Machine learning general rules and concepts

Justin

# Variance and covariance

One conclusion out of the 2nd equations is when and are unrelated, .

# Why i.i.d

The short answer is in order to learn useful model one needs to assume the training samples to be representative of the entire data. Check this [link](https://stats.stackexchange.com/questions/213464/on-the-importance-of-the-i-i-d-assumption-in-statistical-learning).

# High bias vs. high variance

High bias: the selected model is too simple; high variance: the selected model is too complex

# Bias-variance decomposition

Assume the true relationship between the independent variable and the dependent variable is given by: , where is normally distributed noise with a mean of 0 and a standard deviation of . The fit function is: . The squared error for any data point is the following:

Note the first term describes the variability of the model if running over many data sets, and hence is the variance term.

# Principles of model selection

Based on speed, performance, or interpretability

# How to validate the model? How do we know the problem is solvable by machine learning models?

The rule of thumb is k-folds cross-validation. There are many reasons why selected models may fail:

* The selected model is too simple or too complicated
* The given data is skewed
* Too much noise
* Too many outliers
* Selected features are not informative enough – add new features
* Not enough data
* i.i.d condition is not satisfied

# Learning curve

The learning curve is a convenient tool for diagnose bias and variance in model fitting.

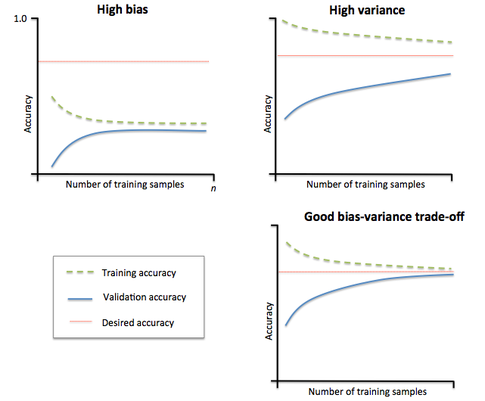


Figure . Learning curve

# Hyper-parameter tuning

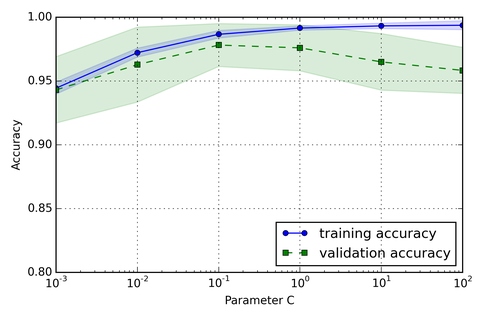


Figure . Dependence of training and validation score on a certain hyper-parameter

# Two fundamental milestones of classification algorithms

## Linear discriminant analysis (LDA)

LDA is essentially a linear transformation technique, which is mainly used for dimensionality reduction. The objective is to find the k-dimensional feature subspace that – linearly – separates the samples from different classes the best. LDA has a close-form solution

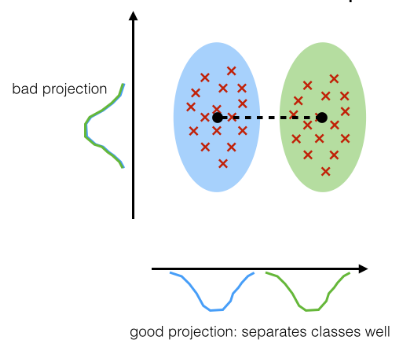


Figure . LDA. The horizontal axis is better than the vertical axis in separating the two groups

## Perceptron algorithm

This category essentially includes logistic regression, SVM, neuron networks. Different from LDA, this is an **incremental learner** – for each training sample, it compares the predicted class label to the actual label and modifies the model weights accordingly.

