CSCI-4/5587: Machine Learning I
Study Guide for Test#3

(Please don't distribute this study guide.
The guide is provided for your study purpose only)

1. With regularization, the RSS equation with minimization target can be expressed as:

$$\underset{\min}{RSS_\lambda(\beta)} = \sum_{i=1}^{N}\left[(\hat{y}(x_i,\beta)-y_i)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right]$$

Derive and show the corresponding equations for

(i) Gradient descent approach as:

$$\beta_j(t+1) = \beta_j(t) + \frac{2\alpha}{N}\left[\sum_{i=1}^{N}\left(y(i) \; - x^T(i)\,\beta\right).\,x(i)_j - \lambda\beta(t)_j\right]$$

(ii) Newton's method for minimization

$$\beta_j(t+1) = \beta_j(t) + \left[\frac{\sum_{i=1}^{N}\left(y(i) - x^T(i)\,\beta\right).x(i)_j - \lambda\,\beta(t)_j}{\sum_{i=1}^{N}\left[x(i)_j.x(i)_j\right] + \lambda}\right]$$

Ans:

(i) We can write the target for gradient descent as:

$$\beta_j(t+1) = \beta_j(t) - \alpha\frac{\partial}{\partial\beta_j}RSS_\lambda(\beta) \qquad \text{(A)}$$

For the term $\frac{\partial}{\partial\beta_j}RSS_\lambda(\beta)$, we can expand it as:

$$\frac{\partial}{\partial\beta_j}RSS_\lambda(\beta) = \frac{\partial}{\partial\beta_j}\sum_{i=1}^{N}\left[\left(y(i)-x^T(i)\,\beta\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right]$$

$$= \sum_{i=i}^{N}\left[2\left(y(i)-x^T(i)\,\beta\right).\frac{\partial}{\partial\beta_j}\left(y(i)-x^T(i)\,\beta\right) + \lambda\frac{\partial}{\partial\beta_j}(\beta_1^2+\beta_1^2+\ldots+\beta_j^2+\ldots)\right]$$

$$= \sum_{i=1}^{N}\left[2\left(y(i)-x^T(i)\,\beta\right).\frac{\partial}{\partial\beta_j}\left(y(i) \; - x(i)_0\beta_0 - x(i)_1\beta_1 - \ldots - x(i)_j\beta_j - \ldots\right) + 2\lambda\beta_j\right]$$

$$= 2\sum_{i=1}^{N}\left[\left(x^T(i)\,\beta - y(i)\right).\,x(i)_j + \lambda\beta_j\right] \qquad \text{(B)}$$

Replacing the result (B) in (A) we get:

$$\beta_j(t+1)=\beta_j(t)+\frac{2\alpha}{N}\left[\sum_{i=1}^{N}\left(y(i)\ -x^T(i)\,\beta\right).\,x(i)_j-\lambda\beta(t)_j\right]$$

<u>(ii) We can write the target for Newton's method as</u>:

$$\beta_j(t+1)=\beta_j(t)-\frac{\dfrac{\partial}{\partial\beta_j}RSS_\lambda(\beta)}{\dfrac{\partial}{\partial\beta_j}\left(\dfrac{\partial}{\partial\beta_j}RSS_\lambda(\beta)\right)} \qquad \text{(C)}$$

For the numerator of the rightmost term, we have the equation (B).

For the denumerator of the rightmost term, we can also start from equation (B), as:

$$\frac{\partial}{\partial\beta_j}\left(\frac{\partial}{\partial\beta_j}RSS_\lambda(\beta)\right)\ =\ 2\frac{\partial}{\partial\beta_j}\left(\sum_{i=1}^{N}\left[\left(x^T(i)\,\beta-y(i)\right).\,x(i)_j+\lambda\beta_j\right]\right)$$

$$=\ 2\frac{\partial}{\partial\beta_j}\sum_{i=1}^{N}\left[\left(\left(x(i)_0\,\beta_0+x(i)_1\beta_1+\ldots+x(i)_j\,\beta_j+\ldots-y(i)\right).\,x(i)_j\right)+\lambda\beta_j\right]$$

$$=\ 2\sum_{i=1}^{N}\left[\left(x(i)_j\ .\,x(i)_j\right)+\lambda\right] \qquad \text{(D)}$$

Integrating the results from Equation (B) and (D) into (C), we get:

$$\beta_j(t+1)=\beta_j(t)+\left[\frac{\sum_{i=1}^{N}\left(y(i)-x^T(i)\,\beta\right).x(i)_j-\lambda\,\beta(t)_j}{\sum_{i=1}^{N}\left[x(i)_j.x(i)_j\right]+\lambda}\right]$$

