Author: Devin Kreuzer
Based on *An Introduction to Statistical Learning* by James, Witten, Hastie and Tibshirani.

1 Metrics

Recall:

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

Interpretation: We want very few False Negatives. We don't want to miss any positives. Lower confidence on positives, but more likely to catch all of them.

Precision:

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

Interpretation: We want very few False Positives. We want to be sure about our positives. High confidence on our positives, but may not have caught all of them.

2 Statistical Learning

2.1 Parametric Models

- Advantage is fewer parameters to learn and model interpretability.
- Disadvantage is the assumption itself: fixed model.

2.2 Non-parametric Models

- Advantage is potential to accurately fit wider range of shapes for f.
- Disadvantage is model needs for more training instances to be able to estimate f and loss of model interpretability.

2.3 Bias, Variance, Model Complexity (from ELS)

- We assume that $Y = f(x) + \epsilon$ where f represents the best possible learned model. There is noise in the learned model $\hat{f}$ coming from the set T it is trained on. We are interested in models $\hat{f}$ that are close to f and models $\hat{f}$ that do not vary drastically in response to the specific training set T it is trained on. This is the gist of bias-variance trade-off. Mathematically:

$$\begin{aligned} Err(x) &= \mathbb{E}[(Y - \hat{f}(x))^2 | X = x] = \mathbb{E}[(f(x) + \epsilon - \hat{f}(x))^2 | X = x] \\ &= (\mathbb{E}[\hat{f}(x)] - f(x))^2 + \mathbb{E}[\hat{f}(x) - \mathbb{E}[\hat{f}(x)]]^2 + \sigma^2 \end{aligned}$$

  The first term can be thought of as how much is the expected model (stochasticity comes from training on different training sets) differing from the best possible model. The second term can be thought of as how sensitive/robust is my model to fluctuations in sets it is trained on.

3 Model Assessment and Selection

3.1 Cross-Validation

- Given models $f(x, \alpha)$ where x denotes training set and α are hyperparameters, let $f^{-k}(x, \alpha)$ denote the αth model trained without the kth part. We select the final chosen model $f(x, \hat{\alpha})$ fit on the whole training set based on the minimum $\hat{\alpha}$ which minimizes:

$$CV(f, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L(y_1, f^{-k(i)}(x_i))$$

  This more effectively estimates the average error $\text{Err} = \mathbb{E}[L(Y, f(X))]$

- Selecting $K = N$ most effectively estimates the $\mathbb{E}[L(Y, f(X))]$, but the best estimator can have very high variance due to training on such similar sets (will tend to overfit training data, still). Also has higher computational burden.

- Selecting low K will produce an estimator with a lower variance, but potentially higher bias model, since the model was trained on much fewer data points.

- K typically depends on the learning curve. If the learning curve has a high slope at the given training size for K, cross-validation will overestimate prediction error (higher MSE, lower AUC).

- Model selection without nested CV uses the same data to tune model parameters and evaluate model performance. Information may thus "leak" into the model and overfit the data.

- Use nested cross validation to get better estimate of true prediction accuracy, but use cross-validation to actually find the final model.

4 Loss functions

4.1 Cross-Entropy Loss/Negative log-likelihood/Maximum Likelihood

- Straight-forward:

$$\sum_i -(y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

  The intuition is to bring the probabilities of belonging to a class as close to 1 as possible. As p_i approach 1, the loss function converges to zero. Using log has a large advantage to counteract the saturation of large absolute values output by activation functions (vanishing gradient problem).

5 Linear Regression

5.1 Multivariate Regression

- Can fit β analytically or with gradient descent.
- $\text{Var}(\hat{\beta}) = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X})^{-1}\sigma^2$, where σ^2 is the variance of y_i which can be estimated with $\hat{\sigma}^2 = \frac{1}{N-p-1} \sum_i (y_i - \hat{y_i})^2$

- To test hypothesis of specific coefficient β_j, compute Z-score:

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{v_j}}$$

  where v_j is the jth diagonal element of $(\mathbf{X^T X})^{-1}$

- We assume z is distributed as t_{N-p-1} (t distribution with $N-p-1$ degrees of freedom). However, if σ is known instead of estimated, z would have a standard normal distribution. Compute p-value to reject or accept null hypothesis $\beta_j = 0$

- For computing significance of groups of coefficients simultaneously, compute F statistic:

$$F = \frac{(RSS_0 - RSS_1)/(p_1 - p_0)}{RSS_1/(N-p-1)}$$

  where RSS_1 is RSS for model with more features (p_1+1) and RSS_0 is RSS for model with less features (p_0+1). Under the null hypothesis that the smaller RSS_0 model is correct, F follows an $F_{p_1-p_0,N-p-1}$ distribution.

- To obtain 95% confidence intervals around β_j, we can report

$$(\hat{\beta}_j - 1.96\sqrt{v_j}\hat{\sigma}, \hat{\beta}_j + 1.96\sqrt{v_j}\hat{\sigma})$$

  where we cn replace 1.96 for varying confidence intervals.

5.2 Ridge Regression

- When using many correlated variables in a linear regression, coefficients are poorly determined and exhibit high variances.

- Ridge regression: need to standardize inputs before solving.

- The solution for β_{ridge} is given by

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

  which actually renders the inversion nonsingular (invertible) if $X^T X$ is singular (noninvertible).

- Taking the SVD of $\mathbf{X} = \mathbf{UDV^T}$, we can substitute this into the ridge regression solutions and observe the following:

$$\begin{aligned}\mathbf{X}\beta &= \mathbf{X}(\mathbf{X^T X} + \lambda \mathbf{I})^{-1}\mathbf{X^T y} \\ &= \mathbf{UD}(\mathbf{D^2} + \lambda \mathbf{I})^{-1}\mathbf{DU^T y} \\ &= \sum_{\mathbf{j=1}}^{\mathbf{p}} \mathbf{u_j} \frac{\mathbf{d_j^2}}{\mathbf{d_j^2} + \lambda} \mathbf{u_j^T y}\end{aligned}$$

  Ridge regression computes the coordinates of y wrpt to orthonormal basis of U, and then shrinks these coordinates by a factor of $\frac{d_j^2}{d_j^2+\lambda}$. u_1 is the normalized first principal components, and thus we apply the least amount of penalization in this direction.

$$\text{Var}[\mathbf{X}v_1] = \text{Var}[\mathbf{u_1}d_1] = \frac{d_1^2}{N}$$

5.3 Partial Least Squares

- Find linear combinations of feature space as so:

$$\mathbf{z_1} = \sum_j \langle \mathbf{x_j}, \mathbf{y} \rangle \mathbf{x_j}$$

  inputs are weighted by their univariate strength of $\mathbf{y}$. We then regress $\mathbf{y}$ onto this direction to find the coefficient, and repeat procedure by orthogonalizing all x with z_1 repeatedly.

6 Tree-based methods

6.1 Regression

- Predict mean value of region belonging to.
- Make splits on recursive binary splitting: select **feature and threshold** which minimize RSS: squared difference between values and mean value in their region:

$$\sum_j \sum_{i \in R_j} (y_i - \hat{y}_{Rj})^2$$

- Prune with Cost Complexity Pruning/Weakest Link Pruning: minimize RSS with added $|T|$ term (number of leaves/regions):

$$\sum_j \sum_{i \in R_j} (y_i - \hat{y}_{Rj})^2 + \alpha |T|$$

  which generates new pruned trees for varying α

6.2 Classification

- Predict most occurring class in region.
- Two common criteria for splitting: Gini index (G) and Entropy (D). Let p_{mk} denote proportion of observations from class k in region m:

$$G = \sum_k p_{mk}(1 - p_{mk})$$

$$D = -\sum_k p_{mk} \log p_{mk}$$

  which both approach 0 for p_k approaching 0 or 1.

6.3 Simple Trees vs. Linear Models

- Trees can only generate decision boundaries orthogonal to input space. The form of the decision boundary is important.
- Trees are generally very interpretable.
- Trees are mostly inferior for their lack of **robustness (high variance)**: small changes in input can lead to large changes in output.

6.4 Bagging/Bootstrap Aggregation

- Bagging **uses** Bootstrapping!
- Train on B distinct bootstrapped training sets and average the predicted result from each model.
- Used to **reduce variance/increase robustness** of trees (without pruning)
- **Out of Bag Prediction** Each model has an expected $\frac{2}{3}$ of the training samples in it (from resampling). Predict i-th variable using the expected $\frac{B}{3}$ models that are not including it.

6.5 Variable importance

- For single tree models, just see which features were first used in split.
- In bagging regression trees, record total amount that the RSS is decreased due to splits using the given predictor, averaged over B trees. Large value indicates important predictor.
- In bagging classification trees, sum total amount of decrease in Gini index, averaged over B trees.

6.6 Random Forests

- Same as bagging, but also only consider a **random sample of predictors** for each split. Typical sample size is $\sqrt{p}$.
- Rationale: induce even more variance across the B different trees. Random Forests reduce to Bagging if $m = p$ samples are chosen.

6.7 Boosting

- In bagging, trees are built separately. In boosting, trees are grown **sequentially**. Goal is to slowly improve model to predict in regions it is not predicting well.
- Instead of fitting to outcomes Y, we fit small trees to the residuals output by previous fitted model and add this model to the function.
- Three tuning parameters: number of trees B, learning rate λ and tree depths d.
- Output of model becomes

$$\hat{y} = \lambda f_1(x) + \lambda f_2(x) + ... + \lambda f_B(x)$$

7 Support Vector Machines

7.1 Maximal Margin Classifier/Hard Margin Classifier

- For linear decision boundaries.
- Solve following constrained optimization problem:

$$\max M$$
$$\sum_j \beta_j^2 = 1$$
$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}) \geq M$$

  Intuition: force norm of β to be 1 to maintain notion of distance between hyperplane and x.

- Prone to overfitting (lacks robustness). Small changes in input near the boundary can completely modify model.
- Unique property of MMC: completely robust to some observations (non-support vectors).

7.2 Support Vector Classifier/Soft Margin Classifier

- For linear decision boundaries.
- Allow some mistakes to increase robustness. Less overfitting:

$$\max M$$
$$\sum_j \beta_j^2 = 1$$
$$y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$
$$\epsilon \geq 0, \sum_i \epsilon_i \leq C$$

- ϵ_i have important function: they tell us where the variable i is located. If $\epsilon_i = 0$, i is on the right side of margin. If $\epsilon > 0$, it is within the margin. If $\epsilon > 1$, it is on the wrong side of the hyperplane.
- C is a hyperparameter determining the budget of ϵ_i. Controls bias/variance trade-off.
- Important concept: only observations with $\epsilon > 0$ affect hyperplane (support vectors). High C generates more support vectors (less sensitivity).
- Unique property of SVC: completely robust to some observations (non-support vectors).
- Classifying function becomes

$$f(x) = \beta_0 + \sum_i \alpha_i \langle x, x_i \rangle$$

  Where $\alpha_i = 0$ for non-support vectors.

7.3 Support Vector Machines

- Extension of SVC using kernels to extend feature space. Linear in transformed space $\longrightarrow$ non-linear in original space.
- SVM combines SVC with non-linear kernel

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i)$$

- Example of polynomial kernel:

$$K(x_i, x_j) = (1 + \sum_k x_{ik} x_{jk})^d$$

- Example of radial kernel:

$$K(x_i, x_j) = \exp(-\gamma \sum_k (x_{ik} - x_{jk})^2$$

- Kernels have large computational advantage over expanding features into desired feature space. Only need to compute constant $\binom{n}{2}$ kernels, no matter dimensionality of feature space.
- Can be extended to multiclass by using one-vs-all approach.

8 Convolutional Neural Networks

- Convolution is an extremely efficient way of describing transformations that apply the same linear transformation of a small local region across the entire input.
- **Sparse connectivity**: Kernels are much smaller than the input.
- **Parameter sharing**: We learn one set of smaller parameters that is applied across the entire input.
- **Equivariance to translation**: If the input changes, the output changes in the same way. More formally, function f is equivariant to function g if $f(g(x)) = g(f(x))$. Simple example on page 329 of Deep Learning: shifting the input dog and then applying the convolution is the same as applying the convolution and then shifting the output.
- A typical CNN layer applies a Convolution Stage (Affine transformation), a Detector Stage (Nonlinearity) and a Pooling stage.

8.1 Pooling

- Pooling functions are used to describe some summary statistic of nearby outputs. **Pooling is used to learn invariances (spatial, rotational etc.)**
- *Invariance to local translation can be a useful property if we care more about whether some feature is present rather than where it is.*
- Can improve computation even more drastically be used fewer pooling units than detector units.

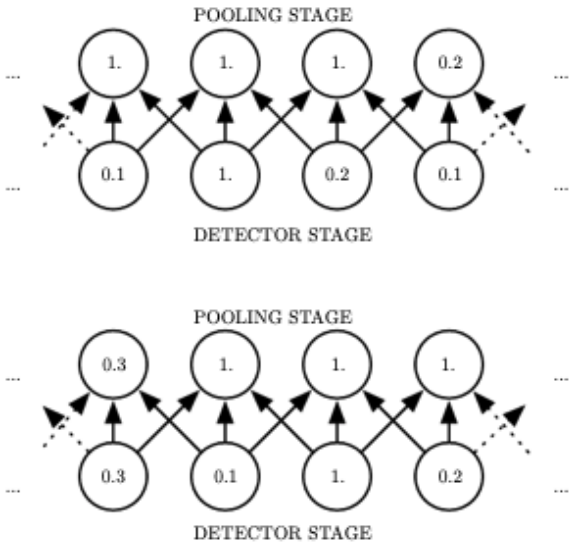

Figure 1: Every value in the bottom row has changed, but only half the values in the top row have changed. This details how pooling can be used to summarize the presence of something regardless of its location.