CS 181 Spring 2024 Section 3 Notes (Model Selection)

1 Model Selection

1.1 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.

1.2 Exercise: Bias-Variance Decomposition

Decompose the generalization error (or mean squared error) into the sum of squared bias (systematic error), variance (sensitivity of prediction), and noise (irreducible error) by following the steps below. You will find the following notation useful:

- f_D: The trained model, $f_D : \mathcal{X} \mapsto \mathbb{R}$.
- D: The data, a random variable sampled from some distribution $D \sim F^n$.
- $\mathbf{x}$: A new input.
- y: The true result of input $\mathbf{x}$. Conditioned on $\mathbf{x}$, y is a r.v. (there may be noise involved.)
- $\overline{y}$: The true conditional mean, $\overline{y} = \mathbb{E}_{y|\mathbf{x}}[y|\mathbf{x}]$.
- $\overline{f}(\mathbf{x})$: The predicted mean from the model, $\overline{f}(\mathbf{x}) = \mathbb{E}_D[f_D(\mathbf{x})]$.
- $E_D(\cdot)$ is the expectation with respect to the data (for a quantity that depends on the data, and over the data's distribution). $E_{y|\mathbf{x}}(\cdot)$ is the expectation of a quantity over the conditional distribution of y given $\mathbf{x}$.

1. Start with the equation for the mean squared error or generalization error:

$$\mathbb{E}_{D,\, y|\mathbf{x}}[(y - f_D(\mathbf{x}))^2]$$

and use the linearity of expectation to derive an equation of the form:

$$\underbrace{\mathbb{E}_{y|\mathbf{x}}[(y - \overline{y})^2]}_{\text{noise}} + \underbrace{\mathbb{E}_D[(\overline{y} - f_D(\mathbf{x}))^2]}_{\text{bias+var}} + \boxed{???} \tag{1}$$

where the $\boxed{???}$ denotes a third term. What is this term? **Hint**: add and subtract $\overline{y}$.

2. Show that this third term is equal to 0 (**Hint**: take advantage of the fact that $\overline{y}$ and $f_D(\mathbf{x})$ do not depend on $y|\mathbf{x}$).
3. The first term in (1) is the noise. We therefore want to decompose the second term into the bias and variance. Again, using the linearity of expectation, re-write the second term in equation (1) in the form:

$$\underbrace{(\overline{y} - \overline{f}(\mathbf{x}))^2}_{\text{bias squared}} + \underbrace{\mathbb{E}_D[(\overline{f}(\mathbf{x}) - f_D(\mathbf{x}))^2]}_{\text{variance}} + 2\mathbb{E}_D[(\overline{y} - \overline{f}(\mathbf{x}))(\overline{f}(\mathbf{x}) - f_D(\mathbf{x}))] \tag{2}$$

Show that the third term is equal to 0.

4. Plug the results of part 3 back into (1) to show that we have decomposed the error into noise, bias, and variance.

1.3 Exercise: Bias-Variance Tradeoff in Estimating an Unknown Parameter

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, we want to estimate the true mean μ - and we are dealing with only one variable, i.e. the x_i's. A very simple hypothesis, for example, is the sample mean

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

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.

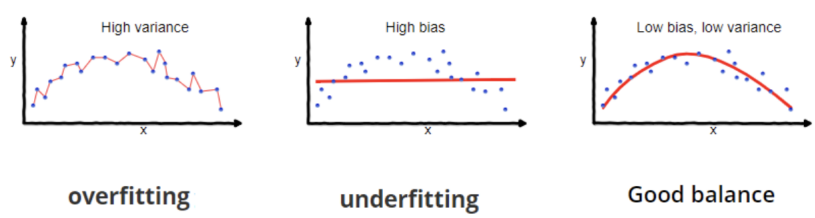

Figure 1: A helpful image to illustrate this tradeoff taken from here.

1.4 Limitations

How can you test the "variance" of your estimate/model (i.e., how it would vary if you estimated it a bunch of times on **a bunch of datasets drawn from the same distribution**), when in practice you only have access to **one dataset**?

We can solve this problem by "bootstrapping" (i.e. sampling a bunch of "synthetic" datasets from the data we have, with replacement). But there are easier ways to solve this problem.

1.5 Validation Set

A *validation set* contains data that are separate from our training set used to fit the regression. Here is a sample process:

1. Separate our full dataset into a training set and validation set (say in a 90/10 split).
2. Train your models with different parameters on the training set. Each time, check the performance on the validation set.
3. This gives you an optimal value for the parameter.