2. Show that the parameter β is shrinking, given the gradient descent approach for regularization is:

$$\beta_j(t+1)=\beta_j(t)+\frac{2\alpha}{N}\left[\sum_{i=1}^{N}\left(y(i)\ -x^T(i)\,\beta\right).\,x(i)_j-\lambda\beta(t)_j\right]$$

Ans: It is given that,

$$\beta_j(t+1)=\beta_j(t)+\frac{2\alpha}{N}\left[\sum_{i=1}^{N}\left(y(i)\ -x^T(i)\,\beta\right).\,x(i)_j-\lambda\beta(t)_j\right]$$

And by rearranging we can rewrite:

$$\beta_j(t+1)=\beta_j(t)-\frac{2\alpha}{N}.\lambda\beta(t)_j+\frac{2\alpha}{N}\left[\sum_{i=1}^{N}\left(y(i)\ -x^T(i)\,\beta\right).\,x(i)_j\right]$$

$$=> \beta_j(t+1) = \beta_j(t)(1-\frac{2\alpha\lambda}{N}) + \frac{2\alpha}{N}\left[\sum_{i=1}^{N} \left(y(i) \;- x^T(i)\,\beta\right).\, x(i)_j\right]$$

We see the $\beta_j(t)$ is actually shrinking due to the term $(1-\frac{2\alpha\lambda}{N})$, which is < 1 because α, λ and N are positive quantities.

3. Write down the steps for hold-out cross-validation or simple cross-validation for model selection.

Ans:

1. Randomly split data (D): D_{train} and $D_{holdout}$ (Say, 25-30% of the data).
2. For $\forall i$, train model M_i using D_{train} and get the corresponding P_i (ith predictor).
3. For $\forall i$, compute the error E_i of P_i using $D_{holdout}$.
4. Pick the predictor where the error is the lowest.

4. Write down the steps for k-fold cross-validation for model selection.

Ans:

1. Randomly split data (D) into k disjoint subsets as: $D_1, D_2, ..., D_k$.
2. For $\forall i$, model M_i is evaluated as:
```
    For j =1 to k:
        Train the $M_i$ using data: ( $D - D_j$ ) and get the predictor $P_{ij}$
        Test $P_{ij}$ using $D_j$ and get the error $E_{ij}$.
    EndFor j

    $E_i$ = average of $E_{ij}$ for $\forall j$ , which is the generalized error of $M_i$.
```
3. Pick the best model M_i having the lowest generalized error E_i.
4. Retrain $M_{i = best}$ using full dataset D.

5. Write down the steps for predicting the best values for the regularization parameter λ .

Ans:

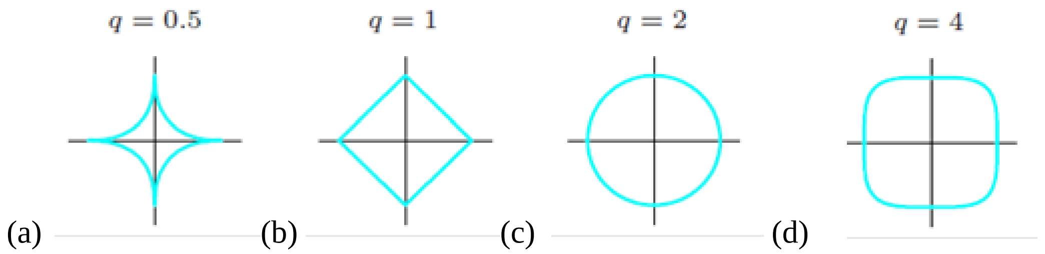
1. Take a range of values with regular intervals: {0, …, R} of real or integer numbers for λ .
2. Evaluate the performance for each of the value of λ using cross-validation approach for example.
3. Pick the best value of λ for which the error was lowest.

6. RSS (β) in terms of more general regularizer is given as:

$$RSS(\beta) = \sum_{i=1}^{N} \{(\hat{y}(x_i, \beta) - y_i)^2\} + \lambda \sum_{j=1}^{p} |\beta_j|^q$$

Draw the contours of the regularization term when q = (a) 0.5, (b) 1, (c) 2 and (d) 4.

Ans:



7. Explain the "no free lunch theorem."

Ans:

Much of machine learning is concerned with devising different models and algorithms to fit them. We can use methods such as cross-validation to empirically choose the best method for our problem. However, there is no universally best model — this is sometimes called the no free lunch theorem (Wolpert 1997). This is because a set of assumptions that work well in one domain may work poorly in another.

