Haoran Zhao

MACHINE LEARNING

Contents

[Basic Statistics 5](#_Toc1599478)

[Confidence Interval 5](#_Toc1599479)

[Central Limit Theorem 5](#_Toc1599480)

[Student’s t-Distribution 5](#_Toc1599481)

[Errors in Hypothesis Testing 6](#_Toc1599482)

[Moments 6](#_Toc1599483)

[Statistical Learning 7](#_Toc1599484)

[Prediction Accuracy vs Model Interpretability 7](#_Toc1599485)

[Mean Squared Error 7](#_Toc1599486)

[Bias-Variance Trade-Off 7](#_Toc1599487)

[Bayes Classifier 8](#_Toc1599488)

[K-Nearest Neighbors 9](#_Toc1599489)

[Linear Regression 10](#_Toc1599490)

[Simple Linear Regression 10](#_Toc1599491)

[Estimating the Coefficients 10](#_Toc1599492)

[Assessing the Accuracy of the Coefficient Estimates 10](#_Toc1599493)

[Assessing the Accuracy of the Model 11](#_Toc1599494)

[Multiple Linear Regression 12](#_Toc1599495)

[Assumptions of Linear Regression 12](#_Toc1599496)

[Relationship between the Response and Predictors 12](#_Toc1599497)

[Mode Fit 13](#_Toc1599498)

[Other Considerations in the Regression Model 13](#_Toc1599499)

[Qualitative Predictors 13](#_Toc1599500)

[Extensions of the Linear Model 13](#_Toc1599501)

[Potential Problems 14](#_Toc1599502)

[Non-linearity of the Data 14](#_Toc1599503)

[Correlation of Error Terms 14](#_Toc1599504)

[Non-constant Variance of the Error Terms 14](#_Toc1599505)

[Non-normality of the Error Terms 14](#_Toc1599506)

[Outliers 15](#_Toc1599507)

[High Leverage Points 15](#_Toc1599508)

[Collinearity 15](#_Toc1599509)

[Perfect Collinearity 16](#_Toc1599510)

[Standardizing and Normalizing 16](#_Toc1599511)

[Dealing with Missing Values 17](#_Toc1599512)

[Classification 18](#_Toc1599513)

[Logistic Regression 18](#_Toc1599514)

[The Logistic Model 18](#_Toc1599515)

[Estimating the Regression Coefficients 18](#_Toc1599516)

[Multiple Logistic Model 19](#_Toc1599517)

[Logistic Regression Derivatives 20](#_Toc1599518)

[Logistic Regression for >2 Response Classes 21](#_Toc1599519)

[Discriminant Analysis 22](#_Toc1599520)

[Using Bayes’ Theorem for Classification 22](#_Toc1599521)

[Linear Discriminant Analysis for 22](#_Toc1599522)

[Linear Discriminant Analysis for 23](#_Toc1599523)

[Quadratic Discriminant Analysis 24](#_Toc1599524)

[A Comparison of Classification Methods 24](#_Toc1599525)

[Confusion Matrix 25](#_Toc1599526)

[ROC Curve 25](#_Toc1599527)

[Resampling Methods 27](#_Toc1599528)

[Cross-Validation 27](#_Toc1599529)

[Leave-one-out Cross-Validation 27](#_Toc1599530)

[k-Fold Cross-Validation 27](#_Toc1599531)

[Bootstrap 27](#_Toc1599532)

[Linear Model Selection and Regularization 28](#_Toc1599533)

[Subset Selection 28](#_Toc1599534)

[Best Subset Selection 28](#_Toc1599535)

[Forward Stepwise Selection 28](#_Toc1599536)

[Choosing the Optimal Model 28](#_Toc1599537)

[AIC, BIC and Adjusted 28](#_Toc1599538)

[Validation and Cross-Validation 29](#_Toc1599539)

[Shrinkage Methods 29](#_Toc1599540)

[Ridge Regression 29](#_Toc1599541)

[Lasso Regression 30](#_Toc1599542)

[Compare Ridge and Lasso Regression 30](#_Toc1599543)

[Dimension Reduction Methods 31](#_Toc1599544)

[Principal Components Regression 31](#_Toc1599545)

[Moving Beyond Linearity 32](#_Toc1599546)

[Polynomial Regression 32](#_Toc1599547)

[Step Functions 32](#_Toc1599548)

[Basis Functions 32](#_Toc1599549)

[Regression Splines 32](#_Toc1599550)

[Piecewise Polynomials 32](#_Toc1599551)

[Constraints and Splines 33](#_Toc1599552)

[The Spline Basis Representation 33](#_Toc1599553)

[Smoothing Splines 34](#_Toc1599554)

[Local Regression 34](#_Toc1599555)

[Generalized Additive Models 34](#_Toc1599556)

[Tree-Based Methods 36](#_Toc1599557)

[The Basics of Decision Trees 36](#_Toc1599558)

[Regression Trees 36](#_Toc1599559)

[Classification Trees 37](#_Toc1599560)

[Bagging, Random Forests, Boosting, Stacking 37](#_Toc1599561)

[Bagging 37](#_Toc1599562)

[Number of Trees 38](#_Toc1599563)

[Out-of-Bag Error Estimation 38](#_Toc1599564)

[Variable Importance Measures 38](#_Toc1599565)

[Random Forests 38](#_Toc1599566)

[Boosting 39](#_Toc1599567)

[Gradient Boosting 39](#_Toc1599568)

[AdaBoost 41](#_Toc1599569)

[Stacking 42](#_Toc1599570)

[Support Vector Machines 43](#_Toc1599571)

[Maximal Margin Classifier 43](#_Toc1599572)

[Support Vector Classifier 43](#_Toc1599573)

[Support Vector Machine 43](#_Toc1599574)

[Unsupervised Learning 46](#_Toc1599575)

[Principal Components Analysis 46](#_Toc1599576)

[Clustering Methods 46](#_Toc1599577)

[K-Means Clustering 46](#_Toc1599578)

[Hierarchical Clustering 47](#_Toc1599579)

[Time Series Models 49](#_Toc1599580)

[Autoregressive Process 49](#_Toc1599581)

[Moving Average Process 49](#_Toc1599582)

[ARMA Process 49](#_Toc1599583)

[ARIMA Process 49](#_Toc1599584)

[Cointegration 49](#_Toc1599585)

[ARCH Process 50](#_Toc1599586)

[GARCH Process 50](#_Toc1599587)

[AB Testing 51](#_Toc1599588)

[2-sample t-test 51](#_Toc1599589)

[2-sample z-test 51](#_Toc1599590)

[ANOVA F-test 52](#_Toc1599591)

[Chi-squared test for independence 52](#_Toc1599592)

[Neural Network 53](#_Toc1599593)

[Anatomy 53](#_Toc1599594)

[Activation Functions 53](#_Toc1599595)

[Output Units and Loss Function 54](#_Toc1599596)

[Architecture 55](#_Toc1599597)

[Regularization 55](#_Toc1599598)

[Norm penalties 55](#_Toc1599599)

[Early stopping 56](#_Toc1599600)

[Data Augmentation 56](#_Toc1599601)

[Sparse Representation 56](#_Toc1599602)

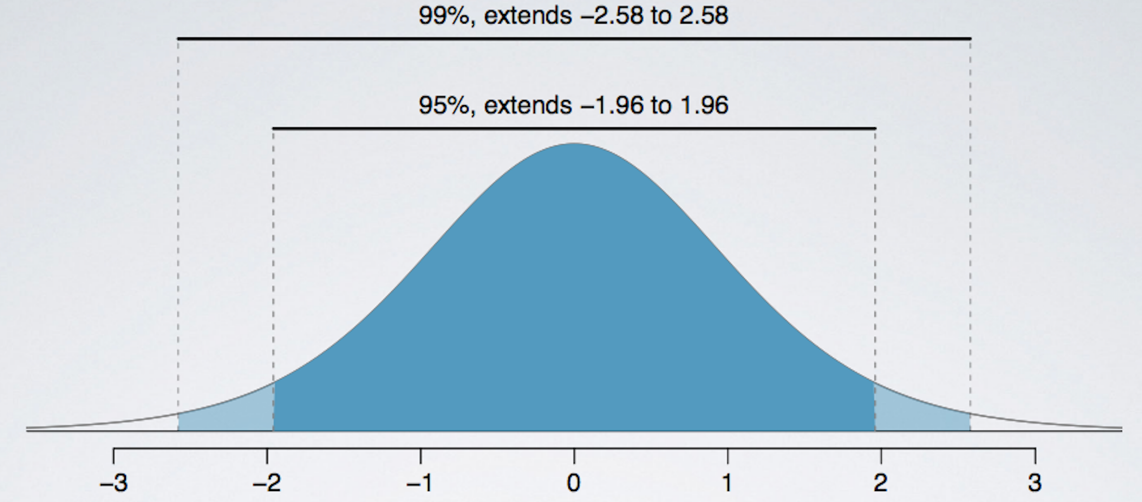
[Bagging 56](#_Toc1599603)

[Dropout 56](#_Toc1599604)

# Basic Statistics

## Confidence Interval

* A 95% confidence interval is a range of values that you can be 95% certain contains the true mean of the population.

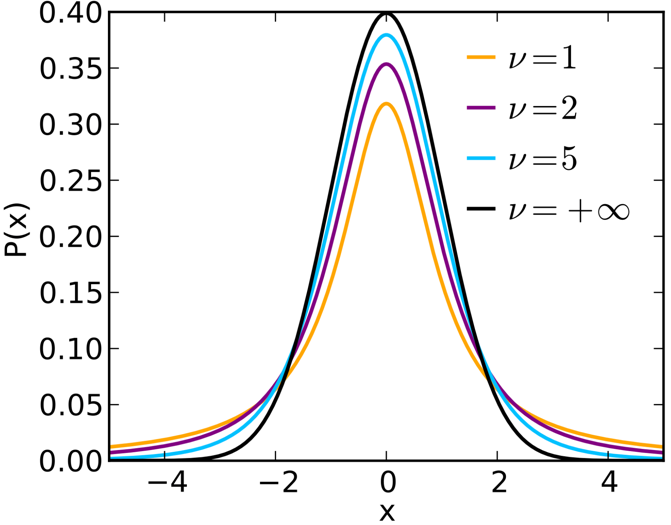


## Central Limit Theorem

* The central limit theorem of statistics states that in selecting simple random samples of size n from a population with a mean and a finite variance , the sampling distribution of the sample mean approaches a normal probability distribution with mean and a variance equal to as the sample size becomes large.
* The central limit theorem is extremely useful because the normal distribution is relatively easy to apply to hypothesis testing and to the construction of confidence intervals.

## Student’s t-Distribution

* Symmetrical (bell shaped)
* Defined by single parameter, degrees of freedom (df), where for hypothesis tests and confidence intervals involving a sample mean.
* Has fatter tails than a normal distribution; the lower the df, the fatter the tails and the wider the confidence interval around the sample mean for a given probability that the interval contains the true mean.
* As sample size (degrees of freedom) increases, the t-distribution approaches normal distribution.



## Errors in Hypothesis Testing

* The significance level is the probability of making a Type I error (rejecting the null when it is true) and is designated by the Greek letter alpha (). A significance level of 5% means there is a 5% chance of rejecting null hypothesis, when it’s actually true.

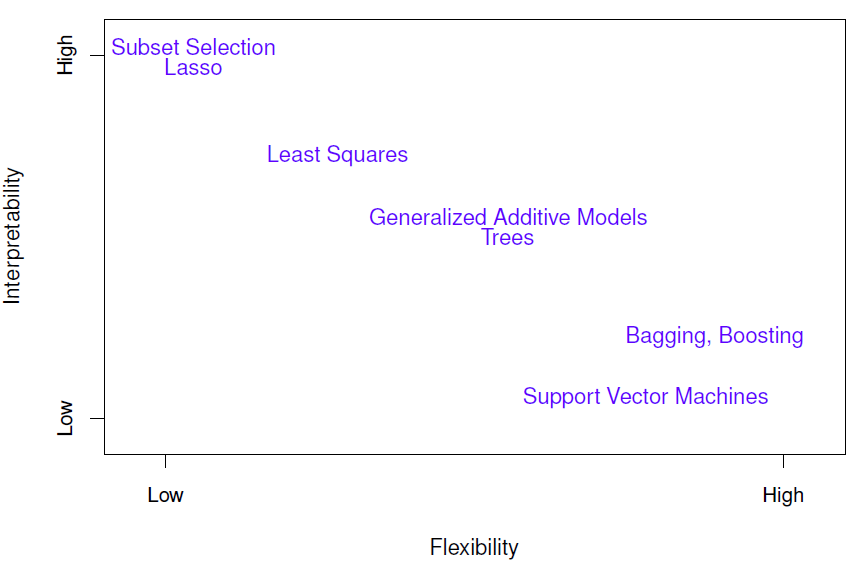
|  |  |  |
| --- | --- | --- |
| Type I and Type II Errors in Hypothesis Testing | | |
| Decision | True Condition | |
| is true | is false |
| Do not reject | Correct decision | Incorrect decision  Type II error |
| Reject | Incorrect decision  Type I error  Significance level = P(Type I Error) | Correct decision  Power of the test = 1 – P(Type II Error) |

## Moments

* 1st moment: mean
* 2nd moment: variance
* 3rd moment: skewness
* 4th moment: kurtosis

# Statistical Learning

## Prediction Accuracy vs Model Interpretability

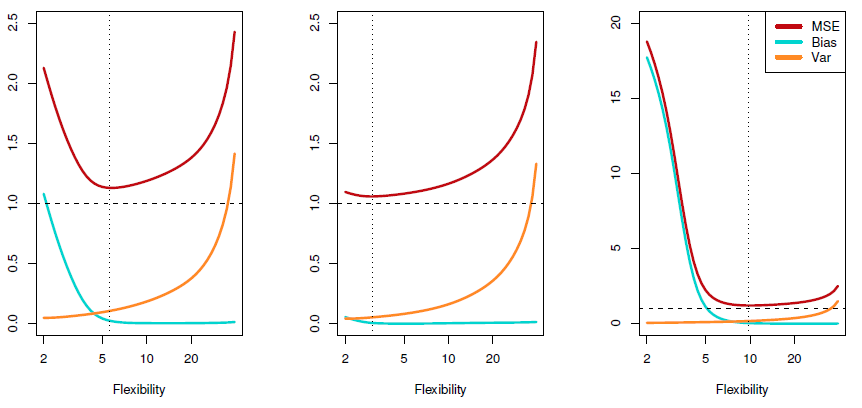


## Mean Squared Error

where is the prediction that gives for the observation.

## Bias-Variance Trade-Off

* The expected test MSE, for a given value , can always be decomposed into the sum of three fundamental quantities:
  + The variance of
  + The squared bias of
  + The variance of the error term
* That is,
* : expected test MSE, and refers to the average test MSE that we would obtain if we repeatedly estimated f using a large number of training sets, and tested each at . The overall expected test MSE can be computed by averaging over all possible values of in the test set.
* : refers to the amount by which would change if we estimated it using a different training data set. Since the training data are used to fit the statistical learning method, different training data sets will result in a different .
* : refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
* As a general rule, as we use more flexible methods, the variance will increase and the bias will decrease.



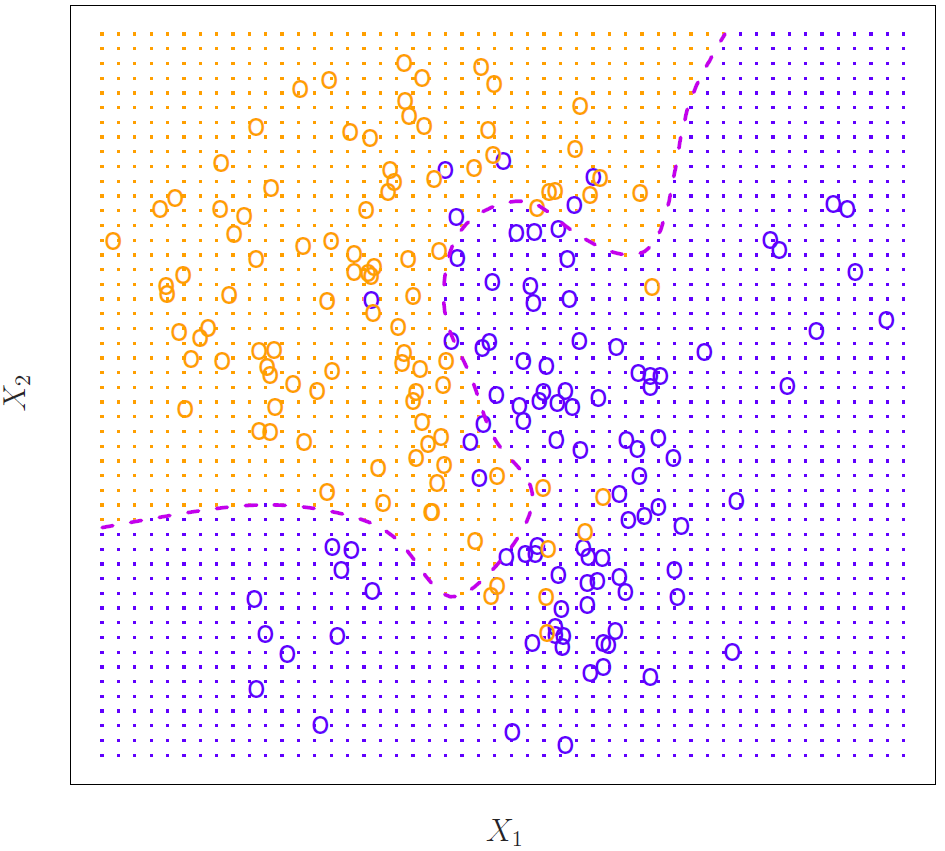
* Regarding the above plot:
  + Left chart: True relationship is not perfect linear, neither too far from linear.
  + Middle chart: True relationship is very close to linear.
  + Right chart: True relationship is very non-linear.