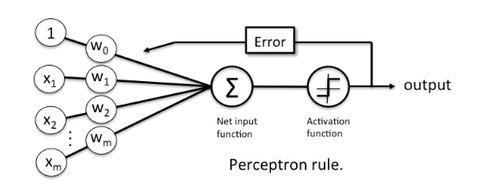


Figure . Perceptron algorithm has an error feedback loop

# A few commonly used activation functions

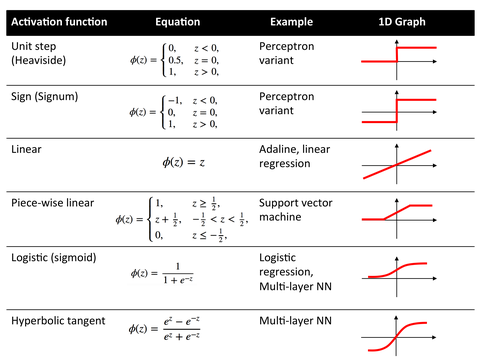


Figure . Commonly used activation functions

Resources:  
<https://github.com/rasbt/python-machine-learning-book/blob/master/faq/classifier-history.md>

# Parametric vs. non-parametric algorithms

Non-parametric models (can) become more and more complex with an increasing amount of data. Hence, a parametric model has a finite number of parameters, while a non-parametric model has (potentially) infinite number of parameters.

Parametric models: linear regression, logistic regression, linear SVM, etc.

Non-parametric models: K-nearest neighbor, decision trees, RBF kernel SVMs

# Gradient descent (GD, or batch GD), stochastic gradient descent (SGD), and min-batch gradient descent (MB-GD)

In GD, updates are made after computing the cost gradient based on the complete training set.

In SGD, updates are made after computing each training example. SGD is sometimes also referred to as iterative or on-line GD. Due to its stochastic nature, the path towards the global cost minimum is not "direct" as in GD, but may go "zig-zag" if visualized in a 2D space. SGD almost surely converges to the global cost minimum if the cost function is convex (or pseudo-convex)

MB-GD is a compromise between batch GD and SGD. MB-GD converges in fewer iterations than GD because the weights are updated more frequently; however, MB-GD enables vectorized operation, which typically results in a computational performance gain over SGD.

Some further notes:

* For SGD, and MB-GD, shuffling the data is important to avoid pre-existing order of the examples
* For MB-GD, the number of samples within each mini-batch is usually a power of 2 due to computer hardware requirement

|  |  |  |  |
| --- | --- | --- | --- |
|  | SGD | | MB-GD |
| Pros | 1. Can converges faster due to more frequent updates 2. Can hop out of local minima, very useful for models that have lots of local minima 3. Less stress on RAM 4. Online algorithm | 1. Can take advantage of vectorized computation 2. Somewhere between SGD and MB-GD | |
| Cons | 1. Too much fluctuation | |  |

Table 1. Pros and cons of SGD and MB-GD compared to batch GD



Figure . Visualization of batch GD, SGD, and MB-GD

Resources:  
<https://stats.stackexchange.com/questions/49528/batch-gradient-descent-versus-stochastic-gradient-descent>  
<https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3>

# Ticks to improve the GD-based learning

There are a few tricks to improve the GD-based learning:

* Use an adaptive learning rate that shrinks over time
* Adding a factor of previous gradient to the weight update for faster updates

