DSCI 552 Cheat Sheet: Katie Foss

Lesson 6: Tree Based Methods

Decision Trees: Used for classification and regression. Regression: mean of obs in terminal node regions. Regions should be convex. Prone to overfitting. ↑ # of trees in model, ↑ accuracy and ↓ interpretation

Greedily Build a Decision Tree:

1. Per iteration we are looking to split one region.
2. Within each established region we find the best cutpoint (s) per predictor and choose the (predictor, cutpoint) pair that gives the lowest RSS for this iteration.
3. $RSS_j = \sum_{i=1}^{n}(y_i - \hat{y}_{R_j})^2$
4. $\hat{y}_{R_j}$ = mean response for training observations in the jth box
5. Uses some stopping criteria (no region > 5 observations)

Cost Complexity Pruning (Weakest Link):

- Find α that minimizes the following equation, where $|T|$=# terminal nodes and each α is tied to a subtree.
- $\sum_{m=1}^{|T|}\sum_{i:x_i \in R_m}(y_i - \hat{y}_{R_m})^2 + \alpha|T|$
- ↑ tree, ↑ variance, ↓ bias. Penalize large trees.

Classification Trees: Use majority polling instead of average at terminal nodes. Cannot use RSS.

RSS alternatives where $\hat{p}_{mk} = \frac{train\_obs\_in\_class\_k\_in\_m}{train\_obs\_in\_region\_m}$:

- Classification Error Rate: $E_m = 1 - max(\hat{p}_{mk})$
- Gini index: $G_m = \sum_{k=1}^{K}\hat{p}_{mk}(1-\hat{p}_{mk})$
- Cross-Entropy: $D_m = -\sum_{k=1}^{K}\hat{p}_{mk}log(\hat{p}_{mk})$

Gini index is measure of node purity, ↓ gini means most observations belong to a single class.

Weighted impurity: Choose weights to be the fraction of data points in each region.

Bagging (Bootstrap aggregation): ↓ *variance*

Variance of $Z_i = \sigma^2$ while variance of mean of n observations is $\bar{Z} = \frac{\sigma^2}{n}$

- Bootstrap B datasets from training data (with repetition).
- Take average (regression) of all predictions ($\frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$) or majority vote (classification)
- OOB (Test) Error Estimation: B/3 trees did not use i^{th} observation for training. Average the predictions of the trees that didn't use the observation for training.
- Perfect correlation ruins variance reduction

Lesson 6: Tree Based Methods Continued

Random Forest: Decorrelate trees to ensure variance reduction. Each iteration choose a random selection of m predictors. Typically $m = p^{\frac{1}{2}}$.

Boosting: Fit small trees to the residuals, residuals of previous tree used to train next tree (sequentially). Can get overfit, shrinkage parameter λ slows the learning and choose number of trees B with cv to help avoid over overfit.

Boosted model: $\hat{f}(x) = \sum_{b=1}^{B}\lambda\hat{f}^b(x)$

Find feature importance: $\frac{\overline{\triangle RSS_i}}{max(\overline{\triangle RSS})}$

Lesson 7: SVM

SVM: Find a hyperplane (hp) that separates the classes. Hyperplane $p-1$ dimensions. The normal vector is orthogonal to the hyperplane.

Hyperplane: $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0$

Normal vector: $\beta = (\beta_1, \beta_2, ..., \beta_p)$

Maximal Margin Classifier: Choose betas to maximize M in $y_i(\beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip}) \geq M$ for all $i = 1, .., N$ subject to constraint $\sum_{j=1}^{p}\beta_j^2 = 1$

Slack Variable ϵ_i: Choose betas and epsilons to maximize M in $y_i(\beta_0+\beta_1 x_{i1}+...+\beta_p x_{ip}) \geq M(1-\epsilon_i)$ subject to same constraint above and $\epsilon_i \geq 0, \sum_{i=1}^{n}\epsilon_i \leq C$

- $\epsilon = 0$, i correct side of margin
- $\epsilon_i > 0$ i wrong side of margin, violated margin
- $\epsilon_i > 1$ i wrong side of the hyperplane

For $C > 0$ no more than C observations be on wrong side of hyperplane. $\downarrow C, \uparrow variance, \downarrow bias$

Support Vectors: Ony observations that lie on the margin or violate the margin are support vectors and affect the hyperplane.

SVM similar to logistic regression because both not affected by observations far from the decision boundary. Unlike LDA.

Kernels: Linear Classifier

$f(x) = \beta_0 + \sum_{i \in S}\alpha_i\langle x, x_i\rangle$ where $\sum_{i \in S}\alpha_i = 0$

$\beta_0 = \frac{1}{y_i} - \sum_{i \in S}\alpha\langle x_i, x_j\rangle$

Kernels are generalizations of inner products, replace inner product with kernel. RBF = Radial = Guassian.

