Section 5 - Problem solutions

Problem 1

Suppose we have a classification problem with a binary response $Y$ and a $p$-dimensional predictor variable $X = (X_1, \ldots, X_p)$. Logistic regression is fitted to a set of $n$ samples. Then, logistic regression is fitted again to the same observations, where we include one additional predictor, such that:

$$X = (X_1, \ldots, X_p, X_{p+1}).$$

Explain how the training error, test error, and coefficients change in each of the following cases:

(a) $X_{p+1} = X_1 + 2X_p$.
(b) $X_{p+1}$ is a random variable independent of $Y$.

Solution:

(a) Since the new predictor is exactly collinear with 2 of the old predictors, the coefficients $\beta_1$, $\beta_p$, and $\beta_{p+1}$ are unidentifiable, as logistic regression maximizes a likelihood which only depends on a linear combination of the predictors. The predictions remain unchanged, and therefore so do the training and test errors.

(b) Since the number of samples is finite, logistic regression may assign a positive coefficient to $X_{p+1}$ even though it is independent of the response; this will likely affect other coefficients as well. The training error can only decrease, whereas the test error will increase because the bias remains the same while variance increases.

Problem 2

Assume we have a set of data from patients who have visited UPMC hospital during the year 2011. A set of features (e.g., temperature, height) have been also extracted for each patient. Our goal is to decide whether a new visiting patient has any of diabetes, heart disease, or Alzheimer (a patient can have one or more of these diseases). We have decided to use a neural network to solve this problem. We have two choices: either to train a separate neural network for each of the diseases or to train a single neural network with one output neuron for each disease, but with a shared hidden layer. Which method do you prefer? Justify your answer.

Solution:

1- Neural network with a shared hidden layer can capture dependencies between diseases. It can be shown that in some cases, when there is a dependency between the output nodes, having a shared node in the hidden layer can improve the accuracy.

2- If there is no dependency between diseases (output neurons), then we would prefer to have a separate neural network for each disease.
Problem 3

(i) (6 points) Suppose you have regression data generated by a polynomial of degree 3. Characterize the bias-variance of the estimates of the following models on the data with respect to the true model by circling the appropriate entry.

<table>
<thead>
<tr>
<th>Model</th>
<th>Bias</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>low / high</td>
<td>low / high</td>
</tr>
<tr>
<td>Polynomial regression with degree 3</td>
<td>low / high</td>
<td>low / high</td>
</tr>
<tr>
<td>Polynomial regression with degree 10</td>
<td>low / high</td>
<td>low / high</td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>Model</th>
<th>Bias</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>low / high</td>
<td>low / high</td>
</tr>
<tr>
<td>Polynomial regression with degree 3</td>
<td>low / high</td>
<td>low / high</td>
</tr>
<tr>
<td>Polynomial regression with degree 10</td>
<td>low / high</td>
<td>low / high</td>
</tr>
</tbody>
</table>

(ii) (8 points) For each part below indicate whether we would generally expect the performance of a flexible statistical learning method to be better or worse than an inflexible method. Justify your answer.

(i) The sample size $n$ is extremely large, and the number of predictors $p$ is small.

Solution: A flexible method is better because we are less at risk of overfitting if we have lots of data and only a few relevant predictors. Note you could make an argument that an inflexible method is better since we don’t have a lot of predictors. This is not incorrect but, if we have lots of data, use it! It doesn’t get as much use in inflexible models.

(ii) The number of predictors $p$ is extremely large, and the number of observations $n$ is small.

Solution: Use an inflexible method. Since $p$ is very large, it is easy to overfit or to incorporate predictors into the model that are not actually helpful in predicting the output (e.g. your model may capture noise/variance). Also for small $n$ you are much more likely to see spurious relationships that aren’t actually present in the population.

(iii) The relationship between the predictors and response is highly non-linear.

Solution: Flexible models. Flexible models will allow you to capture different (non-linear) relationships. Unless you know exactly what the relationship between $X$ and $Y$ is and you choose a very inflexible model that happens to capture a specific non-linear relationship that is true in the world, a flexible hypothesis space is more likely to work better.

(iv) The variance of the error terms, i.e. $\sigma^2 = \text{Var}(c)$ is extremely high.

Solution: Inflexible. You are very likely to find relationships that are just due to noise. Flexible models will also try to find and fit patterns in the irreducible noise which will cause high variance in the final model.

Problem 4:
Applying the methodology seen in class, you split your dataset in 3 sub-datasets, train 10 models on the training set and then choose the model on the validation set. We achieve good performance on the validation set but then the testing error is terrible. What could be the problem? Give 2 reasons

Two sources of problems: the data or the model. Think about could go wrong with either?
1. We chose the model with lower error on validation, therefore we have a lower estimate of the test error. This is overfitting on the validation set.
2. Training-validation data was not representative of the global data: selection bias. Example: you classify images. All your train & validation examples are day time images, but your test images are night time images.

Problem 5:
Assume we try to estimate a quantity let’s call it $\mu$. We know that $Y$ is random variable whose distribution is the Normal distribution centered on $\mu$ and with standard deviation $\sigma$.
Imagine we can sample (do random draws) $Y$. This gives $Y_1, Y_2, ..., Y_n$
Question 1: How would you estimate $\mu$?
Question 2: The problem of the previous estimator is that it depends a lot on the standard deviation $\sigma$. What can we do to avoid that problem?
Question 3: If the cost of sampling is high, we need another way to get a better estimate of $\mu$. Can you think of another measure to estimate $\mu$? How is the bias/variance of this new estimator?

1. We take the sample mean (we can say it is maximizing likelihood)
2. Sample more data points (Central Limit Theorem, we converge in $\sqrt{n}$)
3. Median: biased but has lower variance if $\sigma$ is high (complex to prove but intuition works: people always told you that the median is less sensitive to outliers, well that’s what it means!)

Problem 6

Your friend is doing a project in genetics. The goal is to identify major cell populations from a data set where each row represents a cell and each column is a measurement of gene expression level. Your friend uses a clustering algorithm on an unlabeled dataset and claims that they have selected the best possible number of clusters by cross-validation. Critique this claim.

How can you perform cross-validation on an unlabeled dataset?

Problem 7

The following graph plots the error distribution of three different models. Comment each model in terms of bias-variance
The three figures above show the difference between bias-variance in terms of prediction. Model 1 has high bias but low variance. Model 2 has low bias and low variance. Model 3 has low bias but high variance.