Day 4

January 12, 2018

Contents

Random Forest and other Ensemble methods

Monte Carlo methods

Many problems do not have closed form solutions, e.g. the mean of the sum of two Negative-Binomial distributed random variables.

Bootstrap

It's physically imposible for us to pull ourselves up from our bootstraps – however, it's viable to use the same data to generate more data! Useful when we don't have enough data for Monte Carlo methods!

Sample from distributions generated from known data, i.e. sample from the *histograms*! This isn't any more complicated than simply sampling from the dataset. However, we must take care to put our samples *back* into the original set else we change the shape of the distribution. Sampling with replacement.

It doesn't matter that we end up with duplicates in the sampled dataset.

Bagging

Bootstrap aggregation

Consider CARTS – a different subset of the data might result in a widely differing tree from that generated from another subset. They're *unstable*, even though the predictions aren't that much different.

The concept is to generate multiple predictors from a lot of bootstrapped datasets and aggregate their predictions. The reason this works is that erroneous answers are typically *equally erroneous*, but those answers that are slightly more correct are correct in consensus!

A downside of this method is that we lose interpretability. A single tree is easy to understand, but how a thousand trees affect the prediction is quite a bit harder.

Out-of-bag cross validation!

There are some samples that are not included in the bootstrapped dataset –typically 1/3– which can be naturally used for cross-validation.

Boosting

Fitting additional predictors to residuals, i.e. the *errors*, from initial predictions. We adjust predictions using a percentage (decided with a learning rate λ) of some known errors.

Parameters:

- Tree number
- Tree depth
- λ value

Random Forests

Bagged trees only have variability in the way that data is selected for the training. Thus, they can still be correlated amongst themselves, limiting their usefulness.

Random forests introduce aditional randomness by **enforcing** the use of different splitting variables by randomly disallowing some percentage (chosen using CV). For example, bagged trees might be constantly misled into choosing a variable like 'age' over some other one, not allowing the model to find some better variable. This is the same idea as the concept of influenced, non-independent voters.

Because of this, each *individual* tree tends to be worse, but the *aggregate* is far better.

We still get OOB Cross-Validation!

Variable importance measurements

Measure whether some variable is important or not.

- Metric 1 Average the total error decrease after a split involving this variable over all trees.
- Metric 2 Shuffle *only the studied variable* and check whether prediction error increases. If it does, it turns out the variable was important!

Advantages

- Very robust under non-informative variables – less variance
- Less prone to overfitting
- No need for pruning!
- Built-in cross validation sets with OOB

DIsadvantages

- Hard to capture additive effects
- Not interpretable

SVM extras

Imbalanced classes

One class occurs far more often than the other, e.g. in fraud detection, medical databses.

Problematic because algorithms work best on even databases, and there's typically poor performance on underrepresented the class.

How do we imporove performance?

- Get more data for underrepresented class!
- Class weighting
  - Penalize more harshly making an error on the smaller class
- Special sampling methods
  - Modify training observations to balance class sizes, tipically by undersampling (+) or oversampling (-). This, however, may throw away important training observations or end up severly overfitting the model.

Classification performance

Machine Learning for Text Data

Non-negative matrix decomposition

Non-negative matrix factorization $X = WH, W >= 0, H >= 0$ found by solving an optimization problem. W becomes our basis conversion matrix, i.e. our dictionary, and H contains the coefficients that represent the data.

Metafeatures!

TODO Advantages:

- Interpretability thanks to non-negativity

Bag-of-Words model

Only store word frequency information.

N-gram

Contiguous sequence of words/characters, frequency counted similar to B.o.W. model.

Term-Document Matrix

A term may be either a word or a gram, representing each row in the matrix, whereby every column stores the frequency count for each term in some document

Red fish blue fish example

TODO Red fish blue fish example

Review Questions

- In highly variable data, it's best to use restrictive models to mitigate overfitting. Only use more flexibility when dealing with data naturally of that kind, i.e. non-linear data.

Crash Course on Neural Networks and Deep Learning

Neural Networks

Neural networks are modeled after physical brain neurons, but only inspirationally. Modern methods have lost most physical analogy.

Outputs of nodes are weighted differently as they become the inputs for other nodes.

We use non-linear *activation functions* like ReLU, sigmoids, or hyperbolic tangents, so as to make the behavior of the network more complicated than a simple linear combination of weights.

Gradient descent is used to train the weights into minimzing some loss function assosciated with training data and predictions made. By itself, gradient descent is prone to not reaching global minima, though this can be useful for avoiding overfitting.

The mathematical theory of neural networks is severly underdeveloped! We use them from a technical perspective without understanding why they perform particularly well on certain datasets.

It's a field of tricks!

Regularization via *early stopping* – walk towards the minimum but don't actually reach it – or *dropout*, which motivates individual nodes to better respond to structures by themselves, rather than relying on other nodes' behavior.

Deep Learning

Many many layers, many many parameters.

Only usable on *obscenely* large datasets.