Machine Learning

Session 4

3rd March 2018

Rama Krishna Bhupathi
ramakris@gmail.com

Agenda

- Regularization (L1 and L2)
- kNN (k Nearest Neighbour) Algorithm
- Wikipedia Doc Retrieval using Nearest Neighbor
- Predicting House Prices using kNN

Regularization

Regularization helps to choose preferred model complexity, so that model is better at predicting. Regularization adds a penalty term to the objective function and control the model complexity using that penalty term.

Regularization helps in preventing overfitting of your model . Overfitted models normally have high coefficients.Size of coefficients increase exponentially with increase in model complexity. Regularization attempts to reduce the complexity of the model by reducing the size of the coefficients (or even make them 0)

Types Regularization

There are two types of regularization that we apply to Linear Regression.

Lasso Regression : Also known as L1 Norm, Performs L1 regularization, i.e. adds penalty equivalent to absolute value of the magnitude of coefficients
Minimization objective :

Least Square Obj + α * (sum of absolute value of coefficients)

Ridge Regression: Also known as L2 Norm ,performs L2 regularization, i.e. adds penalty equivalent to square of the magnitude of coefficients
Minimization objective :

Least Square Obj + α * (sum of square of coefficients)

So what is Alpha (α) ? Tuning Parameter

LASSO (Least Absolute Shrinkage and Selection Operator)

Lasso can set some of the coefficients to zero, which is equivalent to the particular feature being excluded from the model.

Since it provides sparse solutions, it is generally the model of choice (or some variant of this concept) for modelling cases where the #features are in millions or more. In such a case, getting a sparse solution is of great computational advantage as the features with zero coefficients can simply be ignored.

Ridge

Ridge includes all (or none) of the features in the model. Thus, the major advantage of ridge regression is coefficient shrinkage and reducing model complexity.

Since it includes all the features, it is not very useful in case of exorbitantly high #features, say in millions, as it will pose computational challenges.

it generally works well even in presence of highly correlated features as it will include all of them in the model but the coefficients will be distributed among them depending on the correlation.

How do you use it?

```
In [15]: my_features_model = graphlab.linear_regression.create(train_data,target='price',features=my_features)
PROGRESS: Creating a validation set from 5 percent of training data. This may take a while.
          You can set ``validation_set=None`` to disable validation tracking.


Linear regression:
--------------------------------------------------------
Number of examples          : 16498
```

```
    them out into different columns.

l2_penalty : float, optional
    Weight on the l2-regularizer of the model. The larger this weight, the
    more the model coefficients shrink toward 0. This introduces bias into
    the model but decreases variance, potentially leading to better
    predictions. The default value is 0.01; setting this parameter to 0
    corresponds to unregularized linear regression. See the ridge
    regression reference for more detail.

l1_penalty : float, optional
    Weight on l1 regularization of the model. Like the l2 penalty, the
    higher the l1 penalty, the more the estimated coefficients shrink toward
    0. The l1 penalty, however, completely zeros out sufficiently small
    coefficients, automatically indicating features that are not useful for
    the model. The default weight of 0 prevents any features from being
```

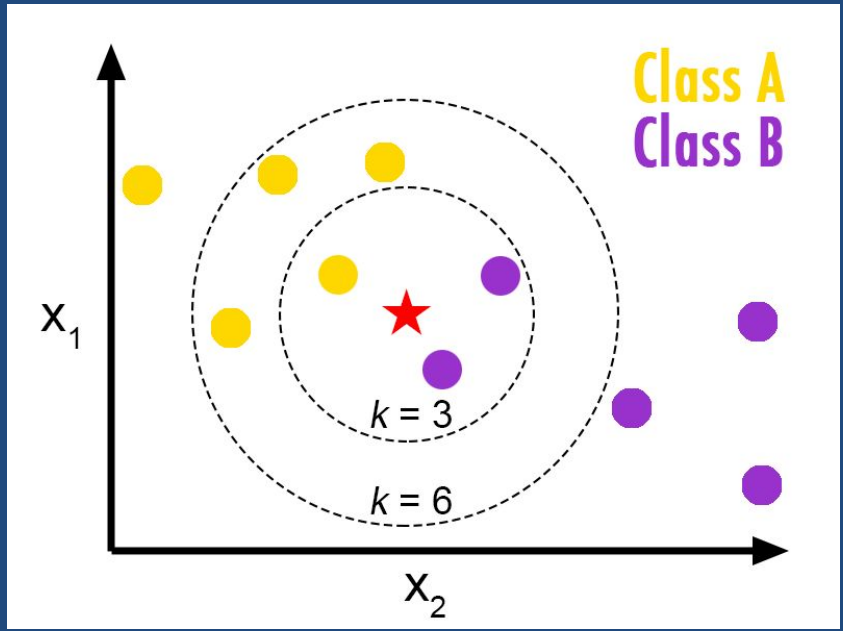
What is KNN?

The k-NN algorithm is among the simplest of all machine learning algorithms.

- ***Used for both Regression and Classification***
- ***Non-parametric..makes no assumptions on the underlying similarity.***
- ***KNN Algorithm is based on feature similarity.How closely out-of-sample features resemble our training set determines how we classify a given data point***

The red star is the query point.

How do you measure the closest point ?
Distance?



Distance Measurements.

Euclidean Distance

Distance metrics:
Defining notion of "closest"

In 1D, just Euclidean distance:

$$\text{distance}(x_j, x_q) = |x_j - x_q|$$

In multiple dimensions:

- can define many interesting distance functions
- most straightforwardly, might want to weight different dimensions differently

Distance Measurements.

Euclidean Distance(multi dimensions)

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Scaled Euclidean distance

Formally, this is achieved via

$$\text{distance}(\mathbf{x}_j, \mathbf{x}_q) = \sqrt{a_1(\mathbf{x}_j[1]-\mathbf{x}_q[1])^2 + \ldots + a_d(\mathbf{x}_j[d]-\mathbf{x}_q[d])^2}$$

weight on each input
(defining relative importance)

Other example distance metrics:

- Mahalanobis, rank-based, correlation-based, cosine similarity, Manhattan, Hamming, …

Distance Measurements.

TF-IDF

Term frequency-inverse document frequency is a statistic that reflects how important a word is to a specific document relative to all of the words in a collection of documents (the corpus). The tf-idf value increases proportionally to the number of times that word appears in the document, but is offset by the frequency of the word in the corpus.

TF-IDF document represenation

- Term frequency – inverse document frequency (tf-idf)
- Term frequency

- Inverse document frequency

$$\log \frac{\#\ \text{docs}}{1 + \#\ \text{docs using word}}$$

TF-IDF document represenation

- Term frequency – inverse document frequency (tf-idf)
- Term frequency

- Inverse document frequency