CS 181 Spring 2017 Section 2 Notes (Model Selection)

1 Validation

1.1 Linear Regression

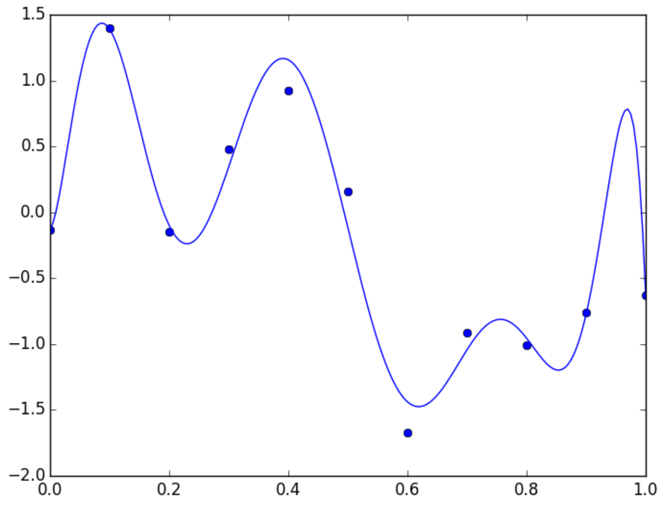
Suppose we have data $\{(x_i, y_i)\}_{i=1}^n$, with $x_i, y_i \in \mathbb{R}$, and we want to fit polynomial basis functions:

$$\boldsymbol{\phi}(x)^\top = [\phi_1(x) = 1, \phi_2(x) = x, \dots, \phi_{d+1}(x) = x^d]$$

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\phi}(x)$$

That is, we fit a degree d polynomial.

With a small dataset and too high of a d, we get overfitting:



Obviously, this will generalize poorly to new data points. How can we solve this problem?

1.2 Ridge Regression

One solution to overfitting linear regression is through ridge regression.

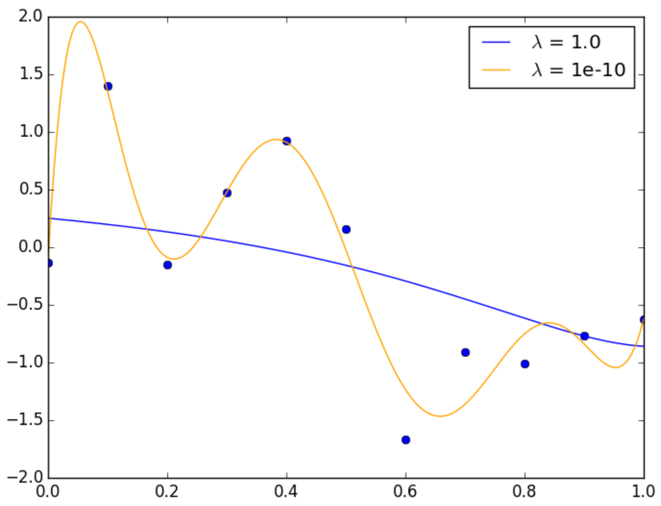
Recall that linear regression minimizes the least squares loss function:

$$\mathcal{L}(D) = \sum_{i=1}^{n} (y_i - h(x_i; \mathbf{w}))^2$$

while ridge regression minimizes

$$\mathcal{L}(D) = \sum_{i=1}^{n}(y_i - h(x_i; \mathbf{w}))^2 + \frac{\lambda}{2}||\mathbf{w}||^2$$

Ridge regression is also known as weight shrinkage. Indeed, the extra term penalizes overly large weights in $\mathbf{w}$, leading to smaller coefficients for a "flatter" polynomial:



Ridge regression is great for regularizing our model, i.e. making it simpler and allowing it to generalize better to new data.

This, however, leads us to a new question: how do we choose λ? For that matter, how do we choose the degree d of our polynomial to begin with?

Exercise: Suppose we have some data matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ and targets $\mathbf{y} \in \mathbb{R}^n$. Suppose the data are orthogonal [1], i.e. satisfies $\mathbf{X}^\top \mathbf{X} = \mathbf{I}$. Show that if $\widehat{\mathbf{w}}$ is the solution to linear regression, and $\widehat{\mathbf{w}}_{ridge}$ is the solution to ridge regression, then

$$\widehat{\mathbf{w}}_{ridge} = \frac{1}{1+\lambda}\widehat{\mathbf{w}}$$

This explicitly illustrates the phenomenon of weight shrinkage.