# Importance of scaling the features

Scaling the data makes the GD algorithm converge faster

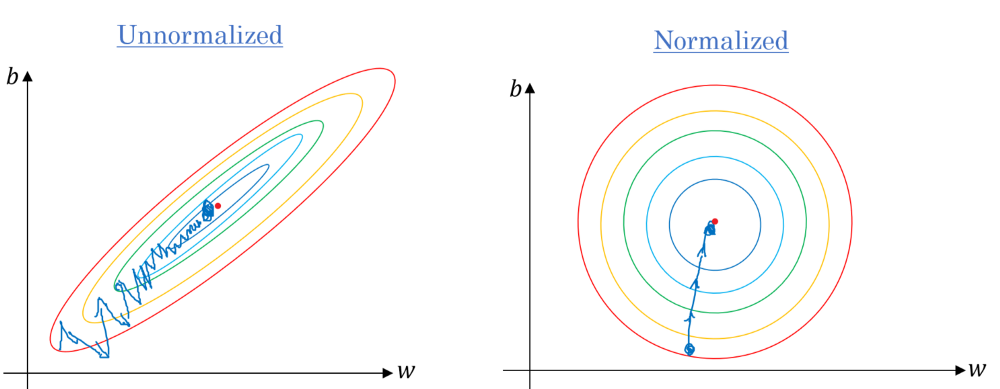


Figure . Normalizing the data puts the weights in roughly the same scale, and hence makes the GD algorithm converges faster. In the left figure, it’s very easy for the algorithm to overshoot in the squashed direction, and hence the minimization path is zig-zag shaped

# Connection between Pearson R and simple linear regression

Where is the slope of simple linear regression, is the Pearson correlation, and are the standard deviations.

Resources:  
<https://github.com/rasbt/python-machine-learning-book/blob/master/faq/pearson-r-vs-linear-regr.md>

# Bagging vs. boosting

Bagging and random forests are "bagging" algorithms that aim to reduce the complexity of models that over-fit the training data. In contrast, boosting (Ada-boost) is an approach to increase the complexity of models that suffer from high bias.

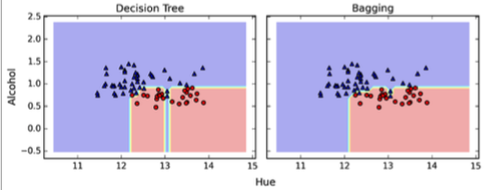


Figure . Bagging is capable of correcting the high variance of a single decision tree

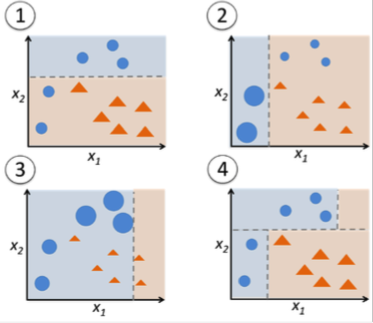


Figure . Boosting is able to incrementally learn from the mistakes and reduce the high bias. In other words, boosting gradually increases the complexity of the model

# The curse of dimensionality

## Counter-intuition in high dimensionality

### Most of the mass of a multivariate Gaussian distribution is not near the mean, but in an increasingly distant “shell” around it

### Most of the volume of a high-dimensional orange is in the skin, not the pulp

### If a constant number of examples is distributed uniformly in a high-dimensional hypercube, beyond some dimensionality most examples are closer to a face of the hypercube than to their nearest neighbor

### If we approximate a hypersphere by inscribing it in a hypercube, in high dimensions almost all the volume of the hypercube is outside the hypersphere

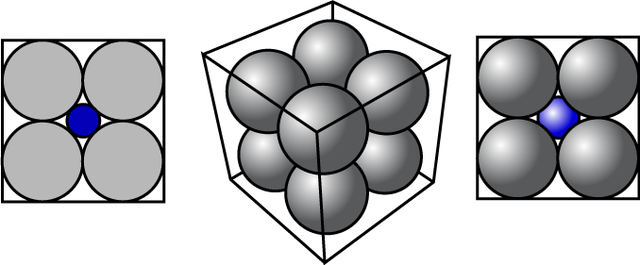


Figure . Another example of the curse of the dimensionality – put a blue hypersphere in the center gap of gray hyperspheres, which have a radius of ¼, and are placed tightly inside a hypercube of sides of length 1. Initially, the volume of the blue sphere is small but grows quickly as the number of dimensions increases. With 16 dimensions, the radius of the blue center sphere is exactly ½ and touches the sides of the hypercube. With > 16 dimensions, its radius is even larger and crashes through the sides of the cube, which is very counter-intuitive. See this [link](https://shapeofdata.wordpress.com/2013/04/02/the-curse-of-dimensionality/) for details

## The consequences of the high dimensionality in machine learning

### Over-fit (one extreme case if the number of features >= the number of samples)

For instance, the goal of the linear regression is the find one hyperplane that best fits the data. In 2D, a line can perfectly fit 2 distant points; similarly in 3D, a plane can perfectly fit 3 distant points. Hence, with the number of features equal to the number of samples, one can always find a hyperplane that perfectly fit all data points. With number of features exceeding the number of samples, more than one solution is possible.

