A Note on Overfitting in Regression and Classification

Ying Nian Wu, UCLA Statistics

Background materials for STATS courses

Contents

1 Regression and classification

1.1 Data and model

The dataset of regression or classification consists of an $n \times p$ matrix $\mathbf{X} = (x_{ij})$, and a $n \times 1$ vector $\mathbf{Y} = (y_i)$. In regression, y_i is continuous. The linear model is of the following form:

$$y_i \approx \sum_{j=1}^{p} x_{ij}\beta_j.$$

In classification, y_i is binary. The linear model is of the following form

$$y_i \approx \text{sign}(\sum_{j=1}^{p} x_{ij}\beta_j).$$

Here we are deliberately ambiguous about the intercept term. If $x_{i1} = 1$ for all i, then β_1 will be the the intercept term. $[\mathbf{X}, \mathbf{Y}]$ is called the training data. y_i is called response variable, outcome, dependent variable. x_{ij} is called predictor, regressor, covariate, independent variable. In the experimental design setting, $\mathbf{X}$ is called the design matrix.

obs	$\mathbf{X}_{n\times p}$	$\mathbf{Y}_{n\times 1}$
1	$x_{11}, x_{12}, ..., x_{1p}$	y_1
2	$x_{21}, x_{22}, ..., x_{2p}$	y_2
...		
n	$x_{n1}, x_{n2}, ..., x_{np}$	y_n

obs	$\mathbf{X}_{n\times p}$	$\mathbf{Y}_{n\times 1}$
1	$X_1^\top$	y_1
2	$X_2^\top$	y_2
...		
n	$X_n^\top$	y_n

We can arrange the data in terms of $X_i^\top = (x_{ij}, j = 1, ..., p)$, where $X_i^\top$ is the i-th row of $\mathbf{X}$. Here X_i is not in bold font. We can write the linear score as $X_i^\top \beta$, where $\beta = (\beta_j, j = 1, ..., p)^\top$.

obs	$\mathbf{X}_{n\times p}$	$\mathbf{Y}_{n\times 1}$
1		
2	$\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p$	$\mathbf{Y}$
...		
n		

We can also arrange the data in terms of $\mathbf{X}_j = (x_{ij}, i = 1, ..., n)^\top$, where $\mathbf{X}_j$ is the j-th column of $\mathbf{X}$. Here $\mathbf{X}_j$ is in bold font. We can write the linear regression model as $\mathbf{Y} \approx \sum_{j=1}^p \mathbf{X}_j \beta_j$, and the linear classification model as $\mathbf{Y} \approx \text{sign}(\sum_{j=1}^p \mathbf{X}_j \beta_j)$, where the sign function is applied element-wise.

1.2 Learning

The process of estimating β is called learning from the training data. The purpose is two-fold.

(1) Explanation: understanding the relationship between y_i and $(x_{ij}, j = 1, ..., p)$.

(2) Prediction: learn to predict y_i based on $(x_{ij}, j = 1, ..., p)$, so that in the testing stage, if we are given the predictor variables, we should be able to predict the outcome.

2 Overfitting in regression

2.1 Least squares projection

We can write the linear regression model as $\mathbf{Y} = \mathbf{X}^\top \beta + \epsilon$. The least squares estimate of β is

$$\hat{\beta} = \arg\min_\beta \|\mathbf{Y} - \mathbf{X}\beta\|^2 = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

If $\mathbf{X}^\top \mathbf{X} = I_p$, i.e., the p column vectors of $\mathbf{X}$ are orthogonal, then $\hat{\beta} = \mathbf{X}^\top \mathbf{Y}$, i.e., $\hat{\beta}_j = \langle \mathbf{Y}, \mathbf{X}_j \rangle$.