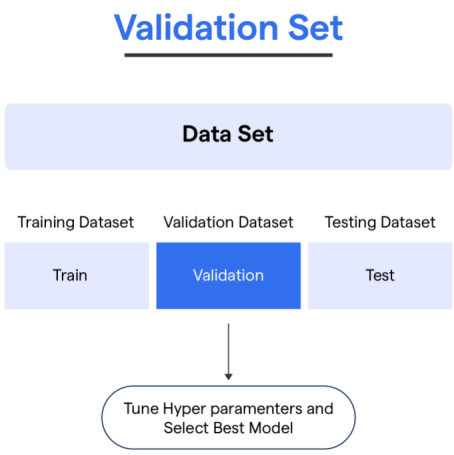

Figure 2: A helpful image to illustrate the validation set taken from here.

1.6 Cross Validation

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

1. In k-fold cross validation, we split our data into k equal chunks.
2. For each chunk, we set it to be the validation set and use the rest of the $k-1$ chunks together as the training data to fit our model.
3. Then, we obtain a validation loss on our current chunk.
4. Averaging loss over the k chunks gives us a final validation loss.

Now, we have an improved way of computing validation losses by averaging. This reduces the variance in the resulting validation loss - since we've used every data point in doing so!

See this notebook for an interactive demo of how cross validation could be used.

1.7 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 are bagging and boosting.

1.7.1 Bootstrap aggregating (Bagging)

- In bagging, we fit each individual model on a random sample of the training set.

- To predict data in the test set, we either use an average of the predictions from the individual models (for regression) or take the majority vote (for classification).
- This tends to lower variance without changing bias, since it's an average of models!
- **Example:** Random forest, which is an average of predictions from decision trees!

1.7.2 Boosting

- In boosting, we train the individual models sequentially. After training the ith model on a sample of the training set, we train the $(i+1)^{th}$ model on a new sample based on the performance of the ith model.
- Thus, examples classified incorrectly in the previous step receive higher weights in the new sample, encouraging the new model to focus on those examples.
- 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).
- **Example:** The Adaboost algorithm is a common example.

1.8 Exercise: Model Selection Using Bias and Variance

You have a dataset about n dogs. Your want to model weight of a dog (dependent var.) using age (independent var.). You fit and evaluate three models using the same train-test split, a linear model, a quadratic model, and a cubic model. Below is a table of the train and test accuracies.

	Linear	Quadratic	Cubic
Train	0.60	0.82	0.93
Test	0.62	0.73	0.54

1. For each model, would you choose to regularize? What would regularization do?
2. For each of the three models, what would be the effect of adding more data and why?
3. How would each of these models perform on a freshly drawn set of dogs? Assume that the draws across both data sets are i.i.d. (i.e. using same breeds, etc. in both data sets).
4. Based on these results, which model do you think is the most appropriate for this data?

2 Regularization

2.1 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:

$$\boldsymbol{\phi}(x)^\top = [\phi_1(x) = 1, \phi_2(x) = x, \ldots, \phi_{d+1}(x) = x^d]$$
$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \boldsymbol{\phi}(x)$$

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?

Recall from Homework 1 that for bases:

- $\phi_j(x) = \cos(f(x)/j)$ for $j = 1 \ldots 9$
- $\phi_j(x) = \cos(f(x)/j)$ for $j = 1 \ldots 49$

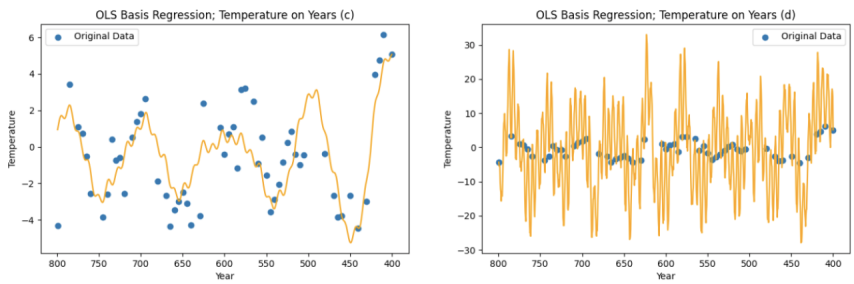

Figure 3: Plots from Hw1 Part 3 Basis Transformations c and d

2.2 Penalized Loss Function

Recall that the standard linear regression problem, known as *ordinary least squares (OLS)*, uses the following loss function (which is just the mean squared error):

$$\mathcal{L}_{OLS}(D) = MSE = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2$$