## Bayes Classifier

* In classification setting, the most common approach for quantifying the accuracy of our estimate is the training/test error rate, the proportion of mistakes that are made if we apply our estimate to the training observations:
* It is possible to show that the test error rate given above is minimized, on average, by a very simple classifier that assigns each observation to the most likely class given its predictor values. In other words, we should simply assign a test observation with predictor vector to the class j for which

is largest.

* In a two-class problem, where there are only two possible response values, say class 1 or class 2, the Bayes classifier corresponds to predicting class one if , and class two otherwise.



The purple dashed line represents the points where the probability is exactly 50%. This is called the Bayes decision boundary.

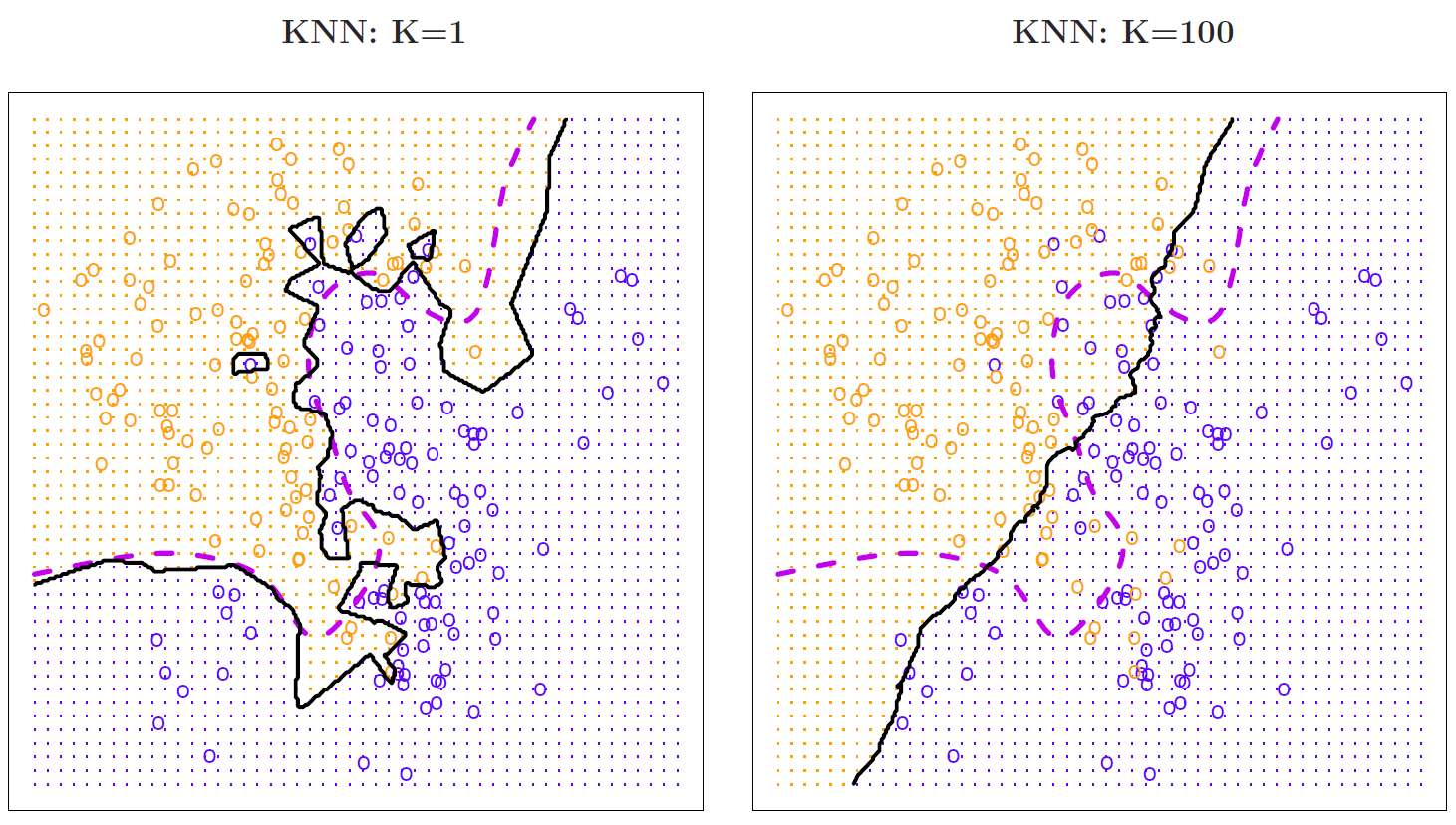
* The Bayes classifier produces the lowest possible test error rate, called the Bayes error rate. Since the Bayes classifier will always choose the class for which the above equation is largest, the error rate at will be . In general, the overall Bayes error rate is given by:

where the expectation averages the probability over all possible values of X. It’s positive in our above chart, because the classes overlap in the true population, so for some values of . The Bayes error rate is analogous to the irreducible error.

## K-Nearest Neighbors

* Given a positive integer K and a test observation , the KNN classifier first identifies the K points in the training data that are closest to , represented by . It then estimates the conditional probability for class j as the fraction of points in whose response values equal j:

Finally, KNN applies Bayes rule and classifies the test observation to the class with the largest probability.



* Prediction vs k:



* Python: from sklearn import neighbors.KNeighborsClassifier

# Linear Regression

## Simple Linear Regression

### Estimating the Coefficients

* The simple linear regression:
* The least squares approach chooses and to minimize the MSE. Using some calculus, one can show that the minimizers are:

where and are the sample means.

### Assessing the Accuracy of the Coefficient Estimates

* Regarding the estimation of population mean of a random variable Y. A natural question is as follows: how accurate is the sample mean as an estimate of ? In general, we answer this question by computing the standard error of , written as SE(). We have the well-known formula:

The above equation tells us how the standard deviation shrinks with n – the more observations we have, the smaller the standard error of .

* If our data is drawn from a larger set of observations, then we can empirically estimate the standard errors, and of and through bootstrapping.
* If we know the variance of the noise , we can compute and analytically, using the formula below:

where .

* In practice, we don’t know the theoretical value of , since we do not know the exact distribution of the noise . However, if we make the following assumptions:
  + The errors and are uncorrelated, for .
  + Each is normally distributed with mean 0 and variance .

Then, we can empirically estimate , from the data and our regression line:

* Standard errors can be used to compute confidence intervals. A 95% confidence interval is defined as a range of values such that with 95% probability, the range will contain the true unknown value of the parameter. The range is defined in terms of lower and upper limits computed from the sample of data. For linear regression, the 95% confidence interval approximately takes the form:
* Standard errors can also be used to perform hypothesis tests on the coefficients. The most common hypothesis test involves testing the null hypothesis of

Versus the alternative hypothesis:

We compute a t-statistic, given by

Which measures the number of standard deviations that is away from 0. If there really is no relationship between X and Y, then we expect that the t-statistic will have a t-distribution with n-2 degrees of freedom. The t-distribution has a bell shape and for values of n greater than approximately 30, it is quite similar to the normal distribution. Consequently, it is a simple matter to compute the probability of observing any value equal to |t| or larger, assuming . We call this probability the p-value.

* **Q:** What are going to happen with the following scenarios?
  + More data:
    - and don’t change, because they are unbiased estimators.
    - and and t-stat
  + Larger coverage (more spread out):
    - and don’t change, because they are unbiased estimators.
    - and t-stat
  + Better data:
    - and don’t change, because they are unbiased estimators.
    - and t-stat
* **Q:** Say you copy and paste the original dataset twice without realizing it, what would happen to your coefficient, t-stat of the coefficient, and R-squared of the simple linear regression?

**A:**

* + Coefficient and R-squared: no change
  + t-stat of the coefficient: becomes larger by a factor of
  + CI of the coefficient: becomes narrower by a factor of
* **Q:** Based on the above idea, how would you construct a dataset that can give you a very high t-stat of a coefficient, but with very low R-squared?

**A:** You keep copying the data.

### Assessing the Accuracy of the Model

* Mean Squared Error:
* **Q:** OLS is minimizing the sum of squares of residuals. Why squares? Why not absolute or power of 3, or 4?

**A:** (1) Absolute could not be taken derivatives. (2) Assuming normal distribution, then due to the PDF of normal distribution, maximum likelihood would effectively minimize the sum of squares of residuals.

* statistic:

measures the proportion of variability of Y that can be explained by X.

Recall that correlation, defined as

is also a measure of linear relationship between X and Y. It can be shown that in the simple linear regression setting, .

* + If the model is as good as the mean values, , then .
  + If the model is perfect, then .
  + can be negative if the model is worse than the average. This can happen when we evaluate the model in the test set. In training set, can be negative if there is no constant term.

## Multiple Linear Regression

* The model:

Thus, the MSE can be expressed in vector matrix as:

Minimizing the MSE using vector calculus yields:

* **Q:** If the values of the independent variables double, what would happen to the estimated coefficient?

**A:** The estimated coefficients would reduce by half.

### Assumptions of Linear Regression

* A linear relationship exists between the dependent and independent variables.
* The independent variable is uncorrelated with the residual term.
* Observations are independent of one another.
* Error terms are i.i.d normally distributed
* No perfect multicollinearity.

### Relationship between the Response and Predictors

* We test the null hypothesis:

versus the alternative:

is non-zero.

This hypothesis test is performed by computing the F-statistic,

where , and

* + If the linear model assumptions are correct, one can show that
  + If is true, one can show that
  + Hence, when there is no relationship between the response and predictors, one would expect the F-statistic to take on a value close to 1. On the other hand, if is true, then , so we expect F to be greater than 1.
  + If TSS=RSS, then prediction is just as good as the average, R-squared = 0, and F = 0.
  + When is true and the errors have a normal distribution, the F-statistic follows an F-distribution. For any given value of n and p, any statistical software can be used to compute the p-value associated with the F-statistic using the distribution.
* **Q:** Given these individual p-values for each variable, why do we need to look at the overall F-statistic? After all, it seems likely that if any one of the p-values for the individual variables is very small, then at least one of the predictors is related to the response.

**A:** This logic is flawed, especially when then the number of predictors p is large. For instance, consider an example in which p=100 and is true, so no variable is truly associated with the response. In this situation, about 5% of the p-values associated with each variable will be below 0.05 by chance. In other words, we expect to see approximately five small p-values even in the absence of any true association between the predictors and the response. In fact, we are almost guaranteed that we will observe at least one p-value below 0.05 by chance! However, the F-statistic does not suffer from this problem because it adjusts for the number of predictors. Hence, if is true, there is only 5% chance that the F-statistic will result in a p-value below 0.05, regardless of the number of predictors or the number of observations.

### Mode Fit

* MSE
* :
  + will always increase when more variables are added to the model, even if those variables are only weakly associated with the response. This is due to the fact that adding another variable to the least squares equations must allow us to fit the training data (though not necessarily the testing data) more accurately.

## Other Considerations in the Regression Model

### Qualitative Predictors

* Create dummy variables to represent categories.
  + There will always be one fewer dummy variable than the number of levels. The level without dummy variable is known as the baseline.
* Python: sklearn.preprocessing.OneHotEncoder

### Extensions of the Linear Model

* Removing the additive assumption:
  + Add interaction terms among predictors
  + The *hierarchical principle* states that if we include an interaction in a model, we should also include the main effects, even if the p-values associated with their coefficients are not significant. In other words, if the interaction between and seems important, then we should include both and in the model even if their coefficient estimates have large p-values. The rationale for this principle is that if is related to the response, then whether or not the coefficients of or are exactly zero is of little interest. Also is typically correlated with and , and so leaving them out tends to alter the meaning of the interaction.
* Removing the linear relationship assumption:
  + Add polynomial terms
* Python: sklearn.preprocessing.PolynomialFeatures

### Potential Problems

#### Non-linearity of the Data

* Detection: Residual plot is helpful.
  + In simple linear regression model, we can plot the residuals, , versus predictor .
  + In multiple linear regression model, we can plot residuals, versus predicted values .
* Remedy:
  + Use non-linear transformations of the predictors, such as .

#### Correlation of Error Terms

* Detection:
  + Residual plot
  + For time-series model only: DW Test (if around 2 then no correlation, 0-2 positive correlation, 2-4 negative correlation.) for autocorrelation check
  + For time-series model only: ACF chart for autocorrelation check
* Effect: If in fact there is correlation among the error terms, then:
  + t-stat
  + 95% CI
  + p-value
  + Type I Error
* Examples:
  + Time-series data
  + Accidentally copied and pasted the data twice
* Remedy:
  + Use Newey-West adjusted standard errors.

#### Non-constant Variance of the Error Terms

* Non-constant variances in the errors, or heteroscedasticity, can be shown in residual plot.
* Detection:
  + Residual plot
  + BP test
* Effect: If in fact there is heteroscedasticity in the error terms, then:
  + t-stat
  + 95% CI
  + p-value
  + Type I Error
* Remedy:
  + Transform the response Y, such as .
  + Use white standard error to correct the standard deviation
  + Use weighted least square estimation. Sometimes we have a good idea of the variance of each response. For example, the ith response could be an average of raw observations. If each of these raw observations is uncorrelated with variance , then their average has variance . In this case a simple remedy is to fit the model by weighted least squares, with weights proportional to the inverse variances – ie. in this case. Another example is that Barra use square root of market cap as weighting scheme.

#### Non-normality of the Error Terms

* How to check:
  + QQ plot: compares the percentile of data series versus percentile of normal distribution
  + SK (Shapiro-Wilk) test: quantifies the distance between sample empirical distribution versus normal distribution
* Why important:
  + The regression assumption that is generally least important is that the errors are normally distributed. In fact, for the purpose of estimating the regression line, the assumption of normality is barely important at all. It doesn’t matter for unbiasedness or consistency of the beta estimates.
  + The only case that it matters is in small samples, where non-normality of error terms will affect t-stats of the coefficient estimates.

#### Outliers

* Residual plot is helpful.
* In practice, it can be difficult to decide how large a residual term needs to be before we consider the point to be an outlier. To address this problem, instead of plotting the residual, we can plot the studentized residuals, computed by dividing each residual by its estimated standard error. Observations whose studentized residuals are greater than 3 in absolute value are possible outliers.
* Another method is Cook’s distance, which measures how much the fitted value would change if the particular observation is deleted.

#### High Leverage Points

* High leverage observations tend to have a sizable impact on the estimated regression line. It is cause for concern if the least squares line is heavily affected by just a couple of observations, because any problems with these points may invalidate the entire fit.
* In simple linear regression, high leverage observations are fairly easy to identify, since we can simply look for observations for which the predictor value is outside of the normal range of the observations. But in a multiple linear regression with many predictors, it is possible to have an observation that is well within the range of each individual predictor’s values, but that is unusual in terms of the full set of predictors, as shown in the middle chart below:



* In order to quantify an observation’s leverage, we compute the leverage statistic. A large value of this statistic indicates an observation with high leverage. For a simple linear regression,

The leverage statistic is always between and 1, and the average leverage for all the observations is always equal to . If a given observation has a leverage statistic that greatly exceeds , then we may suspect that the corresponding point that has high leverage.

#### Collinearity

* Adding a new predictor, there are three scenarios:
  + Pure noise and don’t change, doesn’t change.
  + Independent with other X & lots of power doesn’t change, ,
  + Some power & some correlation with other X , but may or .
* Assuming uncorrelated noise then we can show that
* If in fact there is collinearity among predictors, then
  + t-stat
  + 95% CI
  + p-value
  + Type II Error. The power of the hypothesis test – the probability of correctly detecting a non-zero coefficient – is reduced.
* Detection:
  + Look at the correlation matrix of the predictors.
  + Unfortunately, not all collinearity problems can be detected by inspection of the correlation matrix: it is possible for collinearity to exist between three or more variables even if no pair of variables has a particularly high correlation. We call this situation multicollinearity. Instead of inspecting the correlation matrix, a better way to assess multicollinearity is to compute the variance inflation factor (VIF). The VIF for each variable can be computed using the formula:

where is the from a regression of onto all of the other predictors. If is close to one, then collinearity is present, and so VIF will be large. Typically, in practice there is a small amount of collinearity among the predictors. As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity.

* Remedy:
  + Drop one of the problematic variables from the regression
  + Or combine the collinear variables into a single predictor. For instance, we might take the average of standardized versions of the two correlated predictors, in order to create a new variable.

#### Perfect Collinearity

* OLS cannot be solved.
* Example: All dummy variables are included while also keeping constant term.