2.2 When Y is a noise vector

If the true value of β is 0, so that $\mathbf{Y} = \epsilon$. Let us assume $\epsilon \sim \mathrm{N}(0, I_n)$, where I_n is the identity matrix of dimension n. Thus $\mathrm{E}[\|\epsilon\|^2] = n$. Let us also assume $\mathbf{X}^\top \mathbf{X} = I_p$ (we will take care of the scaling issue later).

The least squares estimate $\hat{\beta} = \mathbf{X}^\top \epsilon = \delta \sim \mathrm{N}(0, I_p)$. Thus $\mathrm{E}[\|\delta\|^2] = p$. δ is the projected coordinates of the noise vector ϵ onto the space spanned by $\mathbf{X}$. The fitted value or the projected vector is $\hat{\epsilon} = \mathbf{X}\hat{\beta} = \mathbf{X}\delta$.

The training error is

$$\mathrm{E}[\|\epsilon - \hat{\epsilon}\|^2] = \mathrm{E}[\|\epsilon\|^2 - \|\hat{\epsilon}\|^2],$$

because of the Pythagarean theorem. We know $\mathrm{E}[\|\epsilon\|^2] = n$, and $\mathrm{E}[\|\hat{\epsilon}\|^2] = \mathrm{E}[\|\delta\|^2] = p$. Thus the training error is $n - p$.

Now consider the testing error. Let us assume that $\mathbf{X}$ is fixed, but the response vector in testing is an independent noise vector $\tilde{\epsilon} \sim \mathrm{N}(0, I_n)$, and $\mathrm{Cov}(\epsilon, \tilde{\epsilon}) = 0$. Then the testing error is

$$\mathrm{E}[\|\tilde{\epsilon} - \hat{\epsilon}\|^2] = \mathrm{E}[\|\tilde{\epsilon}\|^2 + \|\hat{\epsilon}\|^2 - 2\langle\tilde{\epsilon}, \hat{\epsilon}\rangle].$$

Since $\mathrm{E}[\langle\tilde{\epsilon}, \epsilon\rangle] = 0$, the testing error is $n + p$.

The overfitting is testing error - training error, which is $2p$.

Now consider a general estimator $\hat{\beta}$ (not necessarily least squares) obtained by fitting $(\mathbf{X}, \epsilon)$. Let $\hat{\epsilon} = \mathbf{X}\hat{\beta}$ be the fitted vector of this general estimator. Then we don't have Pythagarean this time. But we can write the training error as

$$\mathrm{E}[\|\epsilon - \hat{\epsilon}\|^2] = \mathrm{E}[\|\epsilon\|^2 + \|\hat{\epsilon}\|^2 - 2\langle\epsilon, \hat{\epsilon}\rangle].$$

The difference between the testing error and training error is $2\mathrm{E}[\langle\epsilon, \hat{\epsilon}\rangle]$, which is the overfitting.

The point is that you may be able to explain noises by overfitting the data in training, you will not be able to do so in testing.

2.3 When Y has both signal and noise

In real life, $y_i = f(x_{i1}, ..., x_{ip}) + \epsilon_i$. We can write $\mathbf{Y} = f + \epsilon$, where f is a $n \times 1$ vector with $f_i = f(x_{i1}, ..., x_{ip})$.

The least squares estimator

$$\hat{\beta} = \mathbf{X}^\top \mathbf{Y} = \mathbf{X}^\top (f + \epsilon) = \mathbf{X}^\top f + \mathbf{X}^\top \epsilon = \beta_{\text{best}} + \delta,$$

where $\beta_{\text{best}} = \mathbf{X}^\top f$ is the best value of β for explaining the signal. It is not observed or known. Again $\delta = \mathbf{X}^\top \epsilon$, and $\hat{\epsilon} = \mathbf{X}\delta$ is the noise absorbed by the model.