Solution: Recall that the linear regression solution is

$$\widehat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

[1]Orthogonal data is a very special case in which the inner product between any two distinct features is zero. Normally we expect features to be correlated. But it is used to gain this clean illustration of the effect of ridge regression. Technically, we have $\mathbf{X} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ where $\mathbf{v}_1, \ldots, \mathbf{v}_m$ are n dimensional, orthogonal column vectors.

and recall from homework that the ridge regression solution is

$$\widehat{\mathbf{w}}_{ridge} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$$

If $\mathbf{X}^\top\mathbf{X} = \mathbf{I}$, we see that

$$\widehat{\mathbf{w}} = \mathbf{X}^\top\mathbf{y}$$
$$\widehat{\mathbf{w}}_{ridge} = \frac{1}{1+\lambda}\mathbf{X}^\top\mathbf{y}$$

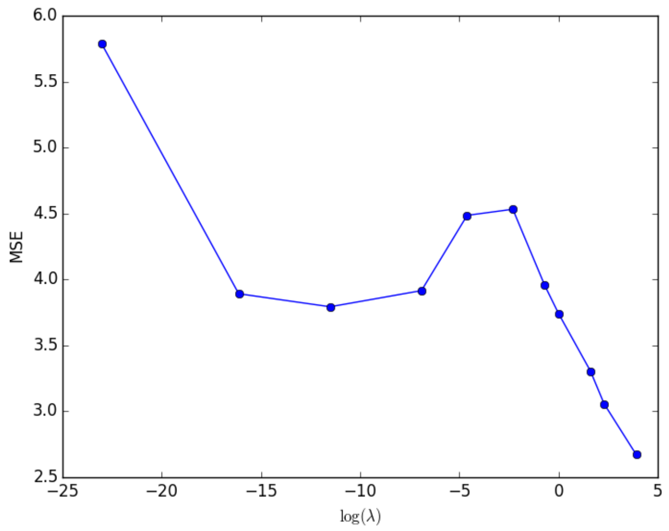
hence giving us the result.

1.3 Validation Set: Model Selection

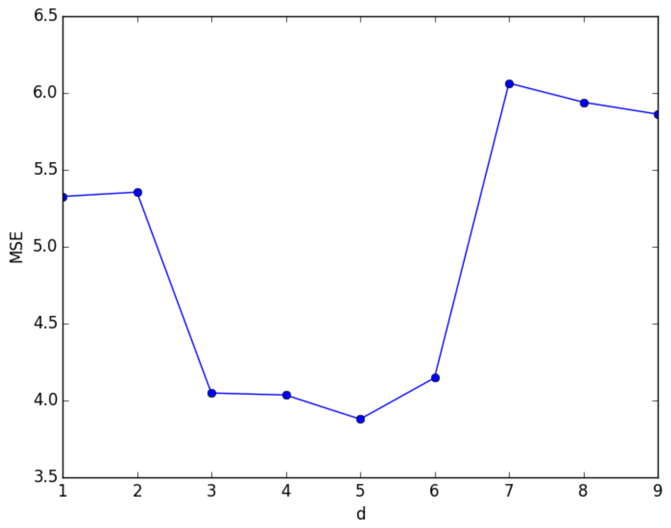
We can do model selection through a validation set, data that are separate from our training set used to fit the regression.

By separating our full dataset into a training set and validation set (say in a 90%-10% Split), we can use our validation set to check our model's generalization ability.

We can vary λ, and for each λ, check the model's least squares loss on the validation set:



We can do the same for d:



We can even do both at the same time! This can be done through grid search, where we search over a grid of values (λ, d) to find the jointly best pair for our problem on the validation set.

Once we have our best (λ, d) pair, we have found our best regularized model.

1.4 Cross Validation

Cross validation is a more sophisticated technique for obtaining validation losses.

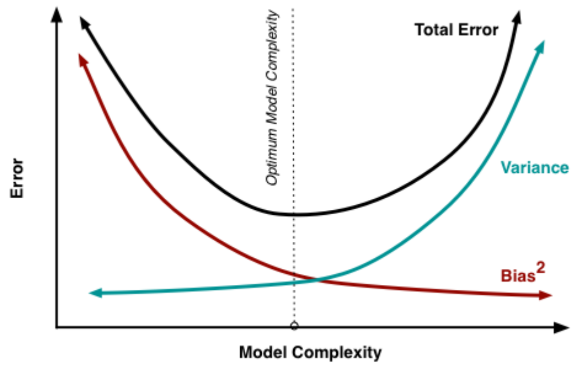
Instead of splitting our data once into a 90%/10% training/validation sets, in 10-fold cross validation, we split our data into 10 equal chunks. For each chunk, we set it to be the validation set and use the rest of the data to fit our model. Then, we obtain a validation loss on our current chunk – averaging over the 10 chunks gives the final validation loss.

Cross validation can be used exactly as described above to find optimal (λ, d) values. We simply have an improved way of computing validation losses by averaging.

The reason to use cross validation is when there is not very much data and we want to get a better estimate of the validation loss. Each example is used exactly once in estimating the validation loss.

2 Bias-Variance Decomposition

Bias-variance decomposition is a way of understanding how different sources of error (bias and variance) can affect the final performance of a model. A tradeoff between bias and variance is often made when selecting models to use, and can be informed by the results of the bias-variance decomposition.



- Simple models under-fit: will deviate from data (high bias) but will not be influenced by peculiarities of data (low variance).
- Complex models over-fit: will not deviate systematically from data (low bias) but will be very sensitive to data (high variance).

For this analysis, we are interested in the generalization error - the expected error, in least squares terms, on an unseen sample. We will show that we can decompose this error into the sum of bias squared (systematic error), variance (sensitivity of prediction), and noise (irreducible error). Let us define some variables as follows:

- h_D: The trained model, $h_D : \mathcal{X} \mapsto \mathbb{R}$.
- D: The data, a random variable sampled $D \sim F^n$.
- x: A new input.
- y: The true result of input x. Conditioned on x, y is a r.v. (may be noise.)
- $\overline{y}$: The true conditional mean, $\overline{y} = \mathbb{E}_{y|\mathbf{x}}[y]$.
- $\overline{h}(\mathbf{x})$: The prediction mean, $\overline{h}(\mathbf{x}) = \mathbb{E}_D[h_D(\mathbf{x})]$.

We're interested in the generalization error at x:

$$\begin{aligned}
&\mathbb{E}_{D,\,y|\mathbf{x}}[(y - h_D(\mathbf{x}))^2] \\
&= \mathbb{E}_{D,\,y|\mathbf{x}}[(y - \overline{y} + \overline{y} - h_D(\mathbf{x}))^2] \\
&= \underbrace{\mathbb{E}_{y|\mathbf{x}}[(y-\overline{y})^2]}_{\text{noise}} + \underbrace{\mathbb{E}_D[(\overline{y} - h_D(\mathbf{x}))^2]}_{\text{bias+var}} + \underbrace{2\mathbb{E}_{D,\,y|\mathbf{x}}[(y-\overline{y})(\overline{y} - h_D(\mathbf{x}))]}_{0}
\end{aligned} \tag{1}$$

The last term can be written as

$$2\mathbb{E}_D[\overline{y} - h_D(\mathbf{x})] \cdot \mathbb{E}_{y|\mathbf{x}}[y - \overline{y}] = 2\mathbb{E}_D[\overline{y} - h_D(\mathbf{x})] \cdot 0 = 0.$$

Expanding the second term in (1), we have

$$\begin{aligned}
&\mathbb{E}_D[(\overline{y} - h_D(\mathbf{x}))^2] \\
&= \mathbb{E}_D[(\overline{y} - \overline{h}(\mathbf{x}) + \overline{h}(\mathbf{x}) - h_D(\mathbf{x}))^2] \\
&= \underbrace{(\overline{y} - \overline{h}(\mathbf{x}))^2}_{\text{bias squared}} + \underbrace{\mathbb{E}_D[(\overline{h}(\mathbf{x}) - h_D(\mathbf{x}))^2]}_{\text{variance}} + \underbrace{2\mathbb{E}_D[(\overline{y} - \overline{h}(\mathbf{x}))(\overline{h}(\mathbf{x}) - h_D(\mathbf{x}))]}_{0}
\end{aligned} \tag{2}$$