### Standardizing and Normalizing

* Normalizing:
  + It means to bound your variable’s observations within [0, 1]. Good when interpretations of “percentage of max value” makes sense.
  + It is only for improving interpretation (and dealing with numerically very large or small measures). Does not improve algorithms otherwise.
* Standardizing:
  + It means to re-center and re-scale your variable’s observations to have mean zero and variance one. Good to put all of your variables on the same scale (have same weight) and to turn interpretations into “changes in terms of standard deviation”.
  + It can be used for improving interpretations and should be used for specific algorithms. Which ones? Regularization and PCA!
  + You can standardize without assuming things to be normally distributed. It just makes the interpretation nice if they are normally distributed.
* **Q:** For which of the following do the units of the predictors matter (e.g. trip length in minutes vs seconds; temperature in F vs C)?
  + k-NN: yes. Scaling affects distance metric, which determines what ‘neighbor’ means
  + Linear regression: no. Multiply predictor by c, and divide coefficient by c.
  + Lasso regression: yes. If we divide coefficient by c, then corresponding penalty term is also divided by c.
  + Ridge regression: yes. Same as Lasso, except penalty divided by .
* Python: from sklearn.preprocessing import StandardScaler

### Dealing with Missing Values

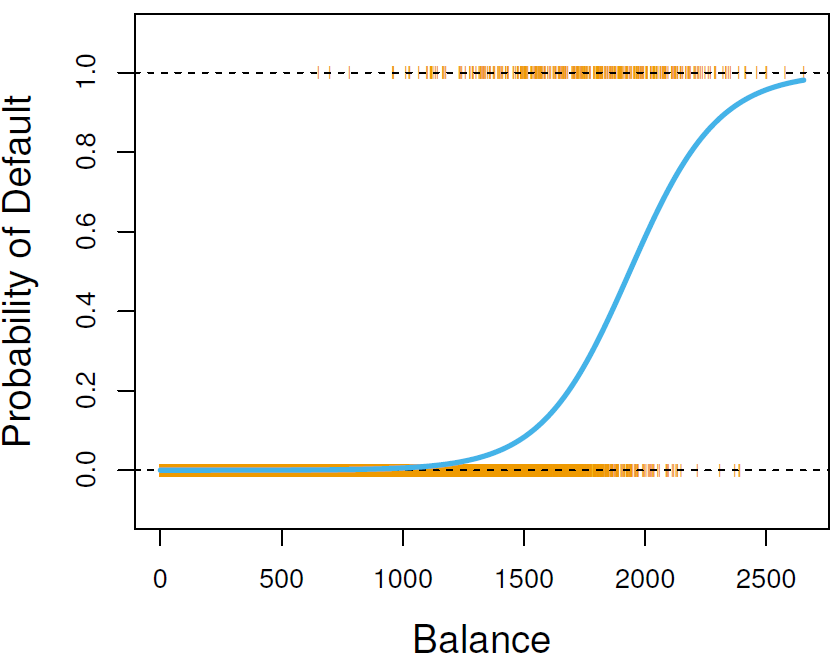
* There are several different approaches to imputing missing values:
  + Plug in the mean or median (quantitative) or most common class (categorical) for all missing values in a variable.
  + Create a new variable that is an indicator of missingness, and include it in any model to predict the response (also plug in zero or the mean in the actual variable).
  + Hot deck imputation: for each missing entry, randomly select an observed entry in the variable and plug it in.
  + Model with imputation: plug in predicted values from a model based on other observed predictors.
  + Model with imputation with uncertainty: plug in predicted values plus randomness from a model based on the other observed predictors.

# Classification

## Logistic Regression

### The Logistic Model

* In logistic regression, we use the logistic function:



* After a bit of manipulation, we find that:

The quantity is called the odds, and can take on any value between 0 and . Values of the odds close to 0 and indicate very low and high probabilities, respectively.

* By taking the logarithm of both sides, we arrive at

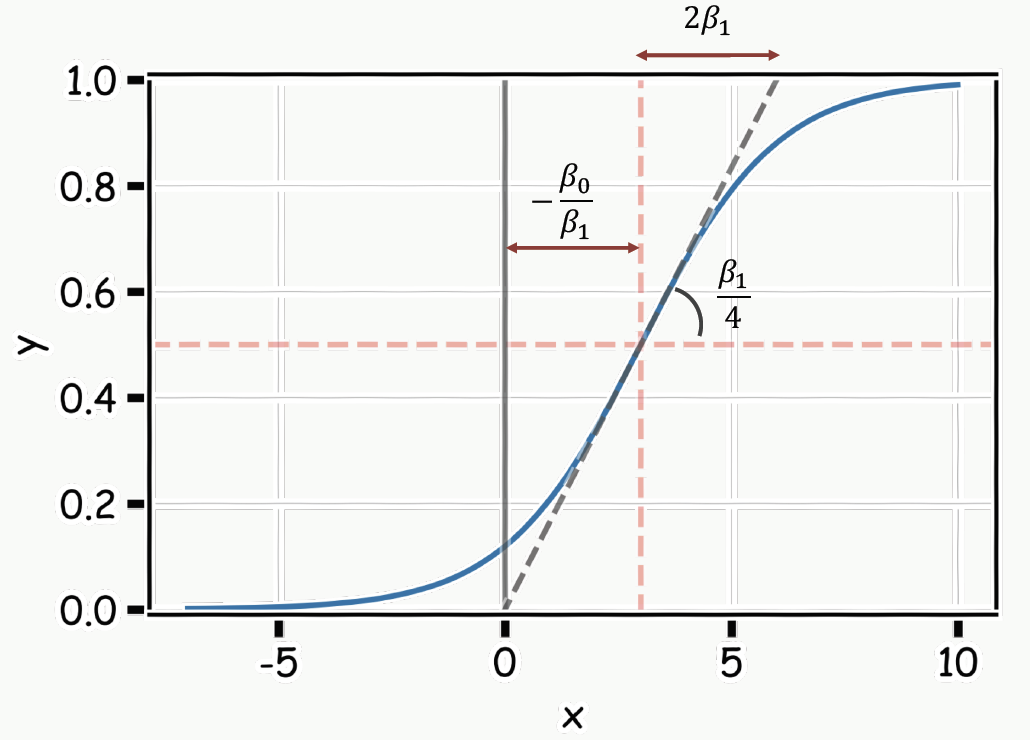
The left-hand side is called the log-odds or logit. We see that the logistic regression model has a logit that is linear in X.

### Estimating the Regression Coefficients

* We use maximum likelihood method to estimate the coefficients. The basic intuition behind using maximum likelihood to fit a logistic regression model is as follows: we seek estimates for and such that the predicted probability of default for each individual, corresponds as closely as possible to the individual’s observed default status. This can be formalized using a mathematical equation called a likelihood function:

The estimates and are chosen to maximize this likelihood function.

* In the linear regression setting, the least squares approach is in fact a special case of maximum likelihood.
* The estimated intercept in logistic regression is typically not of interest; its main purpose is to adjust the average fitted probabilities to the proportions of ones in the response variable.
* As a result, the model will predict with an S-shaped curve, which is the general shape of the logistic function.
  + shifts the curve right or left.
  + controls steepness of the S-shaped curve.



### Multiple Logistic Model

* With multiple predictors, we can generalize as follows:

where are p predictors. Equation can be rewritten as

* Maximizing the log-likelihood is equivalent to minimizing the following loss function:
* A penalty factor can be added to this loss function and results in a new loss function that penalizes large values of the parameters:

The result is just like in a linear regression: shrink the parameter estimates towards zero.

### Logistic Regression for >2 Response Classes

* Multinomial logistic regression:
  + It sets one of the categories in the response variable as the reference group, and then fits separate logistic regression models to predict the other cases based off the reference group.

Class K is as the reference group above.

* + There are K – 1 models to fit.
  + Final probabilities:
  + It relies on the assumption of [independence of irrelevant alternatives](https://en.wikipedia.org/wiki/Independence_of_irrelevant_alternatives) (IIA), which states that the odds of preferring one class over another do not depend on the presence or absence of other "irrelevant" alternatives.
* One-over-rest logistic regression:
  + The probability of each category is predicted over the rest of the categories combined.
  + There are K models to fit.
  + Final probabilities adjusted by softmax function:
  + It does not rely on the assumption of [independence of irrelevant alternatives](https://en.wikipedia.org/wiki/Independence_of_irrelevant_alternatives) (IIA).
* Python: from sklearn.linear\_model import LogisticRegression

## Discriminant Analysis

### Using Bayes’ Theorem for Classification

* Logistic regression involves directly model using the logistic function. We now consider an alternative approach, we model the distribution of the predictors X separately in each of the response classes (i.e. given Y), and then use Bayes’ theorem to flip these around into estimates for .
* The Bayes’ theorem states that

### Linear Discriminant Analysis for

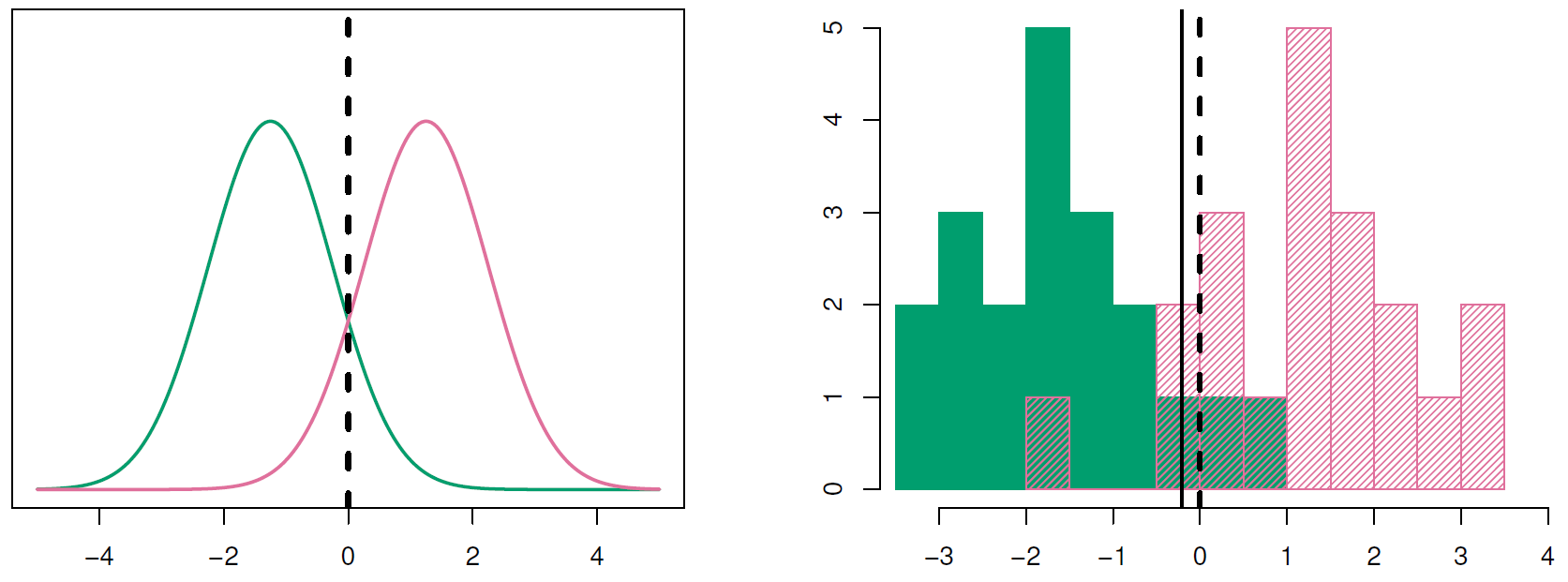
* Suppose we assume that is normal or Gaussian. In the one-dimensional setting, the normal density takes the form

where and are the mean and variance parameters for the kth class. For now, let’s further assume that : that is, there is a shared variance term across all K classes, which for simplicity we can denote by .

* Taking the log and rearranging the terms, it can be shown that this is equivalent to assigning the observation to the class for which

is the largest. The word linear in the classifier’s name stems from the fact that the discriminant function are linear functions of .

* If and , the Bayes decision boundary corresponds to the point where

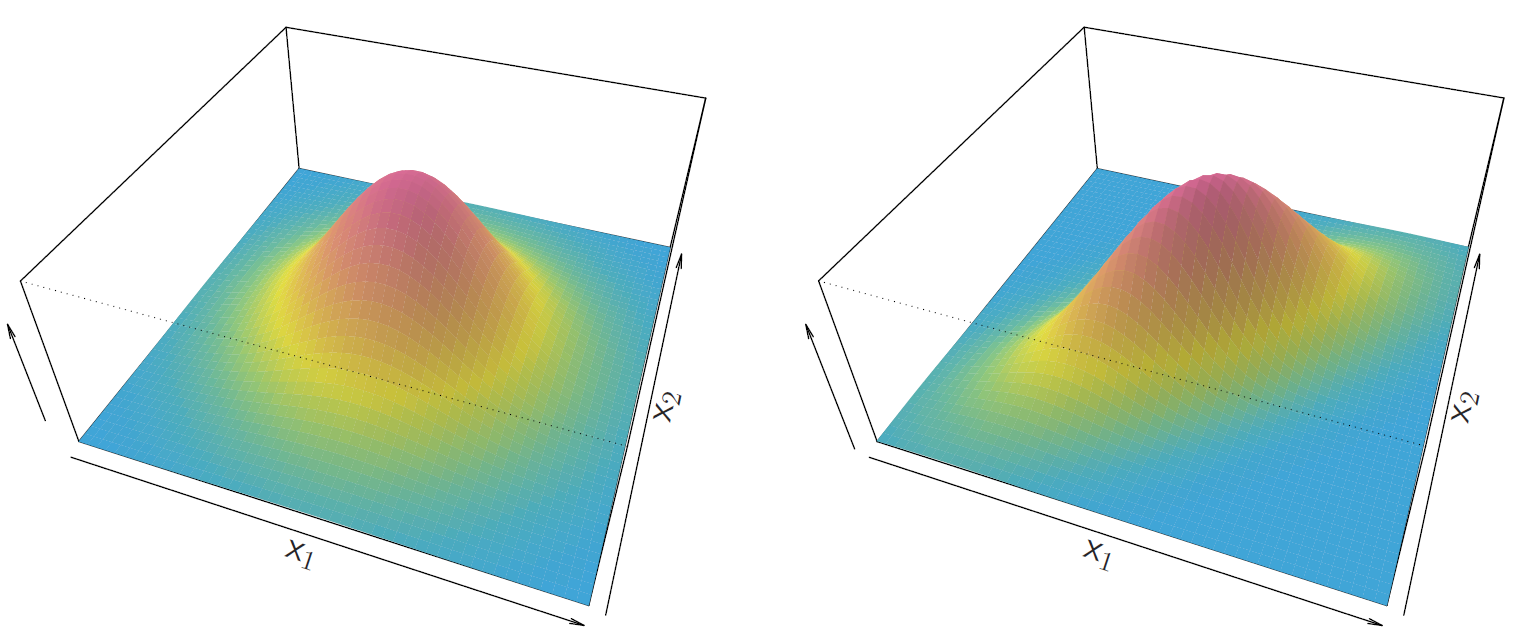


* The linear discriminant analysis (LDA) method approximates the Bayes classifier by plugging estimates for and . In particular, the following estimates are used:

can be seen as a weighted average of the sample variances for each of the K classes.

### Linear Discriminant Analysis for

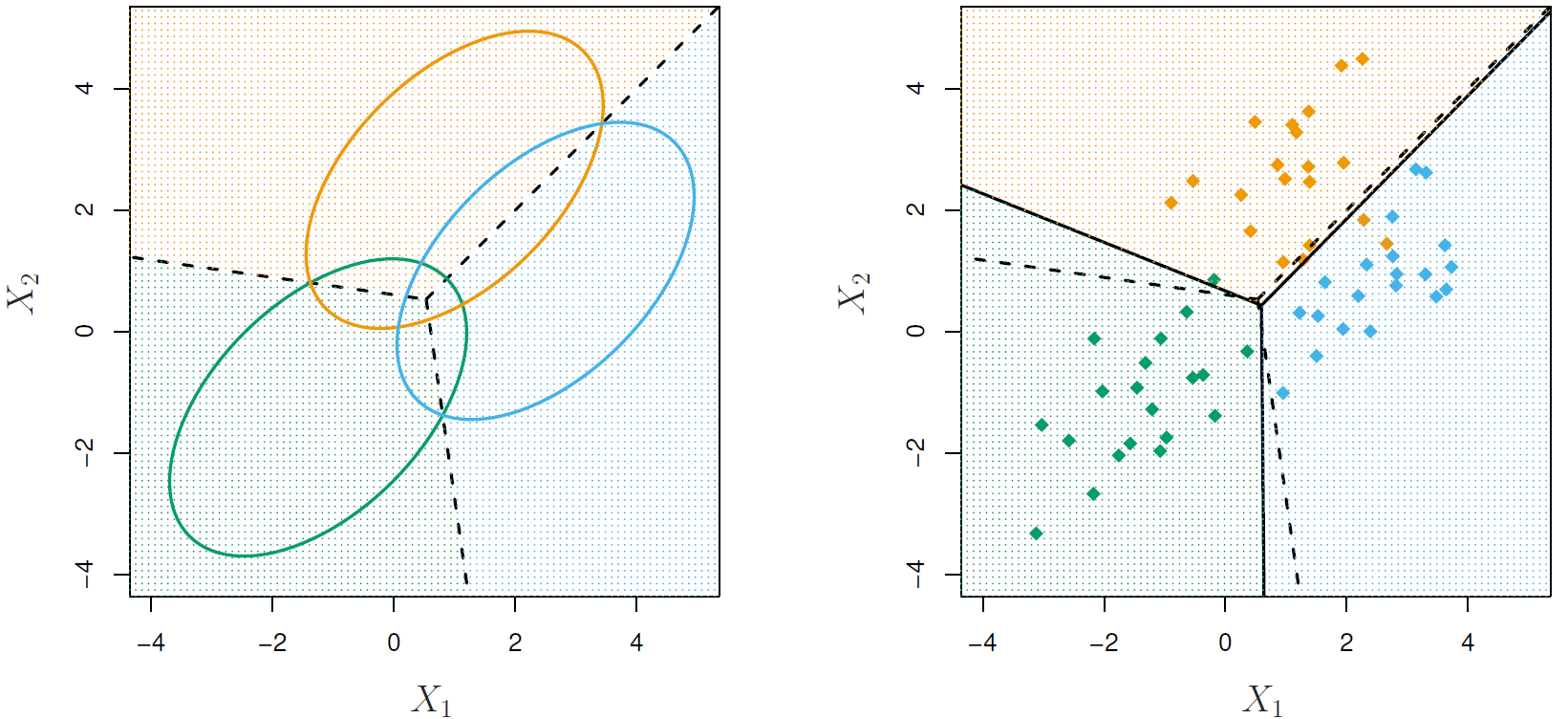
* With multiple predictors, we will assume that is drawn from a multivariate Gaussian distribution, with a class-specific mean vector and a common covariance matrix. Two multivariate Gaussian density functions are shown below. Left: The two predictors are uncorrelated. Right: The two variables have a correlation of 0.7.