The training error is

$$\begin{aligned}
\mathrm{E}[\|\mathbf{Y} - \mathbf{X}\hat{\beta}\|^2] &= \mathrm{E}[\|(f + \epsilon) - (\mathbf{X}\beta_{\text{best}} + \mathbf{X}\delta)\|^2] \\
&= \|f - \mathbf{X}\beta_{\text{best}}\|^2 + \mathrm{E}[\|\epsilon - \hat{\epsilon}\|^2] \\
&= \|f - \mathbf{X}\beta_{\text{best}}\|^2 + (n - p),
\end{aligned}$$

where the first term is the model bias, and the second term is the same as in the previous subsection.

For the testing error, again let us assume that $\mathbf{X}$ is fixed, but we observe a new response, which is $\tilde{\mathbf{Y}} = f + \tilde{\epsilon}$. Then

$$\begin{aligned}
\mathrm{E}[\|\tilde{\mathbf{Y}} - \mathbf{X}\hat{\beta}\|^2] &= \mathrm{E}[\|(f + \tilde{\epsilon}) - (\mathbf{X}\beta_{\text{best}} + \mathbf{X}\delta)\|^2] \\
&= \|f - \mathbf{X}\beta_{\text{best}}\|^2 + \mathrm{E}[\|\tilde{\epsilon} - \hat{\epsilon}\|^2] \\
&= \|f - \mathbf{X}\beta_{\text{best}}\|^2 + (n + p),
\end{aligned}$$

where again the second term is the same as in the previous subsection.

Thus again the overfitting is $2p$.

If we increase the model complexity p (by adding more predictors), the training error will keep decreasing. But for the testing error, if will decrease first, because the model bias decreases, but after a certain point, the testing error will begin to increase, because the decrease in the model bias cannot compensate for the increase in the second term.

Now consider a general estimator $\hat{\beta}$, which is computed from $(\mathbf{X}, \mathbf{Y})$. The training error is

$$\mathrm{E}[\|\mathbf{Y} - \mathbf{X}\hat{\beta}\|^2] = \mathrm{E}[\|(f + \epsilon) - \mathbf{X}\hat{\beta}\|^2].$$

The testing error is

$$\mathrm{E}[\|\tilde{\mathbf{Y}} - \mathbf{X}\hat{\beta}\|^2] = \mathrm{E}[\|(f + \tilde{\epsilon}) - \mathbf{X}\hat{\beta}\|^2].$$

Again the overfitting is $2\mathrm{E}[\langle \epsilon, \hat{\epsilon} \rangle]$. Comparing it with the case for least squares, we may interpret $\mathrm{E}[\langle \epsilon, \hat{\epsilon} \rangle]$ as the effective degrees of freedom, which measures the capacity of the estimator to absorb the noises.

2.4 Bias and variance tradeoff

The testing error can be written as

$$\begin{aligned}
\mathrm{E}[\|\tilde{\mathbf{Y}} - \mathbf{X}\hat{\beta}\|^2] &= \mathrm{E}[\|(f + \tilde{\epsilon}) - \mathbf{X}\hat{\beta}\|^2] \\
&= \|f - \mathbf{X}\beta_{\text{best}}\|^2 + \mathrm{E}[\|\hat{\beta} - \beta_{\text{best}}\|^2] + \mathrm{E}[\|\tilde{\epsilon}\|^2] \\
&= \|f - \mathbf{X}\beta_{\text{best}}\|^2 + \|\mathrm{E}[\hat{\beta}] - \beta_{\text{best}}\|^2 + \mathrm{E}[\|\hat{\beta} - \mathrm{E}[\hat{\beta}]\|^2] + n,
\end{aligned}$$

where the second term of the second line is the estimation error. For the third line, the first term is model bias, the second term is the estimator bias, the third term is the estimator variance, and the last term is the observation error.

For the least squares estimator, the second term is 0, and the third term is p.

For general estimator, the second term may be greater than 0, but the third term can be much smaller than p.

We want p to be large to have small model bias. But we want our estimator to have some bias due to regularization, in order to greatly reduce the variance.