### Sparsity (bad of clustering)

### Distance metric starts to fail (related to sparsity)

Both the sparsity and the failure of the distance metric can be understood in the following. Suppose we have features all uniformly distributed in the range of (-1, 1). As increases, more and more samples are found around the corners of the -D hypercube (due to the fact that the volume of the inscribed hypersphere diminishes). In addition, since there are corners, if the number of samples , it’s likely that each sample occupies one corner. Hence, each pair of samples has roughly the same distance, making the distance based algorithms (k-means, KNN) fail to work.

## How to overcome the curse of dimensionality

### Non-uniformity blessing

First of all, the curse of dimensionality occurs due to uniform distribution. The real data is neither uniformly distributed nor random, e.g., the handwriting in a figure is more distributed in the center. This is known as “non-uniformity blessing”.

### Dimensionality reduction, e.g., LDA, PCA

### Change the algorithm

Some algorithms (e.g., neuron networks) are more resistant to the curse of dimensionality than others. The idea behind is the Manifold Hypothesis. At a high level the Manifold Hypothesis suggests that the high dimensional data actually sits on a lower dimensional manifold embedded in higher dimensional space.

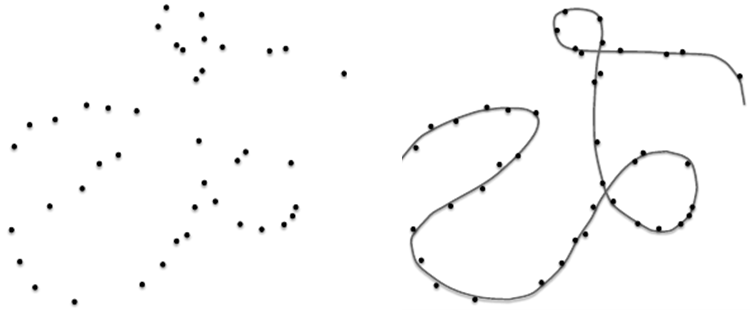


Figure . Manifold Hypothesis -- the high dimensional data sits on a lower dimensional manifold

## Relation to other concepts

### Random forest only selects randomly a subset of features to split each node because this reduces the chance of over-fit (or the curse of dimensionality)

### For larger degree of freedoms, the bulk part of the PDF of the chi-squared distribution shifts outward

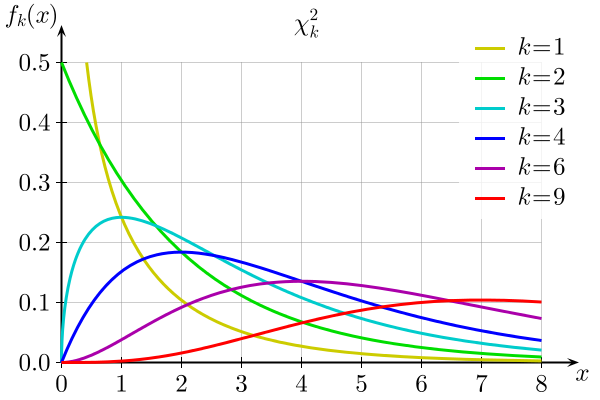


Figure 12. For larger degree of freedoms, the bulk part of the PDF of the chi-squared distribution shifts outward

# Chi-squared () distribution

If , …, are independent, standard normal random variables, then the sum of their squares,

is distributed according to the chi-squared distribution with k degrees of freedom.

## Derivation of

Let , the PDF of is given by:

Note is standard normally distributed, , hence:

The PDF of is then the derivative of against :