^* = \phi(\boldsymbol{x}^*) \left(\Phi(\boldsymbol{X})^T \Phi(\boldsymbol{X})\right)^{-1} \Phi(\boldsymbol{X})^T \boldsymbol{Y}$$

From Probabilistic View

Different from minimizing empirical errors from observations, we can examine the whole problem in a probabilistic view, that is, minimize the uncertainty from observations. If we reformulate the problem as a summation of a deterministic function and a indeterministic noise from a certain probabilistic distribution, we can write the linear model as,

$$\boldsymbol{y} = \boldsymbol{w}^T \phi(\boldsymbol{x}) + \epsilon$$

, where $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$ which is defined as a Gaussian noise, the bias term b is again folded in $\boldsymbol{w}$. Moreover, we can get the response $\boldsymbol{Y}$ as a Gaussian distribution as well.

$$\boldsymbol{Y} \sim \prod_{i=1}^{n} \mathcal{N}\left(\boldsymbol{w}^T \phi(\boldsymbol{x}_i), \sigma^2\right)$$

In order to get the optimal parameter $\boldsymbol{w}$, we need to maximize the likelihood $p\left(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{w}\right)$. It equals to maximize the log-likelihood, and the log-likelihood gives,

$$\ln p\left(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{w}\right) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{2\sigma^2}\|\boldsymbol{Y} - \Phi(\boldsymbol{X})\boldsymbol{w}\|^2 \tag{4}$$

From Eq.4, we can see maximizing the log-likelihood (*MLE*) equals minimizing the least squared error. We will get the exact same solution as in Eq.3.

Overfitting

Now suppose we have four 1-dimensional observations $\boldsymbol{X}$, and define an arbitrary feature mapping ϕ, we expect to find a linear model to minimize the least squared error, as we see in (2).

$$\begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} = \phi\left(\begin{bmatrix} 1.5 \\ 0.5 \\ 2.5 \end{bmatrix}\right)\boldsymbol{w}$$

If we define the feature mapping ϕ as

$$\begin{bmatrix} 1.5 \\ 0.5 \\ 2.5 \end{bmatrix} \xrightarrow{\phi} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We can see that $\boldsymbol{w}$ is exactly the response vector $\boldsymbol{y}$, and the least squared error is minimized as zero. A feature mapping can always be defined to achieve zero error, if there's no constraints at all. But obviously, there is no benefit to use this linear model on any prediction task, and mostly likely we will get a very high prediction error based on that. If the model performs well on training dataset, but poorly on unseen data, we can call this situation as overfitting. And certainly, overfitting is a central problem that machine learning attempts to solve.

To avoid overfitting, we can firstly introduce constraint by adding a penalty term on the complexity of the $\boldsymbol{w}$, which is known as *regularization*. For example, by penalizing a 2-norm of $\boldsymbol{w}$, we can generalize the least squared error problem as,

$$\arg\min_{\boldsymbol{w}} \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{w}^T\phi(x_i) + b - y_i\|^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{5}$$

Similarly taking the derivative w.r.t. $\boldsymbol{w}$ and set to zero, we can get the optimal parameters as,

$$\boldsymbol{w}^* = \left(\Phi(\boldsymbol{X})^T\Phi(\boldsymbol{X}) + \lambda\boldsymbol{I}\right)^{-1}\Phi(\boldsymbol{X})^T\boldsymbol{Y} \tag{6}$$

Again from probabilistic view, we introduce uncertainty on $\boldsymbol{w}$ instead of only considering uncertainty on response $\boldsymbol{y}$. Define a prior on $\boldsymbol{w}$ following a D-dimensional Gaussian distribution,

$$\boldsymbol{w} \sim \mathcal{N}\left(\boldsymbol{0}, \Sigma_{\boldsymbol{w}}\right)$$

We want to capture the posterior distribution on $\boldsymbol{w}$ after observations of $(\boldsymbol{X}, \boldsymbol{Y})$, that is, the objective is to maximize the posterior according to Bayes theorem,

$$\max p\left(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{Y}\right) = \frac{p\left(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{w}\right)p\left(\boldsymbol{w}\right)}{\int p\left(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{w}\right)p\left(\boldsymbol{w}\right)d\boldsymbol{w}} \tag{7}$$