2.5 ℓ_2 regularization: Ridge, shrinkage, Stein

The least squares estimator minimizes $\|\mathbf{Y} - \mathbf{X}\beta\|^2$, which gives us

$$\hat{\beta}_{\text{LS}} = \mathbf{X}^\top \mathbf{Y} = \beta_{\text{best}} + \delta \sim \mathrm{N}(\beta_{\text{best}}, I_p),$$

where $\delta \sim \mathrm{N}(0, I_p)$.

The ridge regression minimizes $\|\mathbf{Y} - \mathbf{X}\beta\|^2 + \lambda\|\beta\|_{\ell_2}^2$, which gives us the shrinkage estimator $\mathbf{X}^\top \mathbf{Y}/(1 + \lambda)$. The Stein estimator

$$\hat{\beta}_{\text{Stein}} = \hat{\beta}_{\text{LS}} \left(1 - \frac{p - 2}{\|\hat{\beta}_{\text{LS}}\|^2}\right)$$

is a special case of the shrinkage estimator, and it has the remarkable property that

$$\mathrm{E}[\|\hat{\beta}_{\text{Stein}} - \beta_{\text{best}}\|^2] \leq \mathrm{E}[\|\hat{\beta}_{\text{LS}} - \beta_{\text{best}}\|^2],$$

no matter what β_{best} is.

2.6 ℓ_1 regularization: Lasso, sparsity

The Lasso estimator minimizes $\|\mathbf{Y}-\mathbf{X}\beta\|^2/2+\lambda\|\beta\|_{\ell_1}$, which gives us the soft-thresholding estimator

$$\hat{\beta}_{j,\text{Lasso},\lambda} = \text{sign}(\hat{\beta}_{j,\text{LS}}) \max(0, |\hat{\beta}_{j,\text{LS}}| - \lambda).$$

If we scale the data properly, we should have $\hat{\beta} = \beta_{\text{best}} + \delta$, but $\delta \sim \text{N}(0, I_p/n)$. Let us assume that β_{best} is sparse, so that only s components of the p-dimensional β_{best} are non-zero. Then on the happy event

$$\max_{j=1,\ldots,p} |\delta_j| \le \lambda,$$

we have

$$\|\hat{\beta}_{\text{Lasso}} - \beta_{\text{best}}\|^2 \le 4s\lambda^2.$$

We can choose λ to make the probability of the happy event to be at least $1-\eta$, by letting

$$\Pr(\max_{j=1,\ldots,p} \delta_j > \lambda) \le p \Pr(\delta_j > \lambda) \le 2pe^{-2n\lambda^2} = \eta,$$

so we can take

$$\lambda = \sqrt{(\log p + \log(2/\eta))/(2n)}.$$

This estimator is doing almost as well as if we know which of the $p-s$ components of β_{best} are zero.

We want to emphasize that hypothesis testing is a form of regularization, which gives us the hard thresholding,

$$\hat{\beta}_{j,\text{HT},\lambda} = \hat{\beta}_{j,\text{LS}} 1(|\hat{\beta}_{j,\text{LS}}| > \lambda),$$

where HT stands for hypothesis testing or hard thresholding.

We also want to emphasize that the regularization term corresponds to a Bayesian prior. For instance, the ridge regression corresponds to a prior $\beta \sim \text{N}(0, \tau^2 I_p)$ for a τ^2.

3 Overfitting in classification

3.1 Empirical risk minimization

We can write the linear classification model as $\mathbf{Y} \approx \text{sign}(\mathbf{X}^\top\beta)$. We may estimate β by minimizing the empirical risk

$$\hat{\beta} = \arg\min_{\beta} \|\mathbf{Y} - \text{sign}(\mathbf{X}\beta)\|^2$$

where $\text{sign}(r) = +1$ if $r \ge 0$, and $\text{sign}(r) = -1$ if $r < 0$. The sign function is applied element-wise. $\mathbf{Y}$ is a vector of +1 or -1, so is $\text{sign}(\mathbf{X}\beta)$. Thus $\|\mathbf{Y}\|^2 = \|\text{sign}(\mathbf{X}\beta)\|^2 = n$.