- Polynomial: $K(x_i, x_i') = (1 + \sum_{j=1}^{p} x_{ij}x_{i'j})^d$
- RBF: $exp(-\gamma\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2)$

Lesson 7: SVM Continued...

Multi-Class Classification:

One versus All: Fit K different 2-class classifiers. classify to model with largest score.

One versus One: Fit all $\binom{K}{2}$ pairwise classifiers, choose class that wins most competitions.

Multi-Label:

Problem Transformation: Transform multi-label into single lable

- Binary Relevance: Build m classifiers, one for each label. (1,0) label
- Classifier Chains: Same as binary relevance, train m classifiers, but they are chained to include previous labels.
- Label Powerset: Turn labels into single multiclass and train labels as multi-class. $2^{\#label}$ classes for binary labels

Evaluation Metrics:

- Exact Match: All labels must match exactly.
- Hamming Loss: $\frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L} I(\hat{y}_{ij} \neq y_{ij})$

Estimate SVM probability: $\frac{1}{1+exp(-y_i f(x_i))}$

VC dimension: max num points arranged so points can be given 0 or 1 and still linearly separated (known as shattered). Linear: Dimension + 1, SVM with RBF and decision tree infinite.

Lesson 8: Unsupervised Learning

K-means clustering: Parition observations into k clusters. Minimize the within-cluster variation of each cluster.

$WCV(C_k) = \frac{1}{C_k}\sum_{i,i' \in C_k}\sum_{j=1}^{p}(x_{ij} - x_{i'j})^2$

1. Randomly assign observations to clusters.
2. Iterate until assignments stop changing:
   (a) Compute cluster centers (average)
   (b) Assign observations to closest cluster center (euclidean distance)

euclidean: $d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$

K-Medoids Clustering: Center of cluster is datapoint with the smallest dissimilarity with other datapoint. Closest point to centroid.

Hierarchical Clustering:

1. Calculate pairwise dissimilary (euclidean)
2. Start with each point is own cluster
3. Merge least dissimilar clusters
4. Repeat until all points in 1 cluster

Lesson 8: Unsupervised Learning Continued...

Dissimilarity between clusters (Linkage):

- Complete: Max inter-cluster dissimilarity
- Single: Minimal inter-cluster dissimilarity
- Average: Avg of all dissimilarities in a cluster.
- Centroid: Dissimilarity between the centroid between two clusters (can cause inverstions)

Between-cluster variation: $B = \sum_{k=1}^{K} n_k \|\bar{X}_i - \bar{X}_k\|_2^2$

CH index: $CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$

$Gap(K) = log(W_{unif}(K)) - log(W(K))$, choose smallest k where $Gap(K) \geq Gap(K+1) - s(K+1)$

Silhoutte Score: $s_i = \frac{b_i - a_i}{max(a_i, b_i)}$, -1 ↓, 0 border, 1 ↑

a_i=avg dist between i and points in cluster, b_1=lowest avg dist of i to all points in other clusters.

PCA: Maximiaze sample variance that is uncorrelated with previous PC. Maximize $\phi_{11}, ..., \phi_{p1}$ in $\frac{1}{n}\sum_{i=1}^{n}(\sum_{j=1}^{p}\phi_{j1}x_{ij})^2$ subject to $\sum_{j=1}^{p}\phi_{j1}^2 = 1$// First PC loading vector defines the line in p-dims closest to n observations.

Fisher's LDA: Maximize mean of projected variance of different classes.

PVE of mth PC: $PVE_m = \frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2}$

Lesson 9: Semi-Sup and Active Learning

Self-Training/Yarowsky Algorithm:

1. Train supervised model on labeled data (L)
2. Temp label unlabeled and choose points that confidently classified and add them to current pool of labeled training data.
3. Repeat (in iterations)

Co-Training: S_1 and S_2 are independent sets of features. Each has a classifier (S_1, C_1), (S_2, C_2). Same as Self-Training, each iteration only (S_1, C_1) or (S_2, C_2) used to label data.

Active Learning: Stream-Base: Decide whether to query label of one unlabeled observation at a time. Pool-Based: Rank large unlabeled pool of observations in order of informativeness. Query n from rank.