The denominator in Eq.7 is also called marginal likelihood, which is independent of $\boldsymbol{w}$, therefore, taking the logarithm of the posterior, we have,

$$\ln p\left(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{Y}\right) = -\frac{(n+d)}{2}\ln(2\pi) - n\ln(\Sigma_{\boldsymbol{w}}) - \frac{1}{2}\ln|\Sigma_{\boldsymbol{w}}| - \frac{1}{\sigma^2}\|\boldsymbol{Y} - \Phi(\boldsymbol{X})\boldsymbol{w}\|^2 - \frac{1}{2}\boldsymbol{w}^T\Sigma_{\boldsymbol{w}}^{-1}\boldsymbol{w} \tag{8}$$

Take the gradient w.r.t. $\boldsymbol{w}$ and set to zero, we can get very similar results as in Eq.6.

$$\boldsymbol{w}_{map} = \left(\Phi(\boldsymbol{X})^T\Phi(\boldsymbol{X}) + \sigma^2\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\right)^{-1}\Phi(\boldsymbol{X})^T\boldsymbol{Y} \tag{9}$$

If the prior is defined with an isotropic Gaussian, we see that the *MAP* solution is equivalent to ℓ_2 regularization form. Now let us look back at the posterior, note that,

$$\begin{aligned}
p\left(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{Y}\right) &\propto p\left(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{w}\right)p\left(\boldsymbol{w}\right) \\
&\propto \exp\left[-\frac{1}{2\sigma^2}\left(\Phi(\boldsymbol{X})\boldsymbol{w} - \boldsymbol{Y}\right)^T\left(\Phi(\boldsymbol{X})\boldsymbol{w} - \boldsymbol{Y}\right)\right]\exp\left(\frac{1}{2}\boldsymbol{w}^T\Sigma_{\boldsymbol{w}}^{-1}\boldsymbol{w}\right) \\
&\propto \exp\left\{-\frac{1}{2}\boldsymbol{w}^T\left(\frac{1}{\sigma^2}\Phi(\boldsymbol{X})^T\Phi(\boldsymbol{X}) + \Sigma_{\boldsymbol{w}}^{-1}\right)\boldsymbol{w} + \frac{1}{\sigma^2}\boldsymbol{Y}^T\Phi(\boldsymbol{X})\boldsymbol{w}\right\}
\end{aligned} \tag{10}$$

By completing the square we can get the mean and covariance of the posterior, that is,

$$\begin{aligned}
\boldsymbol{w}^* &\sim \mathcal{N}\left(\boldsymbol{m}^*, \boldsymbol{A}^{-1}\right) \\
\boldsymbol{m}^* &= \frac{1}{\sigma^2}\boldsymbol{A}^{-1}\Phi(\boldsymbol{X})^T\boldsymbol{Y} && (11) \\
\boldsymbol{A} &= \frac{1}{\sigma^2}\Phi(\boldsymbol{X})^T\Phi(\boldsymbol{X}) + \Sigma_{\boldsymbol{w}}^{-1} && (12)
\end{aligned}$$

And we see that the *MAP* solution is exactly the same as the mode of the posterior. To predict a new input sample $\boldsymbol{x}^*$, we can derive a predictive distribution instead of just a single value at the mode, and it is again a Gaussian with posterior mean multiplied with the test sample, and variance of the predictive distribution is the quadratic form on the posterior covariance, which grows with the magnitude of test samples.

$$\begin{aligned}
p(\boldsymbol{y}^* \mid \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{x}^*) &= \int p(\boldsymbol{y}^* \mid \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{w}, \boldsymbol{x}^*)p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{Y})d\boldsymbol{w} \\
&\sim \mathcal{N}\left(\boldsymbol{\Phi}(\boldsymbol{x}^*)^T\boldsymbol{m}^*, \Phi(\boldsymbol{x}^*)^T\boldsymbol{A}^{-1}\Phi(\boldsymbol{x}^*)\right)
\end{aligned} \tag{13}$$

1.2 Classification

Now let us consider classification problem, where we expect a functional $\pi(\boldsymbol{x})$ to map input $\boldsymbol{x}$ to class labels, in binary case $\boldsymbol{y} = (+1, -1)$. More commonly, we can define a Bernoulli distribution $p(y = +1 \mid x)$ for one class and $1 - p\left(y = +1 \mid x\right)$ for another. Typically, we would choose a sigmoid function, *e.g.*logistic function or *tanh* to warp a possibly infinite value into a bounding box, *e.g.*, $[0, 1]$ for logistic function. This is a desired behavior, since any function value will be transformed to a probability. A logistic function is defined,

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

Obviously, we also have,

$$\begin{aligned}\sigma(-a) &= 1 - \sigma(a) \\ \tanh(a) &= 2\sigma(2a) - 1 \\ \frac{\partial \sigma(a)}{\partial a} &= (1 - \sigma(a))\,\sigma(a)\end{aligned}$$

Thus, the conditional class distribution given input dataset can be defined as a sigmoid function on the linear model $p\left(y \mid \boldsymbol{x}\right) = \sigma(yf(\boldsymbol{x}))$, again we fold the bias term b in the parameters $\boldsymbol{w}$.