We can define the training error as

$$\|\mathbf{Y} - \text{sign}(\mathbf{X}\beta)\|^2/(4n) = [\|\mathbf{Y}\|^2 + \|\text{sign}(\mathbf{X}\beta)\|^2 - 2\langle \mathbf{Y}, \text{sign}(\mathbf{X}\beta)\rangle]/(4n) = 1/2[1 - \langle \mathbf{Y}, \text{sign}(\mathbf{X}\beta)\rangle/n].$$

We can treat $\langle \mathbf{Y}, \text{sign}(\mathbf{X}\beta)\rangle/n$ as the training score. Minimizing the training error is the same as maximizing the training score.

We write in the above form in order to seek an analogy to the regression setting. It is not really used in practice, but it makes the theoretical understanding concrete.

3.2 When Y is a noise vector

Consider the special case $\mathbf{Y} = \epsilon$, where $\epsilon_i \sim$ Bernoulli(1/2), i.e., $\Pr(\epsilon_i = +1) = \Pr(\epsilon_i = -1) = 1/2$. That is, ϵ is a sequence of independent coin flips. Suppose we estimate β by maximizing the training score $\langle \mathbf{Y}, \text{sign}(\mathbf{X}\beta)\rangle/n$ to obtain $\hat{\beta}$. Let $\hat{\epsilon} = \text{sign}(\mathbf{X}\hat{\beta})$ be the fitted vector. Then the expected training score is $\mathrm{E}[\langle \epsilon, \hat{\epsilon}\rangle/n]$.

For testing, suppose we observe a testing vector $\tilde{\epsilon}$ of coin flips, which is independent of the training vector ϵ. Then $\mathrm{E}[\langle \tilde{\epsilon}, \hat{\epsilon}\rangle] = 0$. Thus the overfitting is $\mathrm{E}[\langle \epsilon, \hat{\epsilon}\rangle/n]$, which is the same as in regression. This is called Rademacher complexity of the model.

3.3 When Y has both signal and noise

In the case of y_i that may depend on X_i. We can consider the following quantity as a measure of overfitting

$$\mathrm{E}[\max_{\beta} \langle \mathbf{Y} \cdot \epsilon, \text{sign}(\mathbf{X}\beta)\rangle/n]$$

where again ϵ is a vector of Bernoulli random variables, where $\mathbf{Y} \cdot \epsilon$ is a vector of $y_i\epsilon_i$. If $\epsilon_i = +1$, we may treat example i as the "training" example, and if $\epsilon_i = -1$, we may treat example i as the "testing" example. Then $\langle \mathbf{Y} \cdot \epsilon, \text{sign}(\mathbf{X}\beta)\rangle/n$ is the difference between the "training" score and the "testing" score, which is overfitting. This is commonly done in cross validation, where we randomly select some examples as "training" examples, and treat the rest as the "testing" examples.

The key is that $y_i\epsilon_i$ is also a Bernoulli random variable., i.e., $\mathbf{Y} \cdot \epsilon$ has the same distribution as ϵ. Thus the overfitting is again $\mathrm{E}[\langle \epsilon, \hat{\epsilon}\rangle/n]$.

The above reasoning is called symmetrization trick.

3.4 Tail bound

Instead of expectation, we may also analyze the tail behavior of $\langle \epsilon, \hat{\epsilon}\rangle/n$.