Regularization refers to the general practice of modifying the model-fitting process to avoid overfitting. Linear models are typically regularized by adding a *penalization term* to the loss function. The penalization term is simply any function R of the weights $\mathbf{w}$ scaled by a penalization factor λ. The loss then becomes:

$$\mathcal{L}_{reg}(D) = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2 + \lambda R(\mathbf{w})$$

There are some common choices for $R(\mathbf{w})$ that will be discussed. They frequently leverage the idea of a vector norm, where $||\mathbf{w}||_p$ represents the L_p-norm of the vector $\mathbf{w}$ for $p \geq 1$:

$$||\mathbf{w}||_p = \left(\sum_d |\mathbf{w}_d|^p \right)^{1/p}$$

A note on gradient descent. Referring to section 3.4 of the textbook, we can use gradient descent to update our weight parameters $\mathbf{w}$ using our loss function $\mathcal{L}(\mathbf{w})$. More specifically, the weight parameters at each time step t can be found using the following update rule:

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \alpha \nabla \mathcal{L}(\mathbf{w}^{t-1})$$

where $\alpha > 0$ is the learning rate.

2.3 LASSO Regression

One common choice for a penalization term is simply $R(\mathbf{w}) = ||\mathbf{w}||$. This is just the L_1-norm of the weights vector, which quite naively means that the penalization term here is just the sum of the magnitudes of all the weights for the model. This form of regularized regression is known as *LASSO (Least Absolute Shrinkage and Selection Operator)* regression. The full modified loss is then:

$$\mathcal{L}_{LASSO}(D) = \sum_{i=1}^{N} (y_i - f(x_i; \mathbf{w}))^2 + \lambda ||\mathbf{w}||$$

There are some notable properties of LASSO regression. One main disadvantage is that it does not have a closed-form solution, meaning that it cannot be analytically solved. Therefore, it needs to be numerically solved through an iterative process, which can be much slower.

> **Concept Question**: Why do you think LASSO has no closed-form solution? Try to solve for it using the same process as for the OLS solution; what goes wrong?

However, it does have the benefit, as the name suggests, that it is good for *variable selection*, meaning that coefficients for some variables (ones that have low predictive power) might be "shrunk" directly to zero.

2.4 Ridge Regression

Another solution to overfitting linear regression is through ridge regression, which minimizes a modified least squares loss function:

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

Ridge regression is used to *regularize* a model, making it simpler and allowing it to generalize better to new data. Indeed, the extra term penalizes overly large weights in $\mathbf{w}$, leading to smaller coefficients for a "flatter" polynomial.

Unlike LASSO, ridge regression has a closed form solution, which makes the solution much more computationally efficient. While it does not shrink coefficients to zero, it has other intuitive properties, such as connection to a Normal prior. The analytical solution is:

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

The above expression can be compared to the OLS solution. The only additional term is $\lambda \mathbf{I}$, which looks like a "ridge" of λ values (hence the name ridge regression). This also helps avoid problems with singular data.

Concept Question: Solve for the closed form solution to ridge regression.

2.5 Exercise: Ridge Regression

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*, 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.

* 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.

2.6 Exercise: Lasso Regularization from Lagrange Multipliers

Show that minimization of the unregularized sum-of-squares error function given by

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n))^2,$$

subject to the constraint

$$\sum_{d=1}^{D}|w_d| \leq \eta,$$

is equivalent to minimizing the regularized error function

$$\frac{1}{2}\sum_{n=1}^{N}(t_n - \mathbf{w}^\mathsf{T}\phi(\mathbf{x}_n))^2 + \frac{\lambda}{2}\sum_{d=1}^{D}|w_d|$$

Hint: Lagrange Multipliers. Lagrange Multipliers are a method for optimizing a function f with respect to an equality constraint, i.e.

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } g(\mathbf{x}) = 0.$$

This can be turned into an unconstrained problem by introducing a Lagrange multiplier λ and constructing the Lagrangian function,

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}).$$

It can be shown that it is a necessary condition that the optimum is a critical point of this new function. We can find this point by solving two equations:

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = 0 \text{ and } \frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} = 0$$