2 Gaussian Process

2.1 Regression

Gaussian process is an important nonparametric regression model which looks for an optimal functional in a space of functions, that minimizes a loss function, although the loss function needs not to be explicitly defined. Rasmussen (2006)

Give a training dataset $\mathcal{D} = \{x_i\}_{i=1}^n$ with $x_i \in R^d$, *i.i.d* drawn from certain distribution, we are interested at the predictive distribution of unknown target for the test sample x_*, denoted as f_*. Suppose a prior over $\boldsymbol{y}$ given input $\boldsymbol{X}$ is a n-variable Gaussian distribution,

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, K(\boldsymbol{X}, \boldsymbol{X}))$$

where $K(\boldsymbol{X}, \boldsymbol{X})$ defines a covariance function over $\boldsymbol{X}$. Therefore the posterior of f_* given the training dataset $\mathcal{D}$ is also a Gaussian.

$$f_* | \boldsymbol{y}, \boldsymbol{X}, x_* \sim \mathcal{N}(\mu_*, \Sigma_*^{-1})$$

where the sufficient statistics can be derived using Bayesian theorem,

$$\bar{f}_* = \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 I)^{-1} \boldsymbol{y} \tag{14}$$

$$\mathbb{V}(f_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 I)^{-1} \boldsymbol{k}_* \tag{15}$$

where σ_n^2 is the noise level, $\boldsymbol{K}$ represents a shorthand for covariance matrix on input $\boldsymbol{X}$, and we denote $\boldsymbol{k}_*$ as the kernel function $k(\boldsymbol{X}, \boldsymbol{x}_*)$ for simplicity.

To obtain the Eq.(14)-(15), we can use the following trick. Given two variables $(\boldsymbol{x}, \boldsymbol{y})$ following a Gaussian distribution,

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \right)$$

Then we have the conditional distribution,

$$\boldsymbol{x} | \boldsymbol{y} \sim \mathcal{N}\left(\mu_x + CB^{-1}(\boldsymbol{y} - \mu_y), A - C^T B^{-1} C\right)$$

Now looking at the predictive mean of training dataset, which can be given by Eq.(14)-(15) on training set $\boldsymbol{X}$ itself,

$$\bar{f} = \boldsymbol{K}(\boldsymbol{K} + \sigma_n^2 I)^{-1} \boldsymbol{y}$$

Since $\boldsymbol{K}$ is symmetric positive definite and its eigendecomposition is $\boldsymbol{K} = \sum_{i=1}^n \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T$, where λ_i is the *i*th eigenvalue and $\boldsymbol{u}_i$ is the *i*th eigenvector. Now define a vector $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$, therefore we have $\boldsymbol{K} = \boldsymbol{U} \Sigma \boldsymbol{U}^T$. According the the matrix inverse lemma [Tylavsky & Sohie (1986)], we can derive a simple form for the predictive mean, where we observe that the predictive mean is a linear smooth on their targets.

$$
\begin{aligned}
\bar{f} &= \boldsymbol{K}\left(\sigma_n^{-2}I - \sigma_n^{-2}I\boldsymbol{U}\left(\Sigma^{-1} + \boldsymbol{U}^T\sigma_n^{-2}I\boldsymbol{U}\right)^{-1}\boldsymbol{U}^T\sigma_n^{-2}I\right)\boldsymbol{y} \\
&= \boldsymbol{K}\left(\sigma_n^{-2}I - \sigma_n^{-2}\boldsymbol{U}\left(\Sigma^{-1} + \sigma_n^{-2}I\right)^{-1}\boldsymbol{U}^T\sigma_n^{-2}\right)\boldsymbol{y} \\
&= \left(\sigma_n^{-2}\boldsymbol{U}\Sigma\boldsymbol{U}^T - \boldsymbol{U}\Sigma\begin{bmatrix}\frac{\lambda_1\sigma_n^{-2}}{\lambda_1+\sigma_n^2} & & \\ & \ddots & \\ & & \frac{\lambda_n\sigma_n^{-2}}{\lambda_n+\sigma_n^2}\end{bmatrix}\boldsymbol{U}^T\right)\boldsymbol{y} \\
&= \left(\sigma_n^{-2}\boldsymbol{U}\Sigma\boldsymbol{U}^T - \sigma_n^{-2}\boldsymbol{U}\begin{bmatrix}\frac{\lambda_1^2}{\lambda_1+\sigma_n^2} & & \\ & \ddots & \\ & & \frac{\lambda_n^2}{\lambda_n+\sigma_n^2}\end{bmatrix}\boldsymbol{U}^T\right)\boldsymbol{y} \\
&= \left(\sigma_n^{-2}\sum_{i=1}^n \lambda_i\boldsymbol{u}_i\boldsymbol{u}_i^T - \sigma_n^{-2}\sum_{i=1}^n\frac{\lambda_i^2}{\lambda_i+\sigma_n^2}\boldsymbol{u}_i\boldsymbol{u}_i^T\right)\boldsymbol{y} \\
&= \sum_{i=1}^n \frac{\lambda_i}{\lambda_i+\sigma_n^2}\boldsymbol{u}_i\boldsymbol{u}_i^T\boldsymbol{y} \\
&= \sum_{i=1}^n \frac{\gamma_i\lambda_i}{\lambda_i+\sigma_n^2}\boldsymbol{u}_i, \quad \text{with } \gamma_i = \boldsymbol{u}_i^T\boldsymbol{y}
\end{aligned}
$$

2.2 Classification

Using Gaussian process for classification task is a bit more complicated than regression. The main idea of it is to use a 'squash' function to convert a predicted function value within $[0, 1]$, *e.g.*, sigmoid function, cumulative Gaussian *pdf*. Given a dataset $\mathcal{D} = \{x_i\}_{i=1}^n$ with $x_i \in R^d$ and corresponding labels $\boldsymbol{y} = [+1, -1]$

3 Support Vector Machines

4 PCA

The basic idea of PCA is to maximally reduce information loss of projecting high dimensional data to lower dimension. Therefore, an intuitive consideration would be that we introduce first of all a projection matrix $\boldsymbol{u}$ on d-dimensional instance x, so that x is mapped on a lower m-dimensional space. Following column vector routine, we expect that matrix $\boldsymbol{u}$ as being $m \times d$. Now given a dataset $\boldsymbol{X} = \{x_i\}_{i=1}^n$, it will be projected on a m-dimensional space by $\boldsymbol{u}$. The objective of the projection is to maximize the covariance of data on the lower dimensional space, that is,

$$
\max \frac{1}{n}\sum_{i=1}^n \|\boldsymbol{u}x - \boldsymbol{u}\bar{x}\|^2
$$

that can be rewritten as,

$$
\begin{aligned}
&\underset{\boldsymbol{u}}{\textit{maximize}} \quad \boldsymbol{u}S\boldsymbol{u}^T \\
&\text{s.t.} \quad \boldsymbol{u}_i\boldsymbol{u}_i^T = 1,\ i = 1, \ldots, m
\end{aligned} \tag{16}
$$

where S is the covariance of $\boldsymbol{X}$. To prevent $\boldsymbol{u}$ goes to infinity, we assume $\boldsymbol{u}$ has unit length, namely, $\boldsymbol{u}$ represents a set of basis of the lower dimension.

According to (16), we introduce m Lagrangian multipliers $\boldsymbol{\lambda}$ as a diagonal matrix, and we have

$$
L = \underset{\boldsymbol{u}}{\textit{maximize}} \quad \boldsymbol{u}S\boldsymbol{u}^T - \boldsymbol{\lambda}\boldsymbol{u}\boldsymbol{u}^T
$$

Take the derivative of L with respect to $\boldsymbol{u}$, we have

$$\begin{aligned} \boldsymbol{u}S &= \boldsymbol{\lambda u} \\ S &= \boldsymbol{u}^{-1}\lambda\boldsymbol{u} \end{aligned} \tag{17}$$

Since $\boldsymbol{u}$ is orthogonal, it is therefore not singular. We see that $\boldsymbol{u}$ and $\boldsymbol{l}ambda$ are the indeed the eigenvectors and eigenvalues for S respectively. And Eq.(17) is exactly the singular value decomposition of $S = U\Sigma V^T$, we can derive $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ from S conveniently.

References

Rasmussen, Carl Edward. Gaussian processes for machine learning. MIT Press, 2006.

Tylavsky, Daniel J. and Sohie, Guy R L. Generalization of the matrix inversion lemma. *Proceedings of the IEEE*, 74(7):1050–1052, 7 1986. ISSN 0018-9219.