Suppose there are K possible values of β, denoted by $\{\beta^{(k)}, k = 1, ..., K\}$. The training score is

$$\max_{k=1,...,K} \frac{1}{n} \langle \epsilon, \text{sign}(\mathbf{X}\beta^{(k)})\rangle.$$

According to the union bound and the Hoeffding inequality,

$$\Pr\left(\max_{k=1,...,K} \frac{1}{n} \langle \epsilon, \text{sign}(\mathbf{X}\beta^{(k)})\rangle > t\right) \le Ke^{-2nt^2}.$$

K measures the model capacity. If we control K, then we control the overfitting.

One may wonder it may not be realistic to assume that there are only K possible values of β. Actually it is very realistic. Consider the set $\mathcal{F} = \{\text{sign}(\mathbf{X}\beta), \forall\beta\}$, for each β, $\text{sign}(\mathbf{X}\beta)$ is a binary sequence. There are at most 2^n such sequences. In fact the number of sequences in $\mathcal{F}$ is bounded by $(en/p)^p$. Thus we can take $K = (en/p)^p$ in the above bound.

3.5 General function class

In the linear classifier, we assume $y_i \approx \text{sign}(X_i^\top \beta)$, and the function class is $\mathcal{F} = \{f(X) = \text{sign}(X^\top\beta), \forall\beta\}$. We can generalize the linear class to a more general function class $\mathcal{F} = \{f(X)\}$. Then the overfitting can be measured by the Rademacher complexity

$$\mathrm{E}\left[\max_{f\in\mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(X_i)\right].$$

Or the so-called growth number K, which counts the number of sequences in $\{(f(X_1), ..., f(X_n))^\top, \forall f \in \mathcal{F}\}$. Then according to the union bound and Hoeffding, the tail probability

$$\Pr\left(\max_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \epsilon_i f(X_i) > t\right) \leq Ke^{-2nt^2/2}.$$

Again this number K is bounded by $(en/p)^p$, where p is the VC dimension, which can be defined as the maximal number of coin flips that can be perfectly explained by finding $f \in \mathcal{F}$. For linear classifier, the VC dimension is p, the dimension of β. This is again analogous to regression, where the dimension p is the number of noises that can be perfectly explained by least squares. The VC dimension and the Rademacher complexity are closely related to each other.

3.6 Regularized estimators

The support vector machine corresponds to the ridge regression that minimize the loss function

$$\sum_{i=1}^{n} \max(0, 1 - y_i X_i^\top \beta) + \lambda \|\beta\|_{\ell_2}^2.$$

The adaboost corresponds to the Lasso regression computed by coordinate descent on the loss function

$$\sum_{i=1}^{n} \exp(-y_i X_i^\top \beta),$$

where X_i is a vector of binary features (or weak classifiers).

4 Take home message

For both regression and classification, the overfitting can be measured by the capacity of the model to fit the noise vector, i.e., $\mathrm{E}[\langle \epsilon, \hat{\epsilon} \rangle]$, where $\hat{\epsilon}$ is computed from $(\mathbf{X}, \epsilon)$. $\hat{\epsilon}$ is the best fit the model can offer for ϵ.

In regression, ϵ is a vector of Gaussian white noises. In classification, ϵ is a vector of Bernoulli coin flips.

We analyze both regression and classification by starting from the situation where $\mathbf{Y} = \epsilon$. We then analyze the more realistic situation where $\mathbf{Y}$ depends on $\mathbf{X}$. In both regression and classification, we reduce this realistic situation to the situation of noise vector. In regression, the reduction can be achieved by cancelation of the signal. In classification, the reduction is achieved by the symmetrization trick, which also targets cancellation of the signal.

The overfitting defines the model capacity, or model complexity, or effective degrees of freedom.

Stronger regularization (big λ) means:

(1) smaller effective model complexity or model capacity

(2) bigger bias, smaller variance

(3) bigger training error, smaller overfitting = testing error - training error.

We want

(1) small testing error = training error + overfitting.

(2) small mean square error = bias^2 + variance.

We can tune λ by cross-validation. An over-regularized model may become too dumb, but an under-regularized model may grow superstitious.