The last term can be written as

$$2(\overline{y} - \overline{h}(\mathbf{x}))\mathbb{E}_D[\overline{h}(\mathbf{x}) - h_D(\mathbf{x})] = 2(\overline{y} - \overline{h}(\mathbf{x}))(0) = 0.$$

Substituting (2) back into (1), we have:

$$\begin{aligned}
&\mathbb{E}_{D,\, y|\mathbf{x}}[(y - h_D(\mathbf{x}))^2] \\
&= \mathbb{E}_{y|\mathbf{x}}[(y - \overline{y})^2] + (\overline{y} - \overline{h}(\mathbf{x}))^2 + \mathbb{E}_D[(\overline{h}(\mathbf{x}) - h_D(\mathbf{x}))^2] \\
&= \text{noise}(\mathbf{x}) + (\text{bias}(h(\mathbf{x})))^2 + \text{var}_D(h_D(\mathbf{x})).
\end{aligned}$$

Considering the expectation over x, the generalization error is:

$$\mathbb{E}_{\mathbf{x}}\left[\text{noise}(\mathbf{x}) + (\text{bias}(h(\mathbf{x})))^2 + \text{var}_D(h_D(\mathbf{x}))\right]$$

Exercise: We consider a very simple example where the data is a univariate Gaussian,

with $x_i \sim \mathcal{N}(\mu, 1)$ with known variance but unknown mean. In this case, there are no features, and the hypothesis doesn't depend on x. A very simple hypothesis, for example, is the sample mean

$$h_D = \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

for data $(x_1, \ldots, x_n) \in \mathbb{R}^n$. Calculate the bias and variance for the following two hypotheses:

1. Estimate 1: Use the same mean of data D.
2. Estimate 2: Use the constant hypothesis, 0.

Solution:

1. The bias is $\mu - E_D[\overline{x}] = \mu - E_D[(1/n)\sum x_i] = 0$. The variance is $E_D[(\overline{x} - \mu)^2] = \sigma^2/n$, the variance of the sample mean on n examples (as is standard).
2. The bias equals μ, the expected difference between the estimator and the true value. This prediction is constant, so the variance of our prediction is 0.

2.1 Limitations

Although the bias-variance decomposition provides some interesting insights into model selection from a complexity perspective, it has limited practical value, as it is based on averages of independent data sets drawn from some distribution. In practice, however, we only have a single observed data set. The bias and variance can be estimated through "bootstrap" style approaches where we sample with replacement to form additional data sets, but still— why not more directly compute validation loss and use this to find the best model? The main interest in the bias-variance decomposition is to gain conceptual insight.

3 Ensemble Methods

Ensemble methods take advantage of multiple models to obtain better predictive accuracy than with a single model alone. The two most common types of ensemble methods are bagging and boosting.

3.1 Bootstrap aggregating (Bagging)

In bagging, we fit each individual model (i.e. learning each of the hypotheses $h_1, ..., h_r$) on a random sample of the training set. When it comes to predicting data in the test set, we either use an average of the predictions from the individual models (in regression problems) or take the majority vote (in classification problems). Because it is an averaging of models, bagging tends to decrease the variance of a learning algorithm without changing the bias. An example is a random forest, which is made up of multiple decision trees.

In particular, a random forest is bagging applied to randomized decision trees. It trains multiple trees, eventually taking the average or mode prediction from the ensemble of learned models. The approach of random forests is to repeat:

- sample a data set of size n with replacement
- train a decision tree (classification) or regression tree, randomly selecting at each split a candidate set of features to split on (e.g., $\sqrt{m}$ features). This provides higher variance, since different features are made available to different trees.

Random Forests work very well in practice!

3.2 Boosting

In boosting, we train the individual models sequentially. Thus, after training the i^{th} model on a sample of the training set, we train the $(i+1)^{th}$ model on a new sample based on what the i^{th} model predicted to be correct. Specifically, examples classified incorrectly in the previous step receive higher weights in the new sample. During testing, we take a weighted average or weighted majority vote of the models' predictions based on their respective training accuracies on their reweighted training data (i.e. higher models have larger weights). A common example is the Adaboost algorithm.

4 scikit-learn

Please refer to sklearn-demo.ipynb. To download Jupyter Notebooks, refer to `http://jupyter.readthedocs.io/en/latest/install.html`.