* To indicate that a p-dimensional random variable X has a multi-variate Gaussian distribution, we write . Here is the mean of X (a vector with p components), and is the covariance matrix of X. Formally, the multivariate Gaussian density is defined as
* A bit of algebra reveals that the Bayes classifier assigns an observation to the class for which

is largest.

* An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with p = 2, with a class-specific mean vector and a common covariance matrix. Left: Ellipses that contain 95% of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries. Right: 20 observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.

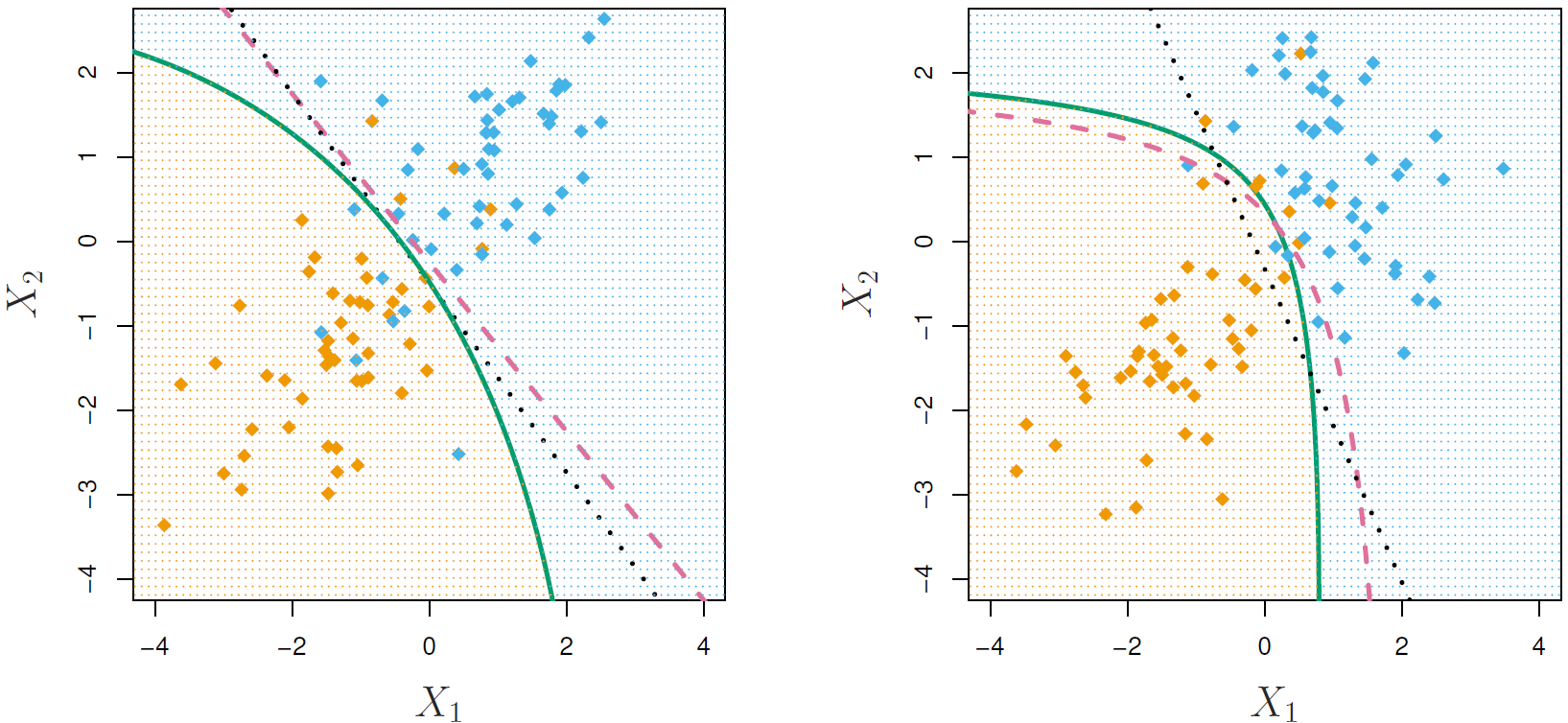


### Quadratic Discriminant Analysis

* Like LDA, the QDA classifier results from assuming that the observations from each class are drawn from a Gaussian distribution, and plugging estimates for the parameters into Bayes’s theorem in order to perform prediction. However, unlike LDA, QDA assumes that each class has its own covariance matrix. That is, it assumes that an observation from the kth class if of the form , where is a covariance matrix for the kth class. Under this assumption, the Bayes classifier assigns an observation to the class for which

is largest.

* If LDA’s assumption that that K classes share a common covariance matrix is badly off, then LDA can suffer from high bias.
* Regarding below chart, left: the Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with . The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA. Right: Details are as given in the left-hand panel, except that . Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.



### A Comparison of Classification Methods

* LDA outperforms logistic regression if the distribution of predictors is reasonably MNV (with constant covariance).
* QDA outperforms LDA if the covariance is not the same among groups.
* K-NN outperforms the others if the decision boundary is extremely non-linear.
* In order of computational speed (generally speaking, it depends on K, p and n of course):
  + LDA/QDA is fast because it only needs the mean and covariance matrix, while logistic regression has to go through the iterative process to find the betas. When you have lots of data (like streaming data), LDA/QDA might be the only way to go.
  + Logistic regression is the best if you want to understand inference.

### Confusion Matrix

* With the following confusion matrix:

|  |  |  |
| --- | --- | --- |
|  | Predict 1 | Predict 0 |
| Actual 1 | True Positive (TP) | False Negative (FN) |
| Actual 0 | False Positive (FP) | True Negative (TN) |

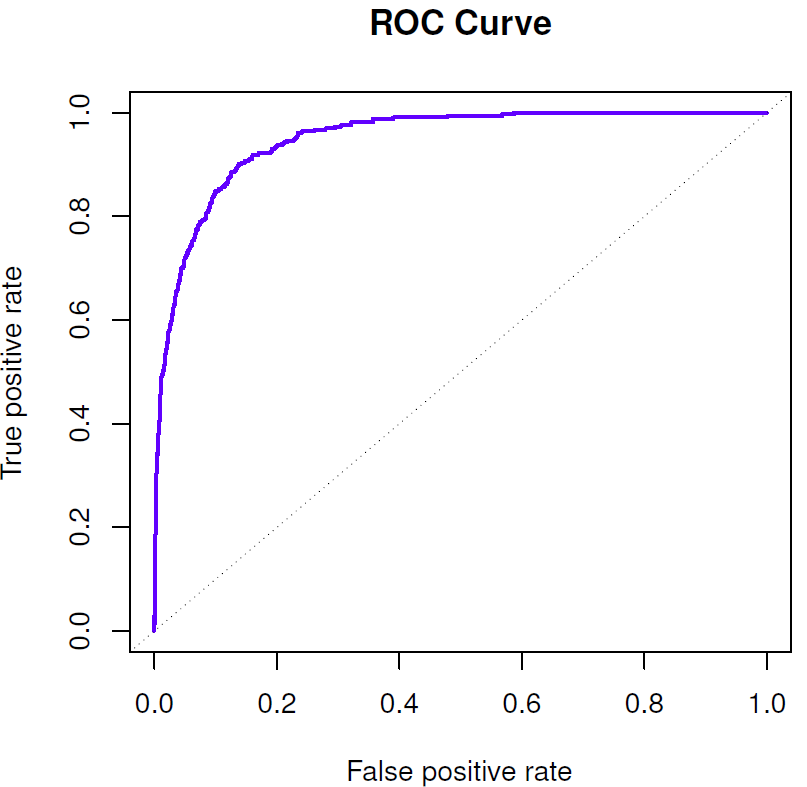
* Precision: Among the predictions that I predict to be 1, how many of them are correct?
* Recall: Among the actual observations that are 1, how many of them that I predict correctly?
* F1-score:

Harmonic mean: If either precision or recall is low, F1-score will be low.

* Accuracy:

### ROC Curve

* ROC (Receiver Operating Characteristics) curve is a popular graphic for simultaneously displaying the two types of errors for all possible thresholds. The overall performance of a classifier, summarized over all possible thresholds, is given by the area under the (ROC) curve (AUC). An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier.



# Resampling Methods

## Cross-Validation

### Leave-one-out Cross-Validation

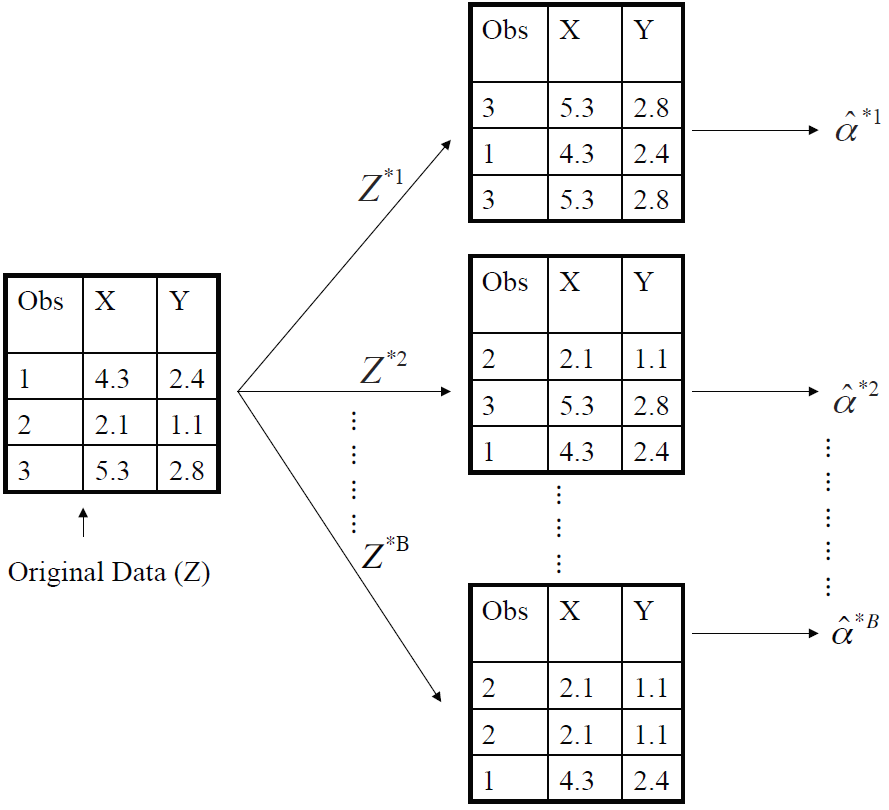
* The statistical learning method is fit on the n – 1 training observations, and a prediction is made for the excluded observation. The LOOCV estimate for the test MSE is the average of these n test error estimates:

### k-Fold Cross-Validation

* This approach randomly divides the set of observations into k groups, or folds, of approximately equal size.

## Bootstrap

* Bootstrap is used to quantify the uncertainty associated with a given estimator or statistical learning method.



* Above chart is a graphical illustration of the bootstrap approach on a small sample containing n=3 observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of .

# Linear Model Selection and Regularization

## Subset Selection

### Best Subset Selection

* Algorithm:
  + Let denote the null model, which contains no predictors. This model simply predicts the sample mean for each observation.
  + For :
    - Fit all models that contain exactly k predictors.
    - Pick the best among these models, and call it . Here best is defined as having the smallest RSS, or equivalently largest .
  + Select a single best model from among using cross-validated prediction error, AIC, BIC or adjusted .
* It needs to fit models in total.

### Forward Stepwise Selection

* Algorithm:
  + Let denote the null model, which contains no predictors. This model simply predicts the sample mean for each observation.
  + For :
    - Consider all models that augment the predictors in with one additional predictor.
    - Choose the best among these models, and call it . Here best is defined as having smallest RSS or highest .
  + Select a single best model from among using cross-validated prediction error, AIC, BIC or adjusted .
* It needs to fit models in total, with significant computational advantage over best subset selection.
* Though forward stepwise tends to do well in practice, it is not guaranteed to find the best possible model out of all models containing subsets of the p predictors.
* Similarly, we have backward stepwise selection, and hybrid selection which combined forward and backward methods.

## Choosing the Optimal Model

### AIC, BIC and Adjusted

* AIC for least squares model containing predictors:

where n is the number of observations, is an estimate of the variance of the error associated with each response measurement.

* BIC for least squares model containing predictors:

The BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models than AIC.

* Adjusted for least squares model containing predictors:

### Validation and Cross-Validation

* In the past, performing cross-validation was computationally prohibitive for many problems with large and/or large , and so AIC, BIC and adjusted were more attractive approaches for choosing among a set of models. However, nowadays with fast computers, the computations required to perform cross-validation are hardly ever an issue. Thus, cross-validation is a very attractive approach for selecting from among a number of models under consideration.

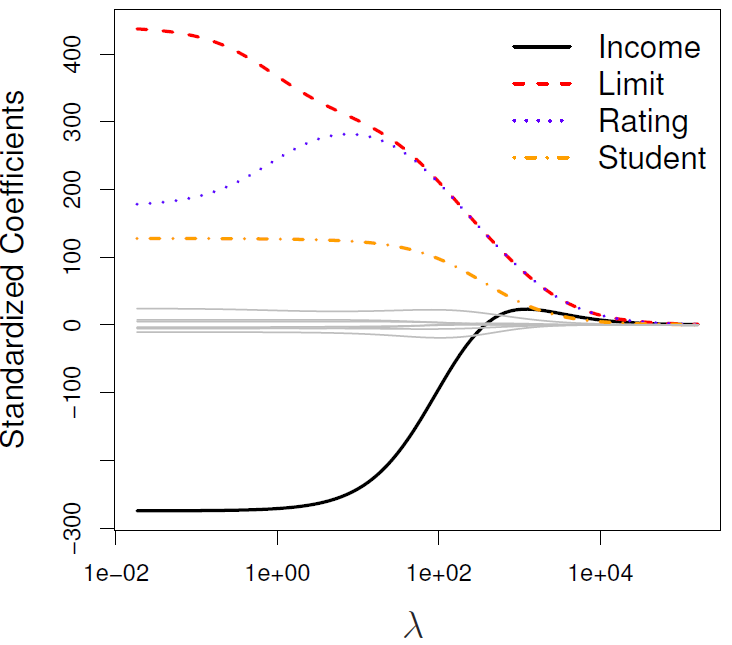
## Shrinkage Methods

### Ridge Regression

* The ridge regression coefficient estimates are the values that minimize

where is a tuning parameter, to be determined separately.

* Note that the shrinkage penalty is applied to , but not to the intercept . We want to shrink the estimated association of each variable with the response; however, we do not want to shrink the intercept, which is simply a measure of the mean value of the response when . If we assume that the variables – that is, the columns of the data matrix X – have been centered to have mean zero before ridge regression is performed, then the estimated intercept will take the form .
* Because of the existence of , it is best to standardize the predictors (z-score) before ridge regression.



* Solution to ridge regression:
* Algorithm:
  + Remove from data
  + Split the rest of the data into K folds
  + For k in :
    - For in :
      * Determine the that minimizes using the train data of the fold
      * Record the using the validation data of the fold .

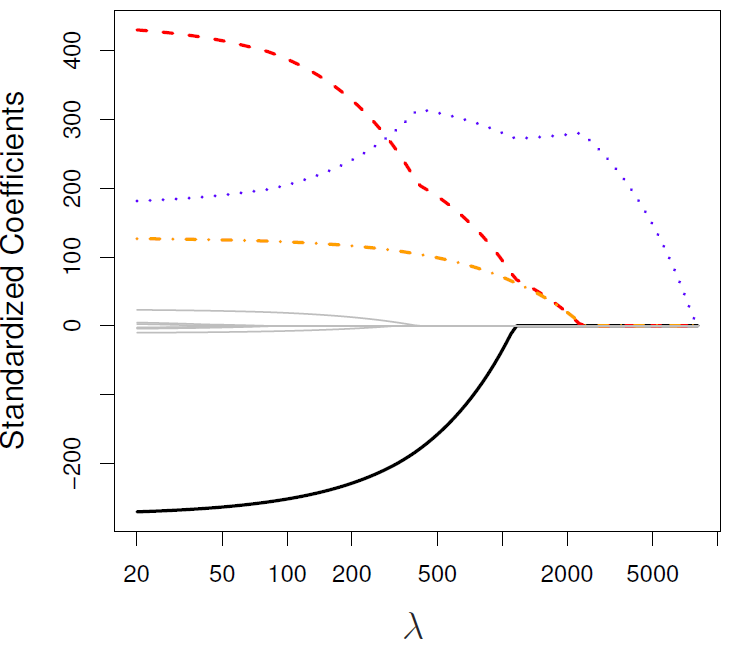
At this point, we have a 2-D matrix, rows are for different , and columns are for different values.