8. Define the following terms: (a) accuracy, (b) sensitivity, (c) specificity, (d) precision, (e) balanced accuracy, and (f) MCC.

Ans: Here, in all cases, in all cases, *TP* stands for True Positive, *TN* stands for True Negative, *FP* stands for Fales Positive, and *FN* stands for False Negative.

(a) ***Accuracy*** is then defined as the sum of the number of true positives and true negatives divided by the total number of examples (where, # indicates 'number of,' and TP stands for True Positive, etc.):

$$\boldsymbol{Accuracy} = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ prediction\ made} = \frac{TP + TN}{TP + FP + TN + FN}$$

(b) ***Sensitivity*** (also known as the ***true positive rate***, ***recall***, ***probability of detection, hit rate***) is the ratio of positive data points that are correctly predicted as positive, with respect to all positive data points.

$$Sensitivity = \frac{TP}{TP + FN}$$

(c) ***Specificity*** (also known as the ***true negative rate***, ***selectivity***) is the ratio of the negative data points that are correctly predicted as negative, with respect to all negative data points.

$$Specificity = \frac{TN}{TN + FP}$$

(d) ***Precision*** (also known as the ***positive predictive value***) is the number of correct positive results divided by the number of positive results predicted by the classifier. That is, precision is defined as the accuracy of the judgment.

$$Precision = \frac{TP}{TP + FP}$$

(e) ***Balanced Accuracy***: In the case of imbalanced data, we can compute the ***balanced accuracy*** as the arithmetic mean of sensitivity and specificity.

$$Balanced\ Accuracy = \frac{(Sensitivity + Specificity)}{2} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right)$$

(f) MCC stands for Matthew's Correlation Coefficient. In the case of imbalanced data, we can compute MCC. MCC is computed as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

9. Suppose you are using Polynomial Regression. You plot the learning curves, and you notice that there is a large gap between the training error and the validation error. (a) What is happening? (b) What are the three ways to solve this?

Ans:
(a) If the validation error is much higher than the training error, this is likely because your model is overfitting the training set.

(b) To fix the problem -
i) One way to would be to reduce the polynomial degree: a model with fewer degrees of freedom is less likely to overfit.
ii) Another thing we can try is to regularize the model—for example, by adding an ℓ_2 penalty (Ridge) or an ℓ_1 penalty (Lasso) to the cost function. This will also reduce the degrees of freedom of the model.
iii) Lastly, we can try to increase the size of the training set.

10. Suppose you are using Ridge Regression, and you notice that the training error and the validation error are almost equal and fairly high. (a) Would you say that the model suffers from high bias or high variance? (b) Should you increase the regularization hyperparameter *α* or reduce it?

Ans:
(a) If both the training error and the validation error are almost equal and fairly high, the model is likely underfitting the training set, which means it has high bias.

(b) You should try reducing the regularization hyperparameter λ (lambda).

11. Why would you want to use: (a) Ridge Regression instead of plain Linear Regression (i.e., without any regularization)? (b) Lasso instead of Ridge Regression? (c) Elastic Net instead of Lasso?

Ans:
(a) A model with some regularization typically performs better than a model without any regularization, so we should generally prefer Ridge Regression over plain Linear Regression.

(b) Lasso Regression uses an ℓ_1 penalty, which tends to push the weights down to exactly zero. This leads to sparse models, where all weights are zero except for the most important weights. This is a way to perform feature selection automatically, which is good if we suspect that only a few features actually matter. When we are not sure, we should prefer Ridge Regression.

(c) Elastic Net is generally preferred over Lasso since Lasso may behave erratically in some cases, such as when several features are strongly correlated or when there are more features than training instances. However, it does add an extra hyperparameter to tune. If we want Lasso without the erratic behavior, we can just use Elastic Net with the appropriate value of the mix ratio *r* in the cost or error function,

$$\text{Cost}(\beta) = \text{MSE}\ (\beta) + \frac{1-r}{2} \times \text{Ridge} + \text{r} \times \text{Lasso}.$$

12. Explain the terms: True-Positive (TP), True-Negative (TN), False-Positive (FP), and False-Negative, describing a context.

Ans:
Assume a new test method is applied to screen people for cancer. The test results can be positive (have cancer) or negative (do not have cancer). The prediction or test results for

each subject may or may not match the subject's actual status. In this context, the terminologies –

- True Positive (TP) refers to the situation, where subjects having cancers are predicted to have cancer;
- True Negative (TN) refers to the situation, where subjects do not have cancer are predicted to have no cancer;
- False Positive (FP) refers to the situation, where subjects do not have cancer but are predicted to have cancer;
- False Negative (FN) refers to the situation, where subjects have cancer but are predicted to have no cancer.

13. Compare the parametric and the non-parametric models with their properties and one example each.

Ans:

Parametric Model	Non-Parametric Model
1. The parametric model has a fixed number of parameters	1. The non-parametric model's parameters grow with the amount of training data.
2. Parametric models have the advantage of often being faster to use but the disadvantage of making stronger assumptions about the nature of the data distributions.	2. Non-parametric models are more flexible but often computationally intractable for large datasets.
3. Example: A least-square approach for the linear model, where we predict the number of parameters βs, equal to the number of features or columns of the given dataset.	3. *K nearest neighbor* (**KNN**) classifier is a simple example of a non-parametric classifier. KNN simply "looks at" the K points in the training set that are nearest to the test input *x*, counts how many members of each class are in this set, and returns that empirical fraction as the estimate.

14. Describe ROC curve, AUC, and the steps to generate it using Figure.

Ans: The Receiver Operator Characteristic curve is called the ROC curve in short. ROC curve is a plot of the percentage of true positives on the *y*-axis against false positives on the *x*-axis; an example is shown in Figure 1 below:

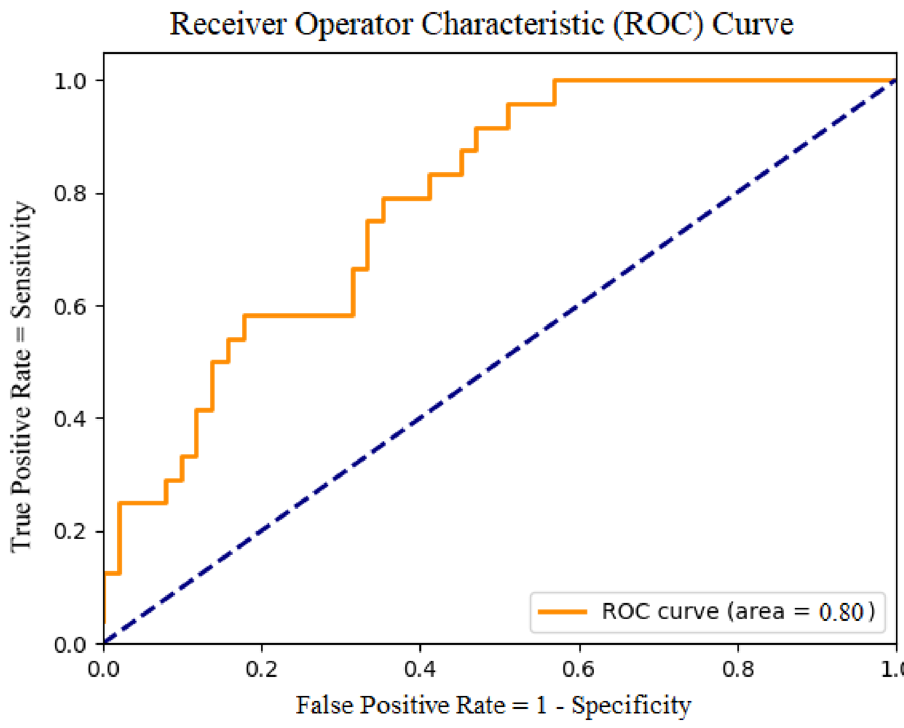


Figure: An example of a ROC curve. The diagonal line represents exactly a chance, so anything above the line is better than chance, and the further from the line, the better.

To draw the ROC curve, TPR and FPR are both computed at threshold values such as (0.00, 0.02, 0.04, …., 1.00) and a graph is drawn (see Figure 1) . The AUC is the area under the curve of the plot. AUC close to or equal to 1.0 indicates the best performance.

A single run of a classifier produces a single point on the ROC plot, and a perfect classifier would be a point at (0, 1) (100% true positives, 0% false positives), while the anti-classifier that got everything wrong would be at (1,0); so the closer to the top-left-hand corner the result of a classifier is, the better the classifier has performed. Any classifier that sits on the diagonal line from (0,0) to (1,1) behaves precisely at the chance level (if the positive and negative classes are equally common). So, presumably, a lot of learning effort is wasted since a fair coin would do just as well.

--- x ---