Query Selection Strategies: Uncertainty Sampling, Query by Committee, Expected Model Change, Expected Error Reduction, Variance Reduction, Density Weighted Mehtods

Passive Learning requires **agreement/confidence**, **Active** requires **disagreement/uncertain**

Lesson 10: Neural Networks

Perceptron:

Activation Function: $f(x) = \begin{cases} \boldsymbol{\beta}^T\mathbf{x} + \beta_0 \geq 0, 1 \\ \boldsymbol{\beta}^T\mathbf{x} + \beta_0 < 0, -1 \end{cases}$

Error: $e(i) = y(i) - f(\boldsymbol{\beta}^T(i)\mathbf{x}(i) + \beta_0(i))$

Update Rule: $\boldsymbol{\beta}(i+1) = \boldsymbol{\beta}(i) + 0.5e(i)\mathbf{x}(i)$

Bias: $\beta_0(i+1) = \beta_0(i) + 0.5e(i)$

Multiclass Perceptron:

Update Rule: $\mathbf{w}(i+1) = \mathbf{w}(i) + \alpha\mathbf{x}(i)\mathbf{e}^T(i)$

Bias: $\mathbf{b^T}(i+1) = \mathbf{b^T}(i) + \alpha\mathbf{e}^T(i)$

Error: $\mathbf{e}(i) = \mathbf{y}(i) - f(\mathbf{W}^T(i)\mathbf{x}(i) + \mathbf{b}(i))$

α commonly 0.5

Multi-layer Perceptron:

Sigmoid: $\frac{e^x}{e^x+1}$, Tahn: $\frac{e^{2x}-1}{e^{2x}+1}$, ReLu: $max(0, x)$

$\frac{\partial}{\partial n}$ Sigmoid: $a^{(m)}(1 - a^{(m)})$, Tahn: $1 - Tanh(x)^2$

Depth: # layers, Width: # of neurons of each layer

i^{th} layer output is $a^{(i)} = f^{(i)}(\mathbf{W}^{(i)^T}\mathbf{a}^{(i-1)} + \mathbf{b}^{(i)})$

Forward Propogation: Output(a) of prev layer is input to next layer. $\mathbf{a}^{(0)} = \mathbf{x}(k)$, $\mathbf{a}^{(j)} = \mathbf{f}^{(j)}(\mathbf{n}^{(j)}$

$\mathbf{n}^{(j)} = (\mathbf{W}^{(j)T}\mathbf{a}^{(j-1)} + \mathbf{b}^{(j)})$

Backpropogation of errors: Method to calculate gradients used to update weights

Objective Func: $J = E[\sum_{i=1}^{S^M}(y_i - a_i^M)^2] \approx \mathbf{e}(k)^T\mathbf{e}(k)$

Update Weights and Bias:

$\mathbf{W}^{(m)}(k+1) = \mathbf{W}^{(m)}(k) - \alpha\mathbf{a}^{(m-1)}\mathbf{s}^{(m)T}$

$\mathbf{b}^{(m)}(k+1) = \mathbf{b}^{(m)T}(k) - \alpha\mathbf{s}^{(m)T}$

Sensitivity:

$S^{(M)} = -2\mathbf{F}'^{(M)}(\mathbf{n}^{(M)})(\mathbf{y\text{-}a}^{(M)})$ (last layer)

$S^{(m)} = \mathbf{F}'^{(m)}(\mathbf{n}^{(m)})(\mathbf{W}^{(m+1)}\mathbf{s}^{(m+1)})$ (hidden layers)

Define $\mathbf{F}'^{(m)}(\mathbf{n}^{(m)})$ as:

$$\begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \dots & 0 \\ 0 & \dot{f}^m(n_2^m) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \dot{f}^m(n_{S^m}^m) \end{bmatrix} \quad \dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}$$

Convolutional Neural Network:

Kernel is applied to input (can add padding), this is called feature map. An activation function is applied to the feature map and max or sum pooling is applied after. This output is then the input to a neural network.

Universal Approximator: Sigmoid in Hidden, Linear output, model any smooth function.

Lesson 11: Hidden Markov Model

DP Solution: Most probable possible path

HMM: Maximize expected number of correct states.

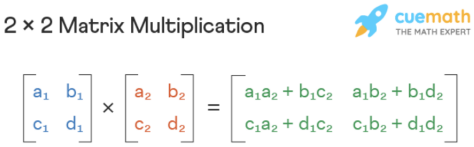
Misc

Probability:

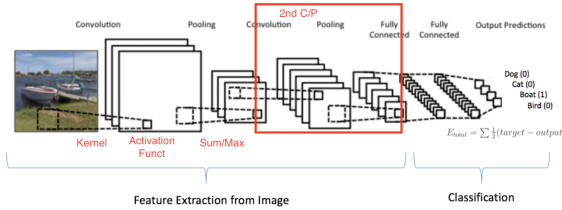
$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$

$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Matrix Multiplication: Multiply elements of each row of first matrix by elements of each column in the second matrix (element by element). $(m \times n) \cdot (n \times p) = (m \times p)$



CNN:



Equation of Circle: (h, k) is circle center

$(x - h)^2 + (y - k)^2 = r^2$

XOR: