ECE M146 Introduction to Machine Learning

Lecture 7 - Spring 2021

Prof. Lara Dolecek

ECE Department, UCLA

Today's Lecture

Recap:

- Linear models

New topics:

- Decision Trees (continuation from last time)
- K nearest neighbors (K-NN)
- Multiclass classification
- Model assessment and selection

Recap: Linear models for classification and regression

- We previously studied parametric methods, such as perceptron, linear regression, and logistic regression.
- These methods are governed by the vector w and specifically how the value $w^T x$ maps to the output.
- Once the vector w is derived, we essentially no longer need the training data.

- Last time we introduced non-parametric methods, and the decision trees in particular.

Today's Lecture

Recap:

- Linear models

New topics:

- Decision Trees (continuation from last time)
- K nearest neighbors (K-NN)
- Multiclass classification
- Model assessment and selection

Decision Trees

- Decision Tree is a greedy modeling method for classification (and regression).
- At each step, split on the most informative attribute.
  - Information Gain.
  - Use Entropy and Conditional Entropy.
- Terminate under given rules.
- At test time, go down the decision tree with the given attributes and declare as the output the label we arrive at.

Let's now look at a bigger example (continuation)

- Our first example was fairly apparent. But in practice we don't always have such clear cut choices.
- Bigger example:

$P(X_1 = T) = \frac{1}{2}$

$P(X_2 = T) = \frac{1}{2}$

X_1	X_2	Y	# occurrences
T	T	T	20
T	F	T	20
F	T	T	10
F	T	F	10
F	F	F	20

- Note that some combinations didn't even appear and note that some are contradictory.

How do we build the decision tree now ?

- First, let's compute the entropy of Y, H(Y):

$P(Y=T) = 5/8$

$P(Y=F) = 3/8$

$H(Y) = -\frac{5}{8}\log\frac{5}{8} - \frac{3}{8}\log\frac{3}{8}$

- Next, compare $IG(Y,X_1)$ vs. $IG(Y,X_2)$:

$IG(Y, X_1) = H(Y) - H(Y|X_1)$

$IG(Y, X_2) = H(Y) - H(Y|X_2)$

choose the one w/ max IG $\Leftrightarrow$
choose the one w/ smallest conditional entropy term

Conditional entropy calculation

- Last time we computed $H(Y|X_1)$ using $H(Y|X_1=T)$ and $H(Y|X_1=F)$.

$$H(Y|X_1) = H(Y|X_1=T)\cdot P(X_1=T) + H(Y|X_1=F)\cdot P(X_1=F)$$

$$0 = H(Y|X_1=T) = -P(Y=T|X_1=T)\cdot \log_2 P(Y=T|X_1=T) - P(Y=F|X_1=T)\cdot \log_2 P(Y=F|X_1=T)$$

$$P(Y=T|X_1=T) = \frac{1/2}{1/2} = 1 \qquad P(Y=F|X_1=T) = 0$$

$$H(Y|X_1=F) = -P(Y=T|X_1=F)\cdot \log_2 P(Y=T|X_1=F) - P(Y=F|X_1=F)\cdot \log_2 P(Y=F|X_1=F)$$

$$P(Y=T|X_1=F) = \frac{1/8}{1/2} = \frac{1}{4} \qquad P(Y=F|X_1=F) = \frac{3}{4}$$

$$H(Y|X_1=F) = \frac{-1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4}$$

$$H(Y|X_1) = 1 - \frac{3}{8}\log 3$$

Conditional entropy calculation

- To compute $H(Y|X_2)$, let's compute both $H(Y|X_2 =T)$ and $H(Y|X_2=F)$ first.

$$H(Y|X_2) = H(Y|X_2=T)\cdot P(X_2=T) + H(Y|X_2=F)\cdot P(X_2=F)$$

$$H(Y|X_2=T) = -P(Y=T|X_2=T)\cdot \log_2 P(Y=T|X_2=T) - P(Y=F|X_2=T)\cdot \log_2 P(Y=F|X_2=T)$$

($P(Y=T|X_2=T) = 3/4$, $P(Y=F|X_2=T) = 1/4$)

$$= -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4}$$

$$= 2 - \frac{3}{4}\log 3$$

$$H(Y|X_2=F) = -P(Y=T|X_2=F)\cdot \log_2 P(Y=T|X_2=F) - P(Y=F|X_2=F)\cdot \log_2 P(Y=F|X_2=F)$$

($P(Y=T|X_2=F) = 1/2$, $P(Y=F|X_2=F) = 1/2$)

$$= 1$$

$$H(Y|X_2) = \frac{1}{2}\left(2 - \frac{3}{4}\log 3\right) + \frac{1}{2}$$

$$= \frac{3}{2} - \frac{3}{8}\log 3$$

Sanity check on calculations

- If the attribute is binary (X), and the label is binary (Y), we can check if the following is satisfied:

$0 \leq H(Y) \leq 1$

$0 \leq H(X) \leq 1$

$0 \leq H(Y|X) \leq 1$

$0 \leq IG(X,Y) \leq 1$

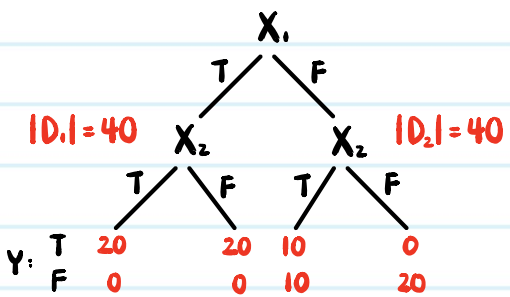
Now, let's build our decision tree

$$IG(X_1, Y) = H(Y) - H(Y|X_1)$$

$$IG(X_2, Y) = H(Y) - H(Y|X_2)$$

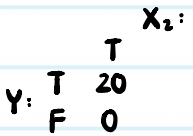
$$H(Y|X_1) < H(Y|X_2)$$

$$\Rightarrow IG(X_1, Y) > IG(X_2, Y)$$

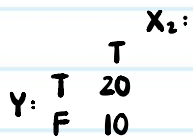


Termination rules

- If a node is pure, i.e., all its examples have the same label y, assign that label to the examples, and terminate.
  - Example:



- If all the examples have the same attribute values, but different labels, assign the one given by the majority, and terminate:
  - Example:



Termination rules

- In case of a tie, either pick the label randomly or declare the label the one governed by the parent and terminate.

- Otherwise, keep going until there is nothing left to query on.

Training Error

- Note that in our last example, we could not get the training error to be zero. Why ?

Due to conflicting examples

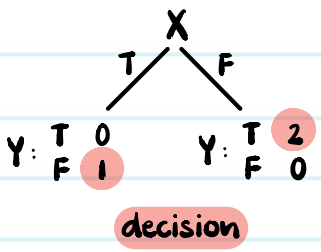
leaf nodes that are not pure

Common issues with decision trees and how to overcome them

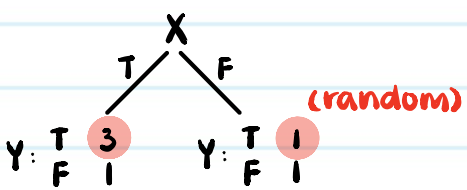
- Decision trees are prone to overfitting.
- E.g. Root node split governs the rest (recall that we are doing a greedy procedure).
- Solution #1: split the data into training set and validation set. Train on the training set, prune using the validation set.
- Example: out of N examples, 0.7*N are the training set, and the remaining 0.3*N are the validation set.
- These two sets are mutually exclusive!
- Solution #2: bagging = bootstrap aggregation (ensemble method)
  - Sampling with replacement.

Training DTs with a validation set

- Example:
- Suppose this is what the training set gives:



- Suppose this what the validation set gives:



6 points, 3+1 are misclassified

suppose we made the decision at x, w/o further branching

2 are misclassified

Training DTs with a validation set

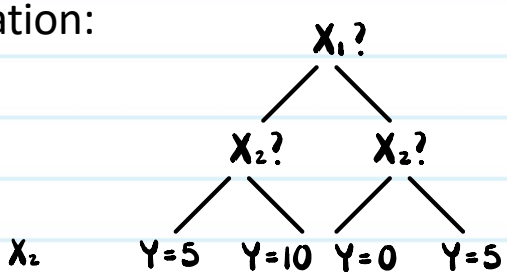
- How many are misclassified under the original rule? 4

- How many are misclassified if we didn't split on x ? 2
  i.e. take majority vote at that point

- Prune the tree ! 2 < 4

Decision trees for regression

- The discussion thus far was focused on the binary classification problem. DTs can also be done for regression.
- Illustration:



- All data points in the given region are assigned the same label.

Summary

- We discussed decision trees, which are used for supervised learning.
- Decision trees are typically interpretable, and as a result, popular in practice.
- A way to build a decision tree is using a greedy approach, combined with statistical (information theoretic) rules for growth.

- Next: k-nearest neighbors.

Today's Lecture

Recap:

- Decision trees and linear models

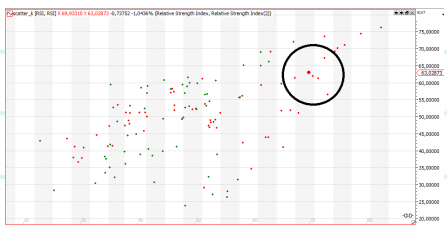
New topics:

- Decision Trees (continuation from last time)
- K nearest neighbors (K-NN)
- Multiclass classification
- Model assessment and selection

K Nearest Neighbors Applications

- Recommender system

- Stock Market predictions



- Image scene classification

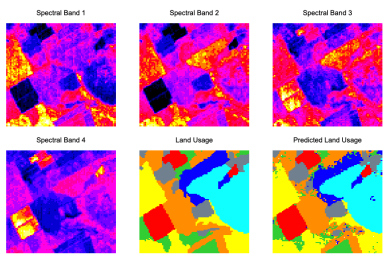


FIGURE 13.6. *The first four panels are LANDSAT images for an agricultural area in four spectral bands, depicted by heatmap shading. The remaining two panels give the actual land usage (color coded) and the predicted land usage using a five-nearest-neighbor rule described in the text.*

T. Hastie et al, "Elements of Statistical Learning"

https://www.knowledgehut.com/blog/data-science/knn-for-machine-learning

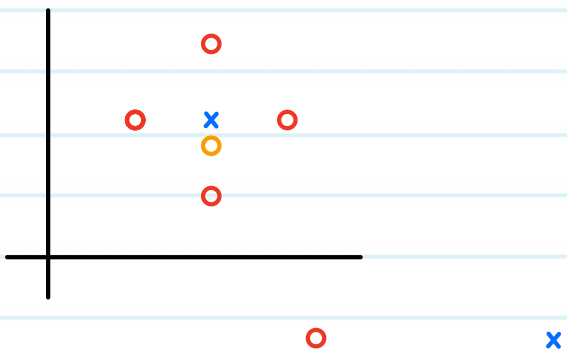
https://www.quanttrader.com/index.php/machine-learning-knn-algorithm/KahlerPhilipp2018

K Nearest Neighbors Algorithm

- Like the decision trees, K-NN is a **non-parametric** method.
- Learning part in K-NN is “easy”: store all the training examples.
- But the testing part is “hard” (computationally more intensive): potentially need to compare the test point against all training points.

Let's consider K-NN for binary classification

- Example with K = 1:



- Would you classify point 'o' as being in Class 1 or in Class 2?
- It depends on what rule we use.

Distance measure

- In K-NN with K=1, we find the closest neighbor in the training set and assign the label of that point as the label of the test point.
- Closest in what sense ?
- It is natural to consider Euclidean distance:

$$d(\vec{x}, \vec{z}) = \sqrt{\sum_{i=1}^{d} (x_i - z_i)^2}$$

$$\vec{x}, \vec{z} \in \mathbb{R}^d$$

At test time

- Suppose that the data is d-dimensional.
- Then, at test time we have:

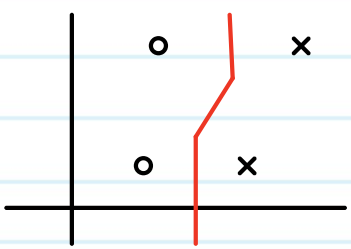
$$\vec{x}^* = \underset{x_j \in \text{training set}}{\operatorname{argmin}} d(\vec{x}_j, \vec{z}) \qquad \text{label}(x^*) = \text{label}(z)$$

- We can avoid taking √ so we just compare the squared norms:

$$\|\vec{x}_j - \vec{z}\|^2 = \sum_{i=1}^{d} (x_{j,i} - z_i)^2$$

Decision boundaries

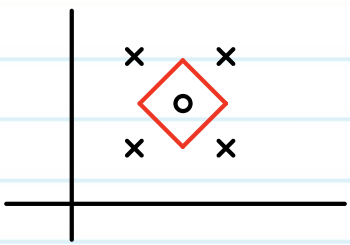
- Decision boundaries are not computed explicitly, i.e., there is no mathematical formula for it.
- For K=1, each decision boundary is equidistant to two closest points from the opposite classes.
- Example:



- These are also called Voronoi regions

Another example

- What about this case ?



- Would you really expect class 1 in the middle of class 2 ? No.
- These situations are typically due to label noise, mislabeled data.
- But, cause overfitting!

Solution for overfitting

many ties

- Increase K, so we look at more than 1 nearest neighbor.
- Why not try K = 2 ?
- Ok, so let's look at 3-NN.

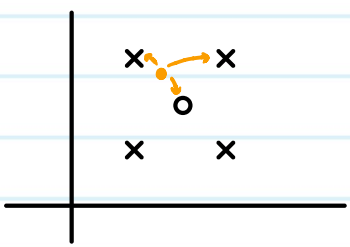
Procedure:

1. For a given test point, compute the distance from it to all training points (N distances).
2. Find 3 closest points.
3. Take the majority vote.

Back to the previous example

- Let's revisit the last example: k = 3

would pick 'x' class



- So if K = 3 is better than K = 1, does that mean K = 103 is better than K = 3 ?

Choice of the parameter K

- For small K, decision boundary is very fine grained; decision regions are squiggly patches (overfitting is a possible issue).
- For large K, decision boundary is smooth; decision regions are large uninterrupted segments (bias is a possible issue).
- As it turns out, 1-NN has misclassification rate at most twice what is achieved by an optimal classifier (mathematical proof) it does make sense to choose smaller K.

Choice of the distance measure

- There are other choices of the distance, that can be more appropriate, depending on the application.

- Hamming distance, e.g., for categorical data:

$$d_H(x, z) = \sum_{i=1}^{d} 1(x_i \neq z_i)$$

- Manhattan distance:

$$d_M(x, z) = \sum_{i=1}^{d} |x_i - z_i|$$

More on the practical issues

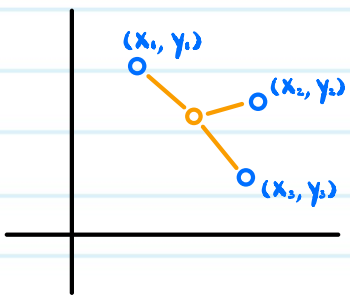
- When implementing K-NN with Euclidean distance, ensure that all dimensions are normalized, so that one dimension does not artificially skew the distance and dominate.

- Check for the presence of “duplicate” attributes, as these too can overly represent a given dimension.

- It is sometimes beneficial to weigh the vote by the distance: inverse

K-NN for regression

- The previous procedure can be easily extended to the regression problem:
- The output label is the (weighted) average of the labels of K nearest neighbors.



$$x_{TEST}$$

$$y_{TEST} = \frac{y_1 + y_2 + y_3}{3}$$

Today's Lecture

Recap:

- Decision trees and linear models

New topics:

- K nearest neighbors (K-NN)
- Multiclass classification
- Model assessment and selection

Multi-class classification

- The methods we saw thus far in the course, we primarily described in the context of the binary classification problems.
- We can extend these approaches to the **multi-class classification** problem as well.

- General techniques for going from binary to multi-class classification:

1. One vs. all (OVA)
2. All vs. all (AVA)

One vs. All in K classes

- For all labeled training points,

  For each class k, $1 \leq k \leq K$,

  Create a binary classifier : class k vs. class not-k

- Example: K=3: There are 3 binary classifiers.
  - Class 1 vs. class not-1 (which is the union of classes 2 and 3)
  - Class 2 vs. class not-2 (which is the union of classes 1 and 3)
  - Class 3 vs. class not-3 (which is the union of classes 1 and 2)

One vs. All in K classes

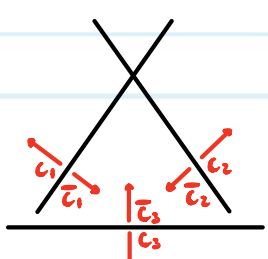
- Advantage: we train K binary classifiers, which we already know how to do.
- Unfortunately, these classifiers are imbalanced in their training data (consequence of the design).

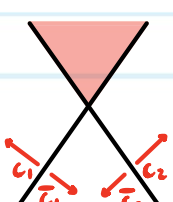
- At test time, pick the class that chose the test point.
- Issue: more than one class chose it. Then the ties are often broken randomly.
- Issue: no class chose this point (no vote).

All vs. All in K classes

- Interpret the problem as the tournament with $\binom{K}{2}$ matches. Each match is a binary classifier.
- Example: K = 3. There are 3 matches: Class 1 vs. class 2; class 2 vs. class 3, class 1 vs. class 3.

- We now partition the data set so that in each match there are only points that are labeled with the match participants.
- Issue: training individual classifiers on a much smaller data set





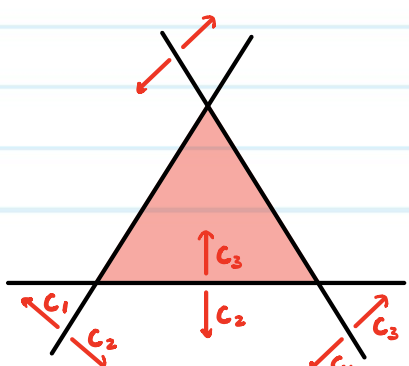
All vs. All in K classes

- At test time, run all $\binom{K}{2}$ classifiers.
- The winner gets +1 point.
- Label of the test point is determined by the class that earned the most points.

- Issue: can also lead to ambiguity, e.g., in K = 3 case each class gets +1 point in each of the three matches.

$k = 3 \quad \binom{3}{2} = 3$

all 3 classes voted one vote for the red region



Today's Lecture

Recap:

- Decision trees and linear models

New topics:

- K nearest neighbors (K-NN)
- Multiclass classification
- Model assessment and selection

How to assess the quality of the learned model ?

1. Use **validation**.
   - Recall that we already discussed this approach in the context of decision trees.
- How ? Split the data. Train on $\mathcal{D}_1$ and validate on $\mathcal{D}_2$.

$D = D_1 \cup D_2$

$D_1 \cap D_2 = \phi$

$|D| = N$

$|D_1| = 0.7N$

$|D_2| = 0.3N$

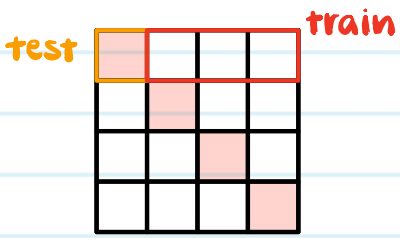
- When we test on $\mathcal{D}_2$ we get the validation error which can be used to estimate the generalization error.

How to assess the quality of the learned model ?

each is now D

2. Use **cross-validation**.

- How ? Train separate models as follows:



average = validation error

model 1 → validation error 1
model 2 → validation error 2
model 3 → validation error 3
model 4 → validation error 4

- If the size of the validation set is 1, this approach is called leave one out (LOO). Note that there are N models in that case. LOO can be done efficiently in the case of linear regression.

How to assess the quality of the learned model ?

2. Use **cross-validation**.

- More generally, we can use K-fold validation.
- There are K models.
- Data is partitioned so that each time different 1/K fraction of data is in the validation set.
- Train each model on its own (K-1)/K fraction of data and validate it on its own validation set (1/K fraction of data).

- Average these K resulting validation errors. This is the overall validation error.

How to assess the quality of the learned model ?

- In both cases, retrain on all available data and discard the smaller models.

Model selection

- Perform model assessment for each choice of the hyperparameter.
- For example, K in K-NN is a hyperparameter.
- Choose the value with the lowest validation error and train on all data.