* + Average the for each , .
  + Find the that minimizes the , resulting to .
  + Refit the model using the full training data , resulting to .
  + Report or on given the .

### Lasso Regression

* The lasso regression coefficient estimates are the values that minimize

where is a tuning parameter, to be determined separately.

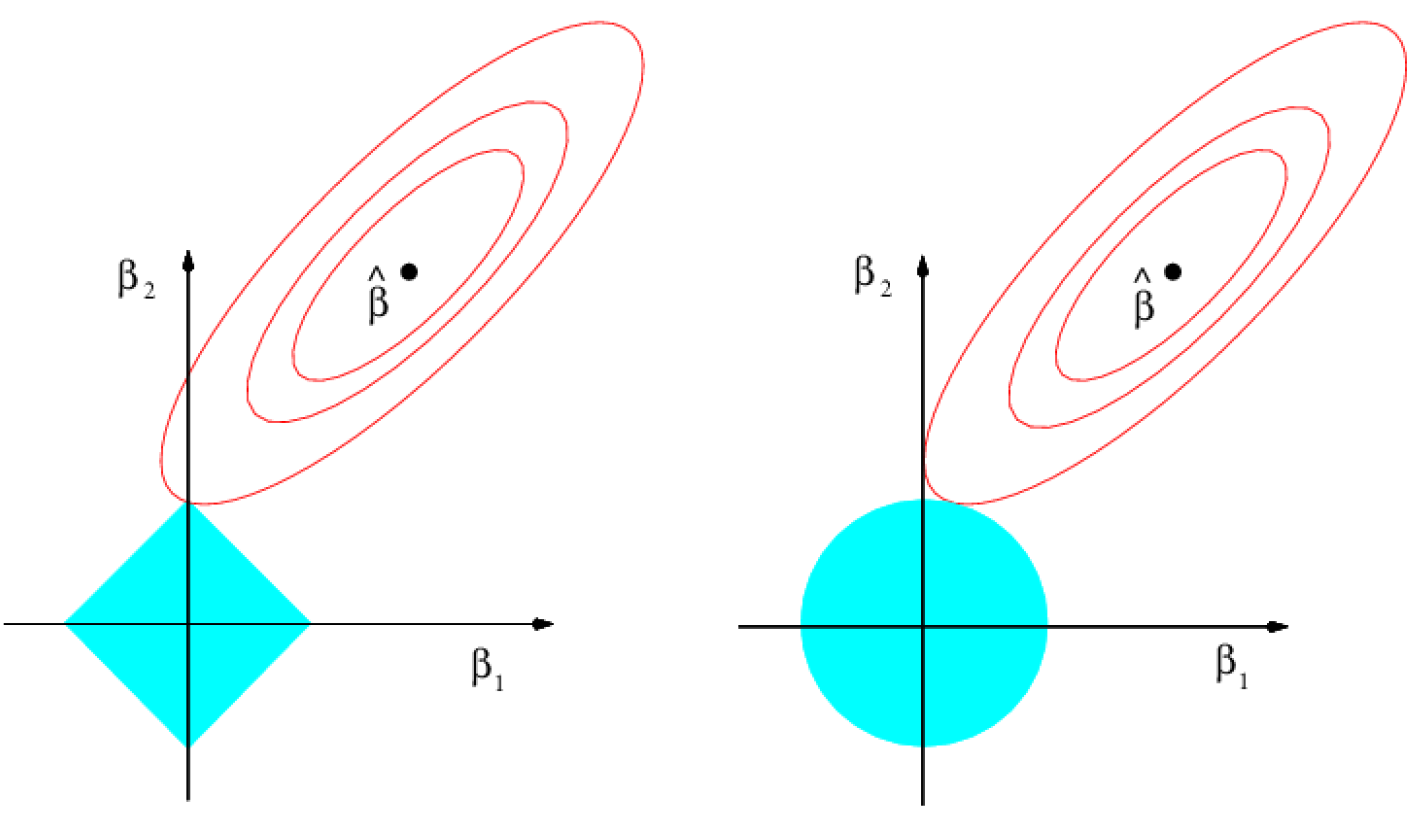


### Compare Ridge and Lasso Regression

* One can show that ridge and lasso regression coefficient estimates solve the problems

and

* The below figure shows the variable selection property of lasso. The least squares solution is marked as , while the blue diamond and circle represent the lasso and ridge regression constraints, respectively. If is sufficiently large, then the constraint regions will contain , and so the ridge regression and lasso estimates will be the same as the least squares estimates. (Such a large value of corresponds to .
* The ellipses that are centered around represent regions of constant RSS. In other words, all of the points on a given ellipse share a common value of RSS. The above two equations indicate that the lasso and ridge regression estimates are given by the first point at which an ellipse contacts the constraint region. The lasso constraint has corners at each of the axes, and so the ellipse will often intersect the constraint point at an axis. When this occurs, one of the coefficients will equal zero.



* Computationally, solving ridge is faster than solving lasso, since ridge has a closed form solution, while lasso has to be solved by a solver.

## Dimension Reduction Methods

* Let represent linear combinations of our original predictors. That is,

for some constants . We can then fit the linear regression model

using least squares.

* The term dimension reduction comes from the fact that this approach reduces the problem of estimating coefficients to the simpler problem of estimating the coefficients.

### Principal Components Regression

* PCA is a technique for reducing the dimension of a data matrix . The first principal component direction of the data is that along which the observations vary the most.
* The principal components regression (PCR) approach involves constructing the first principal components, , and then using these components as the predictors in a linear regression model that is fit using least squares.
* When performing PCR, we generally recommend standardizing each predictor, prior generating the principal components. This standardization ensures that all variables are on the same scale. In the absence of standardization, the high-variance variables will tend to play a larger role in the principle components obtained, and the scale on which the variable are measured will ultimately have an effect on the final PCR model.

# Moving Beyond Linearity

## Polynomial Regression

* Replace the standard linear model with a polynomial function:

## Step Functions

* Using polynomial functions of the features as predictors in a linear model imposes a global structure on the non-linear function of X. We can instead use step functions in order to avoid imposing such a global structure. Here we break the range of X into bins, and fit a different constant at each bin. This amounts to converting a continuous variable into an ordered categorical variable.
* In greater detail, we create cut points in the range of X, and then constructs new variables

…

where is an indicator function that returns a 1 if the condition is true, and returns 0 otherwise. These are sometimes called dummy variables. Notice that for any value of X, , since X must be in exactly one of the intervals.

* We then use least squares to fit a linear model using as predictors:

We exclude as a predictor because it is redundant with the intercept. Alternatively, we could include , and exclude the intercept.

## Basis Functions

* Polynomial and piecewise-constant regression models are in fact special cases of a basis function approach. The idea is to have at hand a family of functions or transformations that can be applied to a variable X.

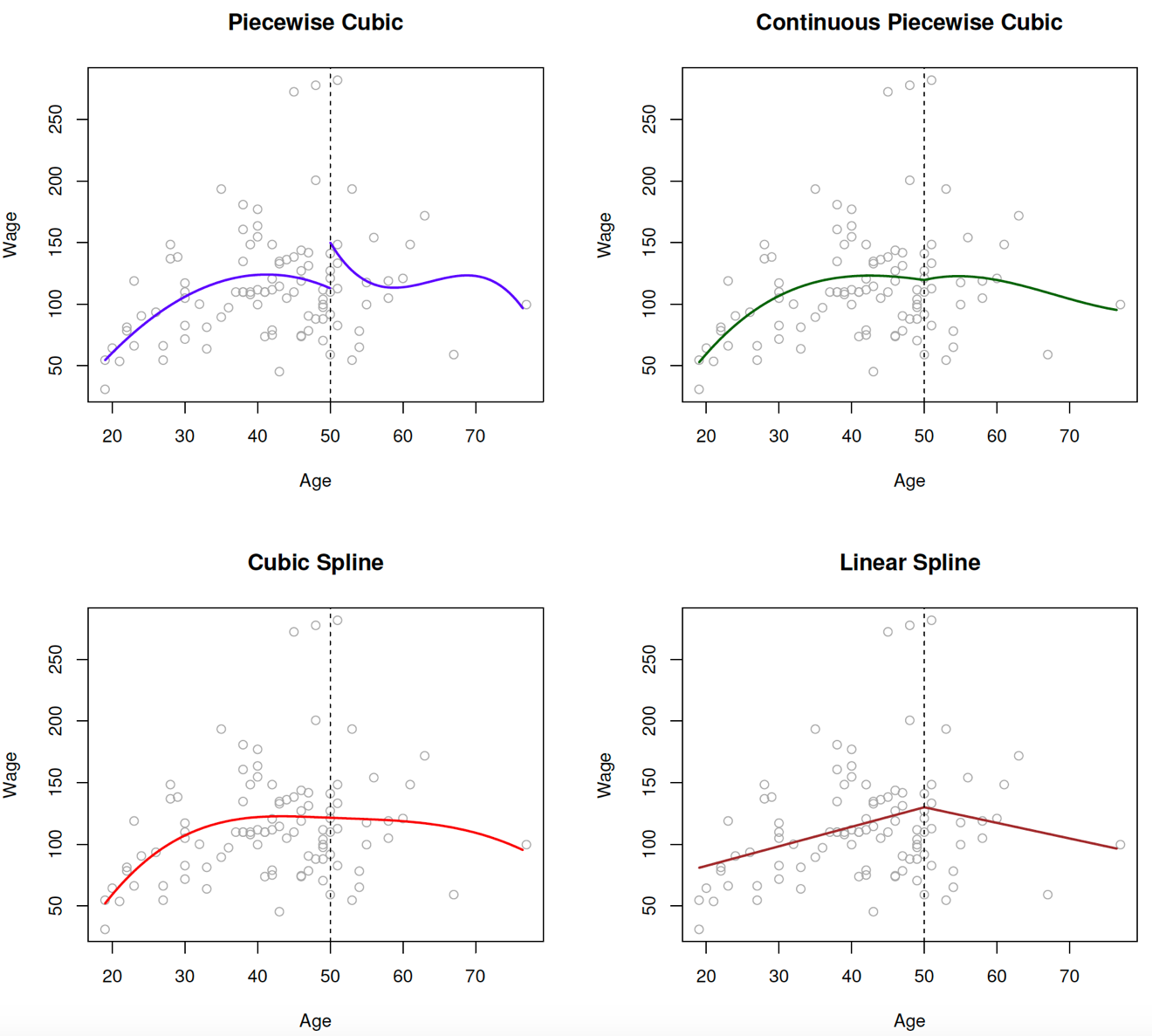
For polynomial regression, the basis functions are , and for piecewise constant functions they are .

## Regression Splines

### Piecewise Polynomials

* A piecewise cubic polynomial with a single knot at a point c takes the form

In other words, we fit two different polynomial functions to the data, one on the subset of the observations with , and one on the subset of the observations with .



### Constraints and Splines

* The top left panel of above figure looks wrong because the fitted curve is just too flexible. To remedy this problem, we can fit piecewise polynomial under the constraint that the fitted curve must be continuous. In other words, there cannot be a jump at the knot. The top right plot shows the resulting fit. This looks better than the top left plot, but the V-shaped join looks unnatural.
* In the lower left plot, we have added two additional constraints: now both the first and the second derivatives of the piecewise polynomials are continuous. The curve in the bottom left plot is called a cubic spline. In general, a cubic spline with K knots uses a total of degrees of freedom.

### The Spline Basis Representation

* A cubic spline with K knots can be modeled as

for an appropriate choice of basis functions, .

* The most direct way to represent a cubic spline is to start off with a basis for a cubic polynomial – namely, – and then add one truncated power basis function per knot. A truncated power basis function is defined as

where is the knot. One can show that adding a term of the form to the model for a cubic polynomial will lead to a discontinuity in only the third derivative at ; the function will remain continuous, with continuous first and second derivatives, at each of the knots.

* In other words, in order to fit a cubic spline to a data set with K knots, we perform least squares regression with an intercept and predictors, of the form , where are the knots. This amounts to estimating a total of regression coefficients; for this reason, fitting a cubic spline with knots uses degrees of freedom.
* A natural spline is a regression spline with additional boundary constraints: the function is required to be linear at the boundary (in the region where X is smaller than the smallest knot, or larger than the largest knot). The additional constraint means that natural splines generally produce more stable estimates at the boundaries.
* In practice, it is common to place knots in a uniform fashion.

## Smoothing Splines

* In fitting a smooth curve to a set of data, what we really want to do is find some function, say , that fits the observed data well: that is, we want to be small. However, there is a problem with this approach. If we don’t put any constraints on , then we can always make RSS zero simply by choosing g such that it interpolates all of the . Such a function would woefully overfit the data – it would be far too flexible. What we really want is a function g that makes RSS small, but that is also smooth.
* A natural approach is to find the function g that minimizes

where is a nonnegative tuning parameter. The function g that minimizes the above function is known as a smoothing spline.

* The function that minimizes the above function can be shown to have some special properties: it is a piecewise cubic polynomial with knots at the unique values of , and continuous first and second derivatives at each knot. Furthermore, it is linear in the region outside of the extreme knots. In other words, the function that minimizes the above objective function is a natural cubic spline (shrunken version) with knots at .

## Local Regression

* Local regression at
  + Gather the fraction of training points whose are closest to .
  + Assign a weight to each point in this neighborhood, so that the point furthest from has weight zero, and the closest has the highest weight. All but these nearest neighbors get weight zero.
  + Fit a weighted least squares regression of the on the using the aforementioned weights, by finding and that minimize
  + The fitted value at is given by

## Generalized Additive Models

* Generalized additive model:

This is an example of a GAM. It is called an additive model because we calculate a separate for each , and then add together all of their contributions.

* Pros and cons of GAMs:
  + Pro: GAMs allow us to fit a non-linear to each , so that we can automatically model non-linear relationships that standard linear regression will miss. This means that we don’t need to manually try out many different transformations on each variable individually.
  + Pro: The non-linear fits can potentially make more accurate predictions for the response Y.
  + Pro: Because the model is additive, we can still examine the effect of each on Y individually while holding all of the other variables fixed. Hence if we are interested in inference, GAMs provide a useful representation.
  + Con: The main limitation of GAMs is that the model is restricted to be additive.

# Tree-Based Methods

## The Basics of Decision Trees

### Regression Trees

* Two steps of building a regression tree:
  + We divide the predictor space – that is, the set of possible values for – into J distinct and non-overlapping regions, .
  + For every observation that falls into the region , we make the same prediction, which is simply the mean of the response values for the training observations in .
* How do we construct the regions ?
  + In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model. The goal is to find boxes that minimize the RSS, given by

where is the mean response for the training observations within the jth box.

* + In order to perform recursive binary splitting, we first select the predictor and the cut point s such that splitting the predictor space into the regions and leads to the greatest possible reduction in RSS. That is, we consider all predictors , and all possible values of the cut point s for each of the predictors, and then choose the predictor and cut point such that the resulting tree has the lowest RSS. In greater detail, for any j and s, we define the pair of half-planes

and

and we seek the value of j and s that minimize the equation

* Building a regression tree:
  + Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
  + Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of .
  + Use K-fold cross-validation to choose . That is, divide the training observations into K folds. For each :
    - Repeat steps 1 and 2 on all but the kth fold of the training data.
    - Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of
  + Average the results for each value of , and pick to minimize the average error. For each value of , there corresponds a subtree such that

is as small as possible. Here indicates the number of terminal nodes of the tree , is the rectangle corresponds to the mth terminal node, and is the predicted response associated with – that is the mean of the training observations in .

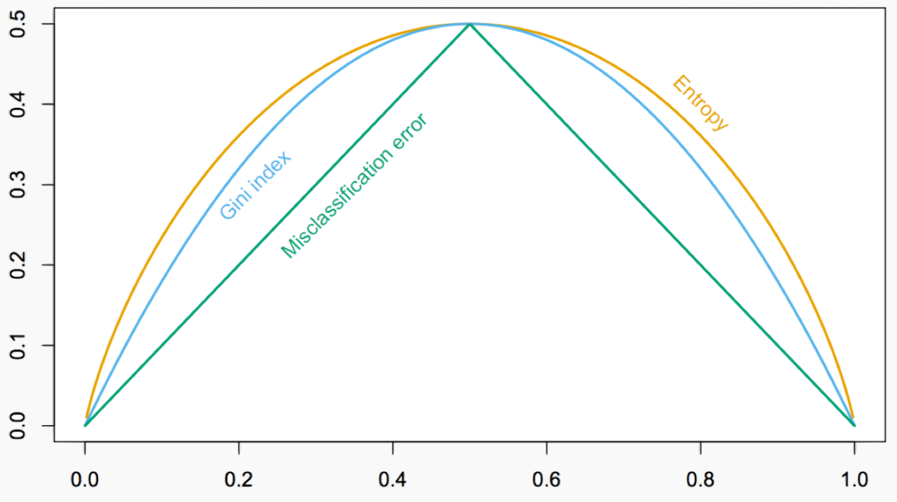
* + Return the subtree from step 2 that corresponds to the chosen value of
* Stopping conditions:
  + Don’t split a region if all instances in the region belong to the same class.
  + Don’t split a region if the number of instances in the sub-region will fall below pre-defined threshold.
  + Don’t split a region if the total number of leaves in the tree will exceed pre-defined threshold.
  + Don’t split if the gain in purity of splitting a region is less than some pre-defined threshold.

### Classification Trees

* In classification setting, classification error rate is the criterion for making the binary splits. The classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class:

Here represents the proportion of training observations in the mth region that are from the kth class. The problem with it is that it’s not differentiable, so Gini index and cross-entropy are better node purity measures.

* Gini index:
* Cross-entropy:



## Bagging, Random Forests, Boosting, Stacking

### Bagging

* Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method. A natural way to reduce the variance and increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.
* To apply bagging to regression trees, we simply construct B regression trees using B bootstrapped training sets, and average the resulting predictions. These trees are grown deep, and are not pruned. Hence each individual tree has high variance, but low bias. Averaging these B trees reduces the variance. Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

#### Number of Trees

* Using a large number of trees is not a critical parameter with bagging, as it will not lead to overfitting. In practice we use a value of B sufficiently large that the error has settled down.
* However, if the number of trees is too large, then the trees in the ensemble may become more correlated, increase the variance.

#### Out-of-Bag Error Estimation

* On average, each bagged tree makes use of around two-thirds of the observations. The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations. We can predict the response for the ith observation using each of the trees in which that observation was OOB. It can be shown that with B sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error. The OOB approach for estimating the test error is particularly convenient when performing bagging on large data sets for which cross-validation would be computationally onerous.

#### Variable Importance Measures

* When we bag a large number of trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree, and it is no longer clear which variables are most important to the procedure. Thus, bagging improves prediction accuracy at the expense of interpretability.
* We can record the total amount that the RSS for bagging regression trees or Gini index for bagging classification trees is decreased due to splits over a given predictor, average over all B trees. A large value indicates an important predictor.

### Random Forests

* Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors. Then in the collection of a bagged trees, most or all the trees will use this strong predictor in the top split. Consequently, all of the trees will look quite similar to each other. Hence the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.
* Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors (typically, ) is chosen as split candidates from the full set of p predictors. In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors.
* **Q:** When the number of predictors is large, but the number of relevant predictors is small, random forests can perform poorly. Why?

**A:** In each split, the chances of selected a relevant predictor will be low and hence most trees in the ensemble will be weak models.

* Hyper-parameters to tune:
  + The number of predictors to randomly select at each split.
  + The total number of trees in the ensemble.
  + The minimum leaf node size.

### Boosting

#### Gradient Boosting

##### The algorithm

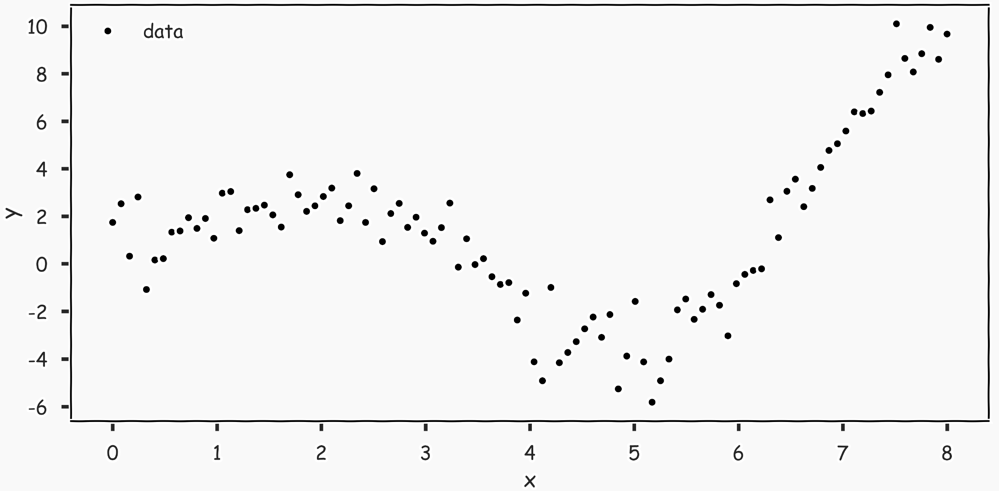
* Gradient boosting is a method for iteratively building a complex regression model T by adding simple models. Each new simple model added to the ensemble compensates for the weaknesses of the current ensemble.
  + Fit a simple model, , on the training data

Set . Compute the residuals for .

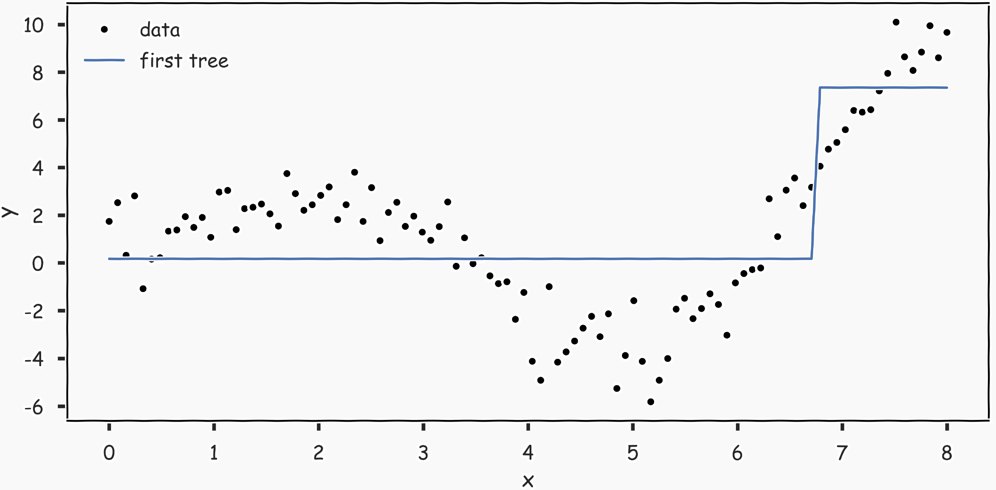
* Fit a simple model, , to the current residuals, i.e. train using
* Set
* Compute residuals, set , where is called the learning rate.
  + Repeat steps 2-4 until stopping condition met.
* The idea is that we will use a bunch of weak “learner” (aka, models) which are fit sequentially. The 1st one fits the signal, the 2nd one fits the 1st model’s residuals, and the 3rd one fits the 2nd model’s residuals, and so on. At each stage, we upweight the places where our previous model did badly on.
* Choosing the learning rate:
  + If is a constant, then it should be tuned through cross validation.
  + For better results, let the value of depend on the gradient, so that
    - is small, when the gradient is small, around the optimum.
    - is large, when the gradient is large, far from the optimum.
* Implementations:
  + XGBoost: An efficient Gradient Boosting model
  + LGBM: Light Gradient Boosted Machines. It is a library for training GBMs developed by Microsoft, and it competes with XGBoost.
  + CatBoost: A new library for Gradient Boosting Decision Trees, offering appropriate handling of categorical features.

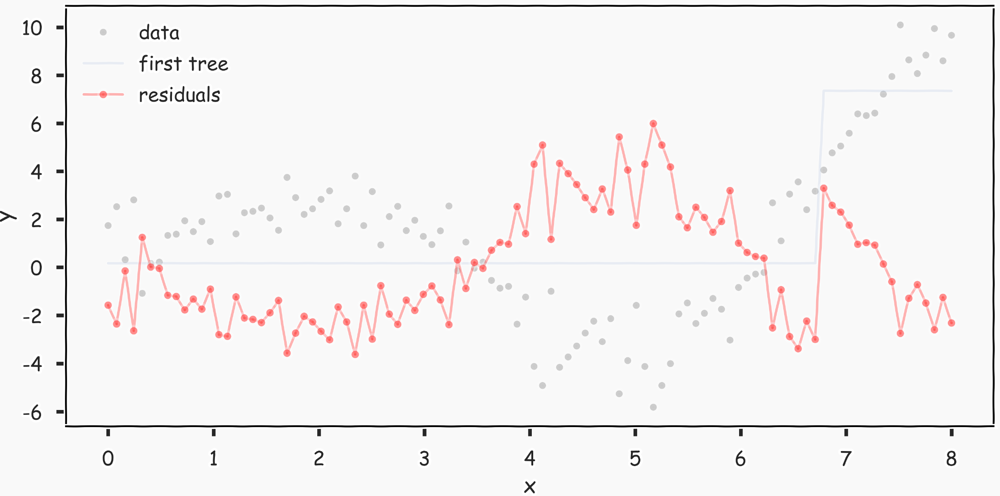
##### The Illustration

* Raw data:

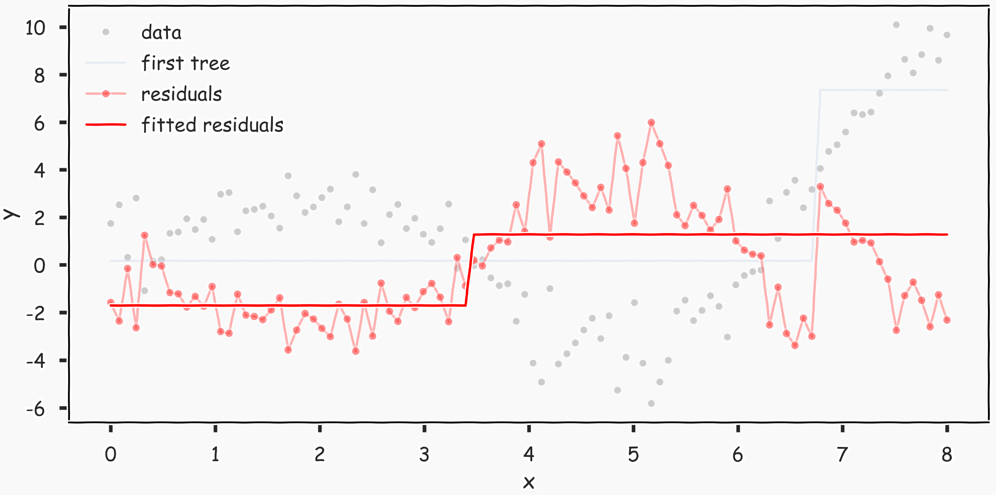


* 1st tree with one split that minimizes the MSE:

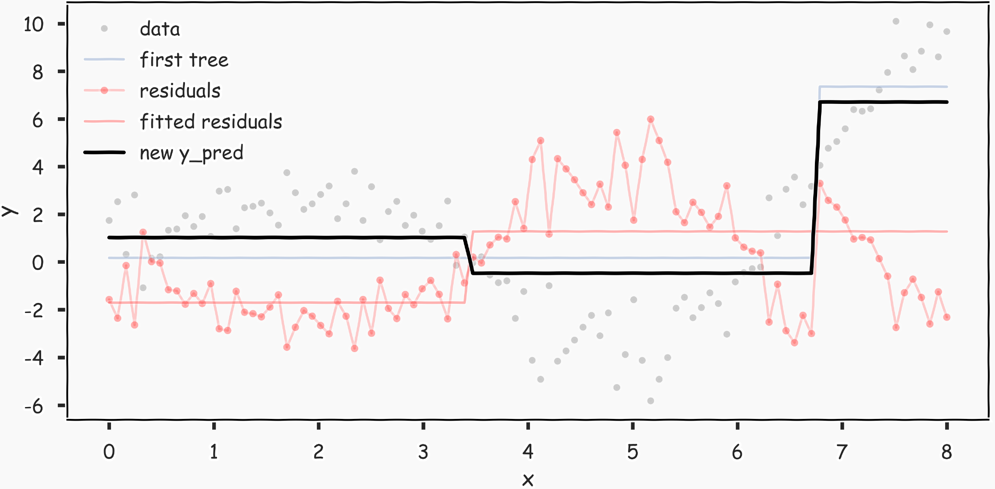




* 2nd tree fitted on the residuals:



New predictions = previous prediction + lambda \* current predictions on the residuals



#### AdaBoost

* In classification, we typically want to minimize the classification error:

Naively, we can try to minimize error via gradient descent, just like we did for MSE in gradient boosting. Unfortunately, error is not differentiable with respect to the predictions.

* Solution is to replace the error function with a differentiable function that is a good indicator of classification error. The function we choose is called exponential loss.

Exponential loss is differentiable with respect to and it is an upper bound of error. If the model did everything wrong, then .

* We first compute the gradient for :

It’s easier to decompose each as , where . This way, we see that the gradient is just a re-weighting applied the target values

Notice that when , the weight is small; when , the weight is large.

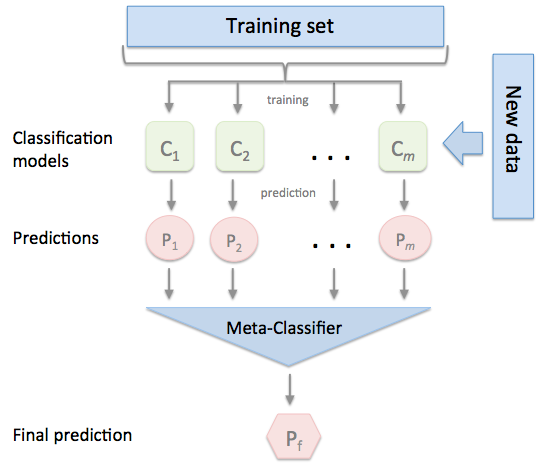
* With a minor adjustment to the exponential loss function, we have the algorithm for gradient descent:
  + Choose an initial distribution over the training data,
* At the step, fit a simple classifier on weighted training data
* Update the weights:

where Z is the normalizing constant for the collection of updated weights.

* Update , where is the learning rate.
* Choosing the learning rate:
  + Unlike in the case of gradient boosting for regression, we can analytically solve for the optimal learning rate for AdaBoost, by optimizing:
  + Doing so, we get that

### Stacking

* Ensemble methods:
  + Bagging:
    - Create an ensemble of full trees, each trained on a bootstrapped sample of training set
    - Average the predictions
    - Parallel model building
  + Random Forest:
    - Create an ensemble of full trees, each trained on a bootstrapped sample of training set.
    - In each tree and each split, randomly select a subset of predictors, choose a predictor from this subset for splitting
    - Average the predictions
    - Parallel model building
  + Boosting:
    - Iteratively build a model from lots of little models.
    - Each subsequent model predicts the residuals from the previous model, overweighting the large residuals.
    - Sequential model building
  + Neural Network:
    - Build layers of models based on overly simple ‘neurons’ of models
    - Uses back-propagation to efficiently communicate between output of models to update earlier models.
    - Sequential model building
  + Stacking
* Stacking
  + Stacking is a way to generalize the ensemble approach to combine outputs of various types of model, and improves on the combination as well.



# Support Vector Machines

## Maximal Margin Classifier

* We can compute the perpendicular distance from each training observation to a given separating hyperplane; the smallest distance is called the margin. The maximal margin hyperplane is the separating hyperplane for which the margin is largest – that is, it is the hyperplane that has the farthest minimum distance to the training observations.
* The observations that are closest to the hyperplane are called support vectors. The hyperplane depends directly on the support vectors, but not on the other observations.
* Briefly, the maximal margin hyperplane is the solution to the optimization problem:

One can show that with constraint, the perpendicular distance from the ith observation to the hyperplane is given by .

## Support Vector Classifier

* We might be willing to consider a classifier based on a hyperplane that does not perfectly separate the two classes, in the interest of
  + Greater robustness to individual observations, and
  + Better classification of most of the training observations.
* It is the solution to the optimization problem:
* is treated as a tuning parameter that is generally chosen via cross-validation. As with the tuning parameters that we have seen, controls the bias-variance trade-off of the statistical learning technique. When is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the date, which may have low bias but high variance. On the other hand, when is larger, the margin is wider and we allow more violations to fit; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.

## Support Vector Machine

* The support vector machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in the specific way, using kernels. We want to enlarge our feature space in order to accommodate a non-linear boundary between the classes. The kernel approach that describe here is simply an efficient computational approach for enacting this idea.
* It turns out that the solution to the support vector classifier problem above involves only the inner products of the observations. The inner product of two p-vectors a and b is defined as . Thus the inner product of two observations is given by
* It can be shown that
  + The linear support vector classifier can be represented as

where there are n parameters , , one per training observation.

* + To estimate the parameters and , all we need are the inner products between all pairs of training observations.
* It turns out that is nonzero only for the support vectors in the solution – that is, if a training observation is not a support vector, then its equals zero. So if S is the collection of indices of these support points, we can rewrite any solution function of the form as

which typically involves far fewer terms.

* Now suppose that every time the inner product appears in the representation, or in the classification of the solution for the support vector classifier, we replace it with a generalization of the inner product of the form

where K is some function that we will refer to as a kernel. A kernel is a function that quantifies the similarity of two observations. For instance, we could simply take

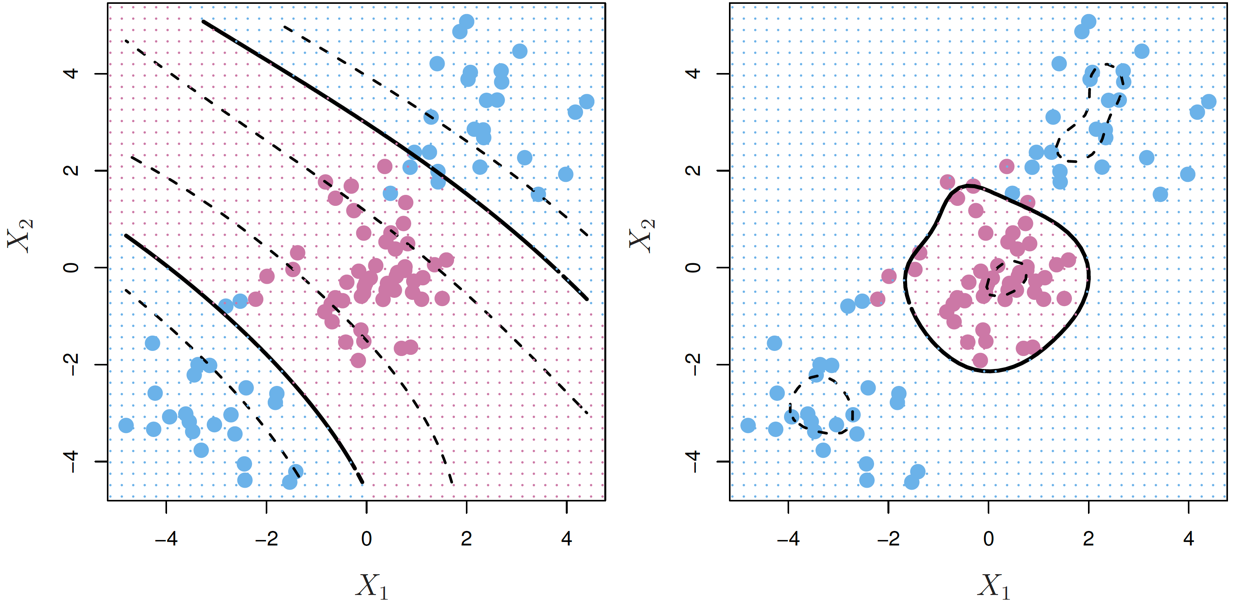
which would just give us back the support vector classifier. The above equation is known as a linear kernel because the support vector classifier in the features; the linear kernel essentially quantifies the similarity of a pair of observations using Pearson (standard) correlation.

* A polynomial kernel of degree d, where d is a positive integer.

Left chart below.

* A radial kernel, which takes the form

Right chart below.



# Unsupervised Learning

## Principal Components Analysis

* The first principal component of a set of features is the normalized linear combination of the features

that has the largest variance. By normalized, we mean that . We refer to the elements as the loadings of the first principal components; together, the loadings make up the principal component loading vector, . We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance.

* In other words, the first principal component loadings vector solves the optimization problem

The second principal component is the linear combination of that has maximal variance out of all linear combinations that are uncorrelated with .

* The principal components directions are the ordered sequence of eigenvectors of the matrix , and the variances of the components are the eigenvalues.
* The proportion of variance explained:
  + The total variance present in a data set (assuming that the variables have been centered to have mean zero) is defined as

and the variance explained by the principal component is

* + Therefore, the PVE of the principal component is given by

## Clustering Methods

### K-Means Clustering

* The idea behind K-means clustering is that a good clustering is one for which the cluster variation is as small as possible. The within-cluster variation for cluster is a measure of the amount by which the observations within a cluster differ from each other. Hence we want to solve the problem

In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

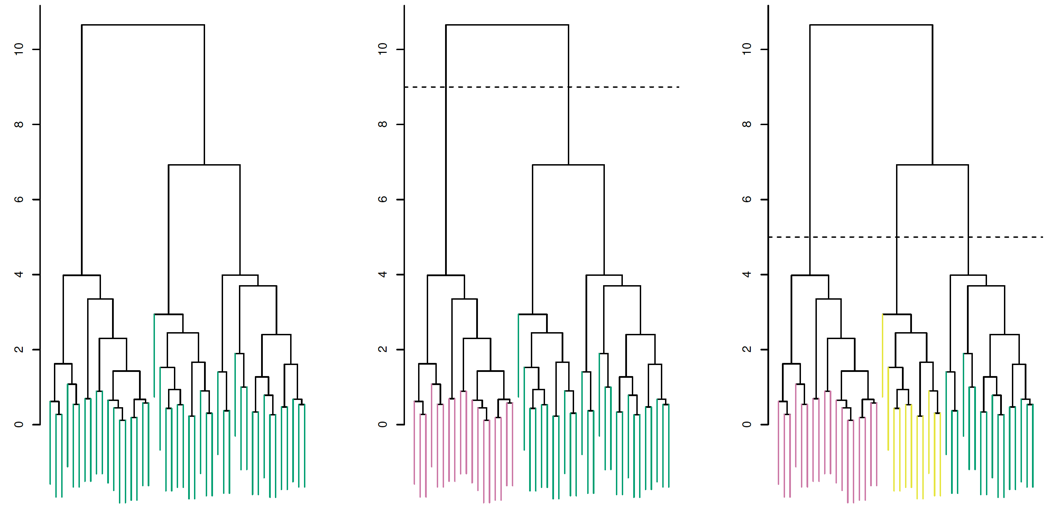
* There are many possible ways to define this concept, but by far the most common choice involves squared Euclidean distance. That is, we define

where denotes the number of observations in the kth cluster. In other words, the within-cluster variation for the kth cluster is the sum of all of the pairwise squared Euclidean distances between the observation in the kth cluster, divided by the total number of observations in the kth cluster.

* Algorithm:
  + Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
  + Iterate until the cluster assignments stop changing:
    - For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
    - Assign each observation to the cluster whose centroid is closest (where closest is defined using Euclidean distance).
* When the above result no longer changes, a local minimum has been reached. It is important to run the algorithm multiple times from different random initial configurations. Then one selects the best solution, i.e. that for which the objective is smallest.

### Hierarchical Clustering

* Potential disadvantage of K means clustering is that it requires us to pre-specify the number of clusters K. Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of K. Hierarchical clustering has an added advantage over K-means clustering in that it results in an attractive tree-based representation of the observations, called a dendrogram.
* Observations that fuse at the very bottom of the tree are quite similar to each other, whereas observations that fuse close to the top of the tree will tend to be quite different.
* We cannot draw conclusions about the similarity of two observations based on their proximity along the *horizontal axis*. Rather, we draw conclusions about the similarity based on the location on the *vertical axis* where branches containing those two observations first are fused.



* We make a horizontal cut across the dendrogram, as shown in the center and right-hand panels above. The distinct sets of observations beneath the cut can be interpreted as clusters. The height of the cut to the dendrogram serves the same role as the K in the K-means clustering: it controls the number of clusters obtained.
* The term hierarchical refers to the fact that clusters obtained by cutting the dendrogram at a given height are necessarily nest within the clusters obtained by cutting the dendrogram at any greater height. However, on an arbitrary data set, this assumption of hierarchical structure might be unrealistic. For instance, suppose that our observations correspond to a group of people with an 50-50 split of males and females, evenly split among Americans, Japanese, and French. We can imagine a scenario in which the best division into two groups might split these people by gender, and the best division into three groups might split them by nationality. In this case, the true clusters are not nested, in the sense that the best division into three groups does not result from taking the best division into two groups and splitting up one of those groups. Consequently, this situation could not be well-represented by hierarchical clustering. Due to situations such as this one, hierarchical clustering can sometimes yield worse results than K-means clustering for a given number of clusters.
* Algorithm:
  + Begin with n observations and a measure (such as Euclidean distance) of all the pairwise dissimilarities. Treat each observation as its own cluster.
  + For
    - Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
    - Compute the new pairwise inter-cluster dissimilarities among the n-1 remaining clusters.
* A summary of the four most commonly-used types of linkage in the hierarchical clustering:

|  |  |
| --- | --- |
| Linkage | Description |
| Complete | Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the largest of these dissimilarities. |
| Single | Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observation are fused one-at-a-time. |
| Average | Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities. |
| Centroid | Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable inversions. |

# Time Series Models

## Autoregressive Process

* Let be . We say is autoregressive AR(p) process if for some constant parameters and , for all t:

represents “memory” or “feedback” of the past into the present value of the process, which determines the amount of feedback. represents “new information”.

This is a multiple linear regression model with lagged values of the time series as the X variables. The model can be expressed as:

where .

* Stationary condition:

## Moving Average Process

* A process is a moving average process if can be expressed as a weighted average (moving average) of the past values of the white noise.
* MA(q) Process:

## ARMA Process

* An ARMA(p,q) model combines both AR and MA terms and is defined by the equation:

## ARIMA Process

* Autoregressive integrated moving average process: A time series is said to be an ARIMA(p,d,q) process if is ARMA(p,q).
  + For example, if log returns on an asset are ARMA(p,q), then the log prices are ARIMA(p,1,q).
  + An ARIMA(p,d,q) is stationary only if d=0.
* Dickey-Fuller Test (Unit Root Test)
  + There is a unit root The process is not stationary.
  + It’s difficult to tell whether a time series is best modeled as stationary or nonstationary. To help decide between these two possibilities, it can be helpful to use hypothesis testing.

## Cointegration

* Spurious Regression:
  + Time series often have trends—either deterministic or stochastic. The [R2 statistic](http://en.wikipedia.org/wiki/Coefficient_of_determination) used in assessing adequacy of regressions gives substantially misleading results for time series with trends. To verify this, pick any consumption series for any country and regress it against GNP for some other, dissimilar country (for example, Fiji and Afghanistan). Unless you are unlucky, you will find a strong correlation, and a regression with very high R-squared will result. This is called [spurious regression](http://en.wikipedia.org/wiki/Spurious_correlation)—even though there is no relationship between the two series, the regression results suggests that there is a strong relationship. Cointegration helps to solve this problem.
* Cointegration:
  + Two time series are cointegrated if each is I(1) but there exists a such that is stationary. For example, the common trends model is:

Where and are nonzero, the trend common to both series is I(1), and the noise process and are I(0). If , then:

is free of the trend and therefore is I(0).

* Time Series Regression Methodology
  + Check all series involved for integration. If there are I(1) series on both sides of the regression relationship, then there is a possibility that you will get misleading results from running a regression.
  + So now check for co-integration between all the I(1) series. If this holds, this is a guarantee that the regression results you get are not spurious.

## ARCH Process

* is an autoregressive conditional heteroskedasticity ARCH(p) process if

where

is the conditional standard deviation of given the past values of this process.

## GARCH Process

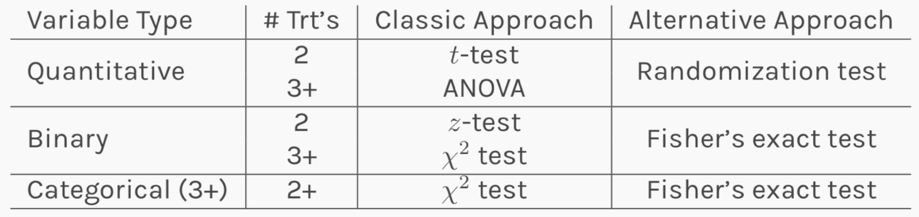
* The generalized autoregressive conditional heteroskedasticity GARCH(p, q) model is

where

* Because past values of the process are fed back into the present value, the conditional standard deviation can exhibit more persistent periods of high or low volatility than seen in ARCH process.

# AB Testing

* In the world of data science, performing experiments to determine causation, is called AB testing. AB-testing is often used in the tech industry to determine which form of website design leads to more ad clicks, purchases, etc… Or to determine the effect of a new app rollout on revenue or usage.
* Summary of analysis:



The classical approaches are typically parametric, based on some underlying distributional assumptions of the individual data, and work well for large n (or if those assumptions are actually true). The alternative approaches are nonparametric in that there is no assumptions of an underlying distribution, but they are slightly less powerful if assumptions are true and may take more time & care to calculate.

* In python:
  + t-test: scipy.stats.ttest\_ind
  + proportion z-test: statsmodels.stats.proportion.proportions\_ztest
  + ANOVA F-test: scipy.stats.f\_oneway
  + test for independence: scipy.stats.chi2\_contingency
  + Fisher’s exact test: scipy.stats.fisher\_exact
  + Randomization test: NA yet

## 2-sample t-test

* Q: If the response is quantitative, what is the classical approach to determining if the means are different in 2 independent groups?

A: A 2-sample t-test for means

* The 2-sample t-test for the mean difference between.2 treatment groups is

The p-value can then be calculated based on a distribution. The assumptions for this test include (i) independent observations and (ii) normally distributed responses within each group (or sufficiently large sample size).

## 2-sample z-test

* Q: If the responses of success are different in 2 independent groups?

A: A 2-sample z-test for proportions.

* The 2-sample z-test for the difference in proportions between 2 treatment groups is

where is the overall ‘pooled’ proportion of successes. The p-value can then be calculated based on a standard normal distribution.

## ANOVA F-test

* The ANOVA F-test for differences in means among 3+ groups can be calculated as follows:

the mean response is equal in all K groups.

there is a difference in mean response somewhere among the groups.

where is the sample size in group k, is the mean response in group k, is the variance of responses in group k, is the overall mean response, and n is the total sample size. The p-value can then be calculated based on a .

## Chi-squared test for independence

* The test for independence can be calculated as follows:

the categorical variables are independent.

the categorical variables are not independent.

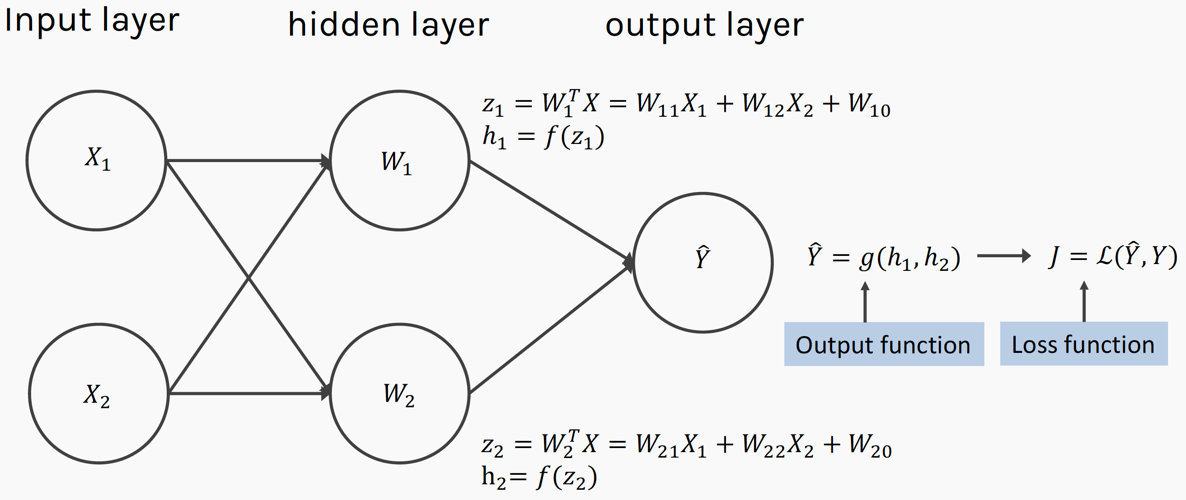
where .

The p-value can then be calculated based on a distribution.

# Neural Network

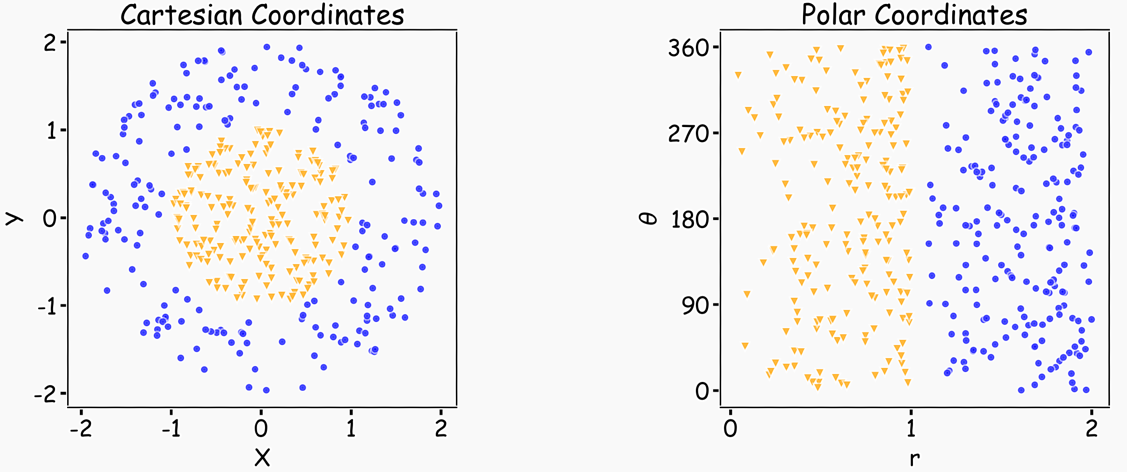
## Anatomy

* Anatomy of NN:



* Q: Why layers?

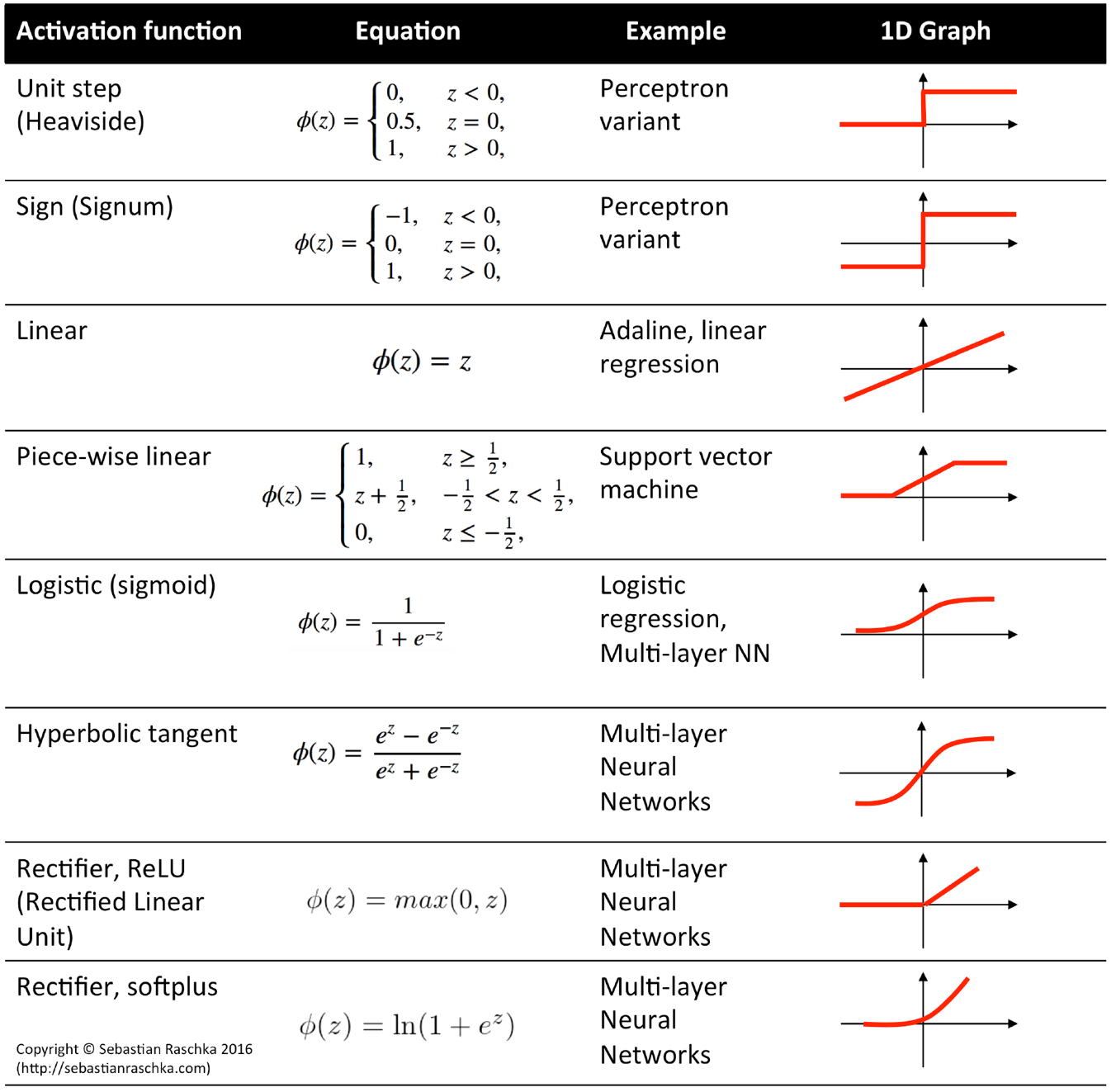
A: Representation matters! If you could find a way to represent the data that’s easier to separate, that would be cool.



## Design Choices

### Activation Functions

* The activation function should:
  + Ensure not linearity.
  + Ensure gradients remain large through hidden unit.
* Common choices are:
  + Sigmoid
  + ReLU, leaky ReLU, Generalized ReLU, MaxOut
  + softplus
  + tanh
  + swish

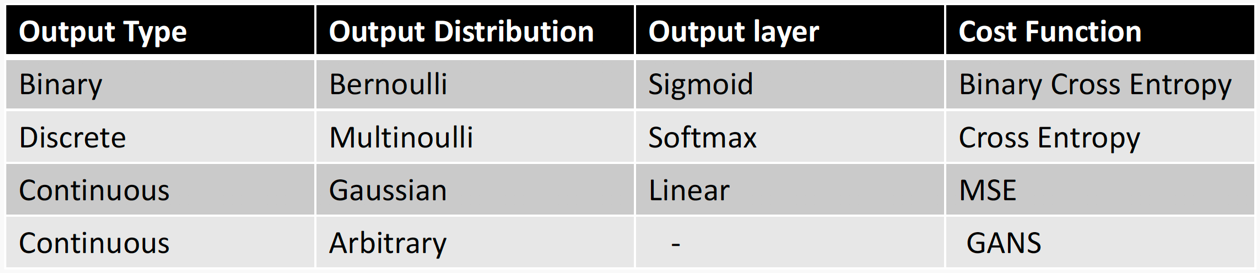


* ReLU is the most used activation function, because it beats everybody else, but there is no strong theoretical reason behind it. However, some potential explanations include (i) simple derivative: either 0 or 1, (ii) When derivative is 0, the algorithm wouldn’t be learning and the weights will be stuck there. It’s like some sort of regularization that will end up with a sparse matrix. It simplifies the NN.
* Q: Why we start from sigmoid, but not using it now?

A: Because sigmoid function is very similar to standard normal distribution, with huge area of 0 derivative at the left and right tails, which makes the model not learning.

### Output Units and Loss Function

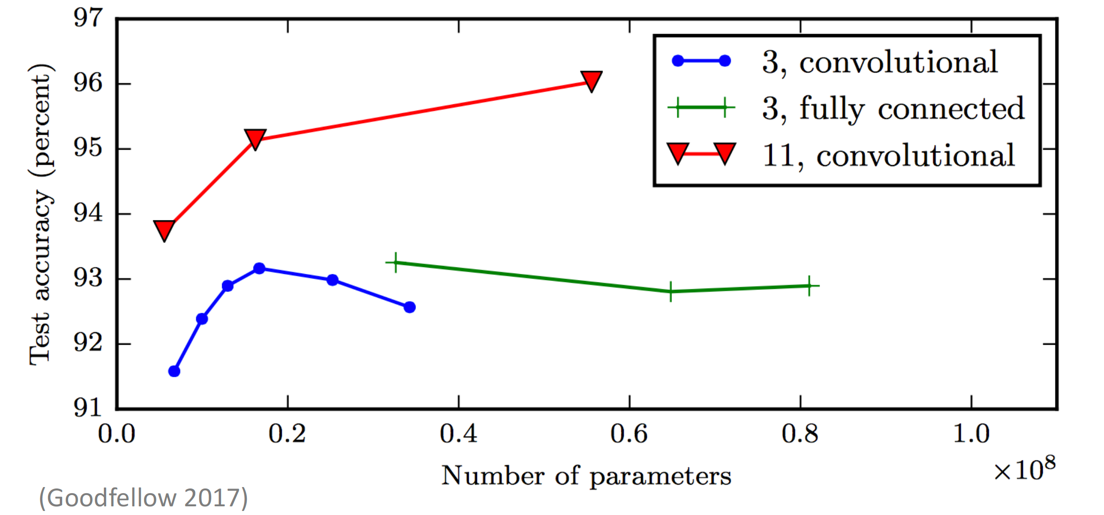
* Do not need to design separate loss functions.
* Gradient of cost function must be large enough, otherwise, the model wouldn’t be learning



### Architecture

* Universal Approximation Theorem:
  + Think of a Neural Network as function approximation.
  + One hidden layer is enough to represent an approximation of any function to an arbitrary degree of accuracy.
  + Q: So why deeper?

A: (i) Shallow net may need (exponentially) more width, (ii) Shallow net may overfit more. Thus, given the same number parameters, go with the model with more layers and less nodes.



## Regularization

### Norm penalties

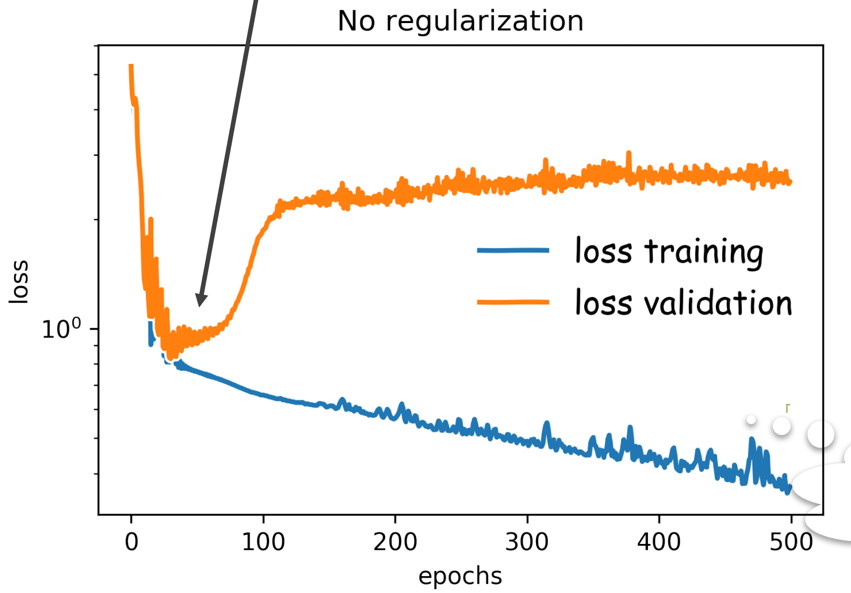
* We used to optimize

Change to:

* regularization:
  + Weights decay in proportion to its size
  + MAP estimation with Gaussian prior
  + Bias is not penalized.
* regularization:
  + Encourages sparsity
  + MAP estimation with Laplacian prior
  + Bias is not penalized.

### Early stopping

* Terminate while validation set performance is better.
  + Training time can be treated as a hyperparameter.
  + 1 epoch is just one iteration (forward and backward), and it updates the weights only once, but not enough for sure.



### Data Augmentation

* Besides bootstrapping, we can more training samples. Take images as example, we can rotate, zoom in, zoom out, add noise to the current samples to generate new samples.

### Sparse Representation

* Norm penalizes in the layers.

### Bagging

* Bootstrap Aggregation

### Dropout

* Randomly drop out units in the neural network.

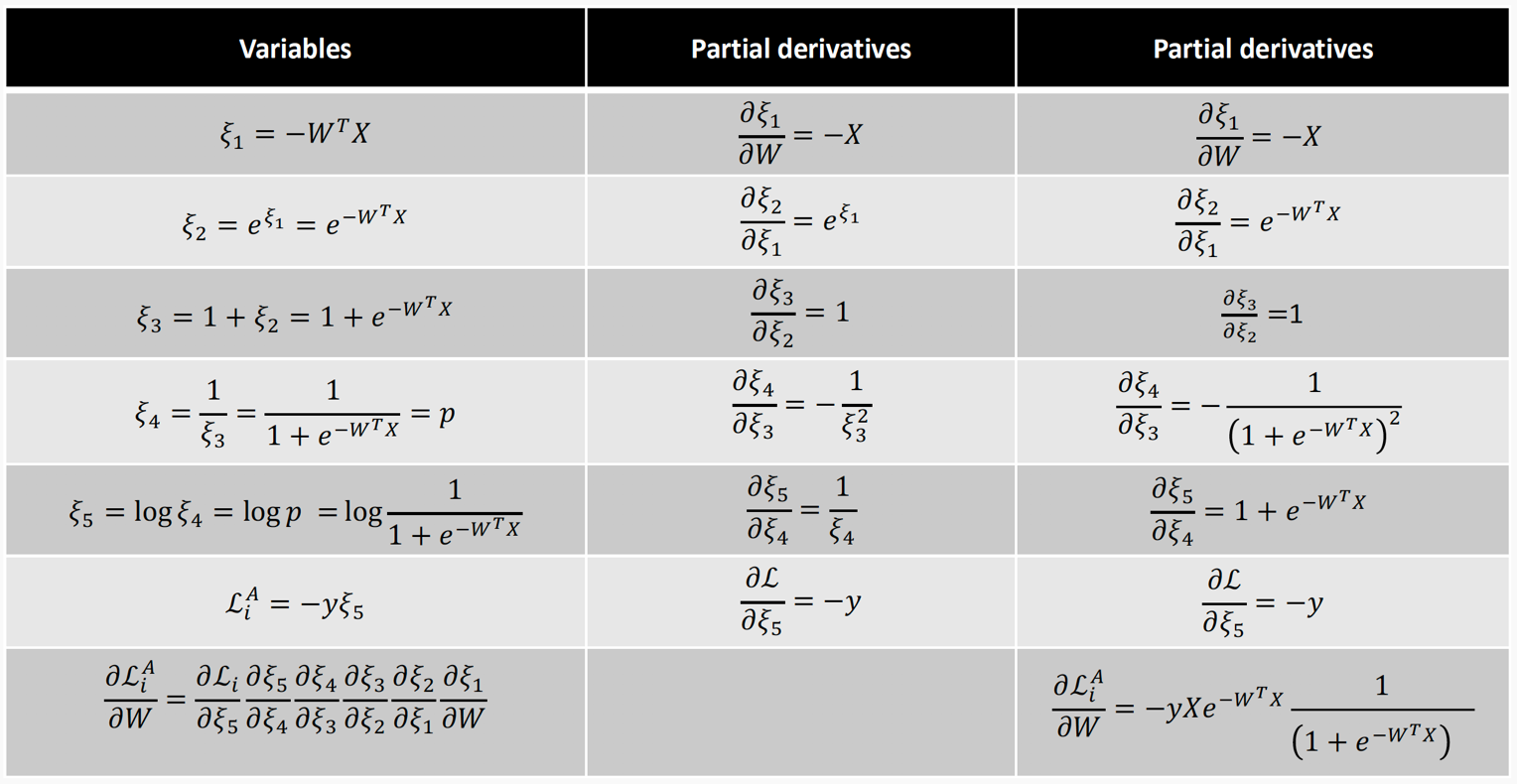
## Backpropagation

* For logistic regression, the objective function is to minimize:

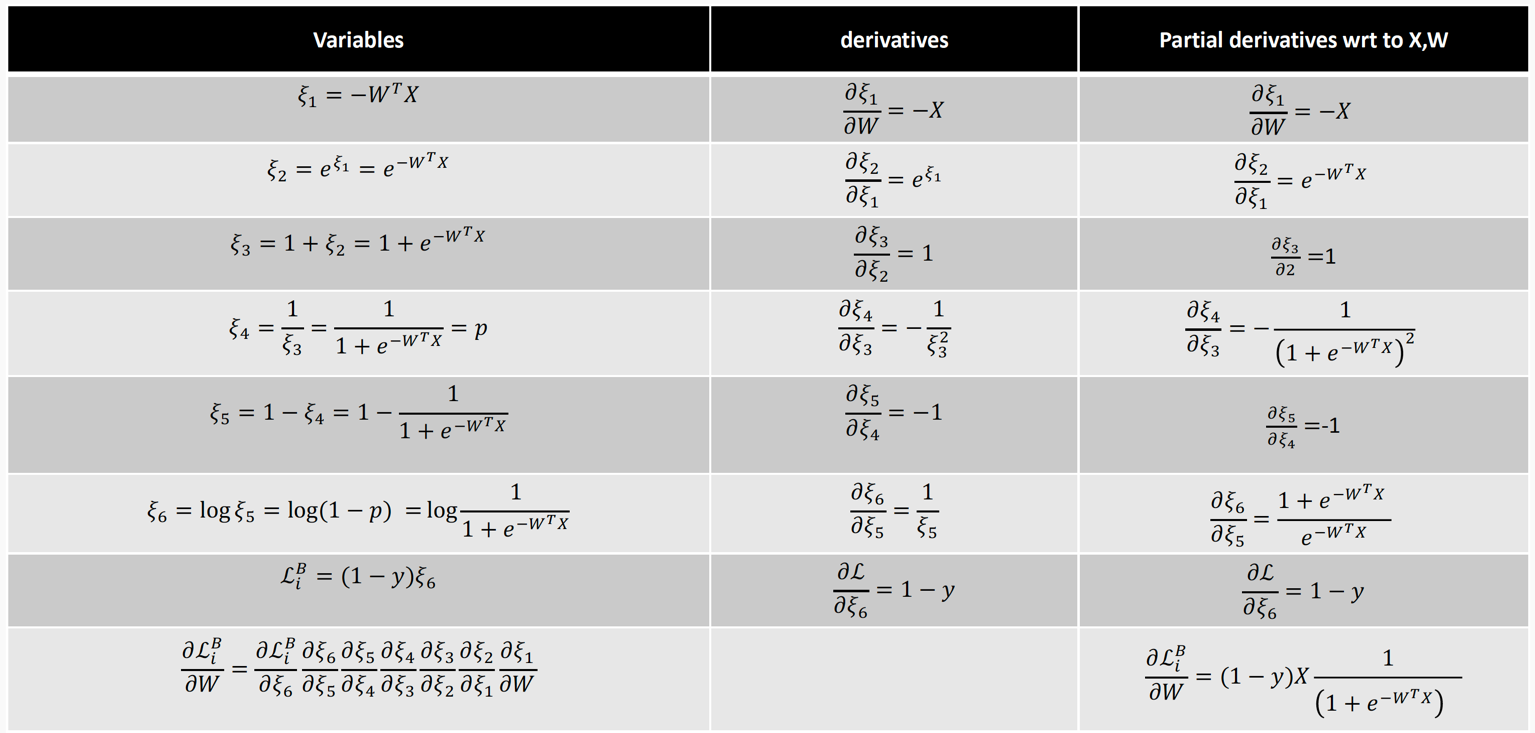
with

so the derivative with respect to W is:

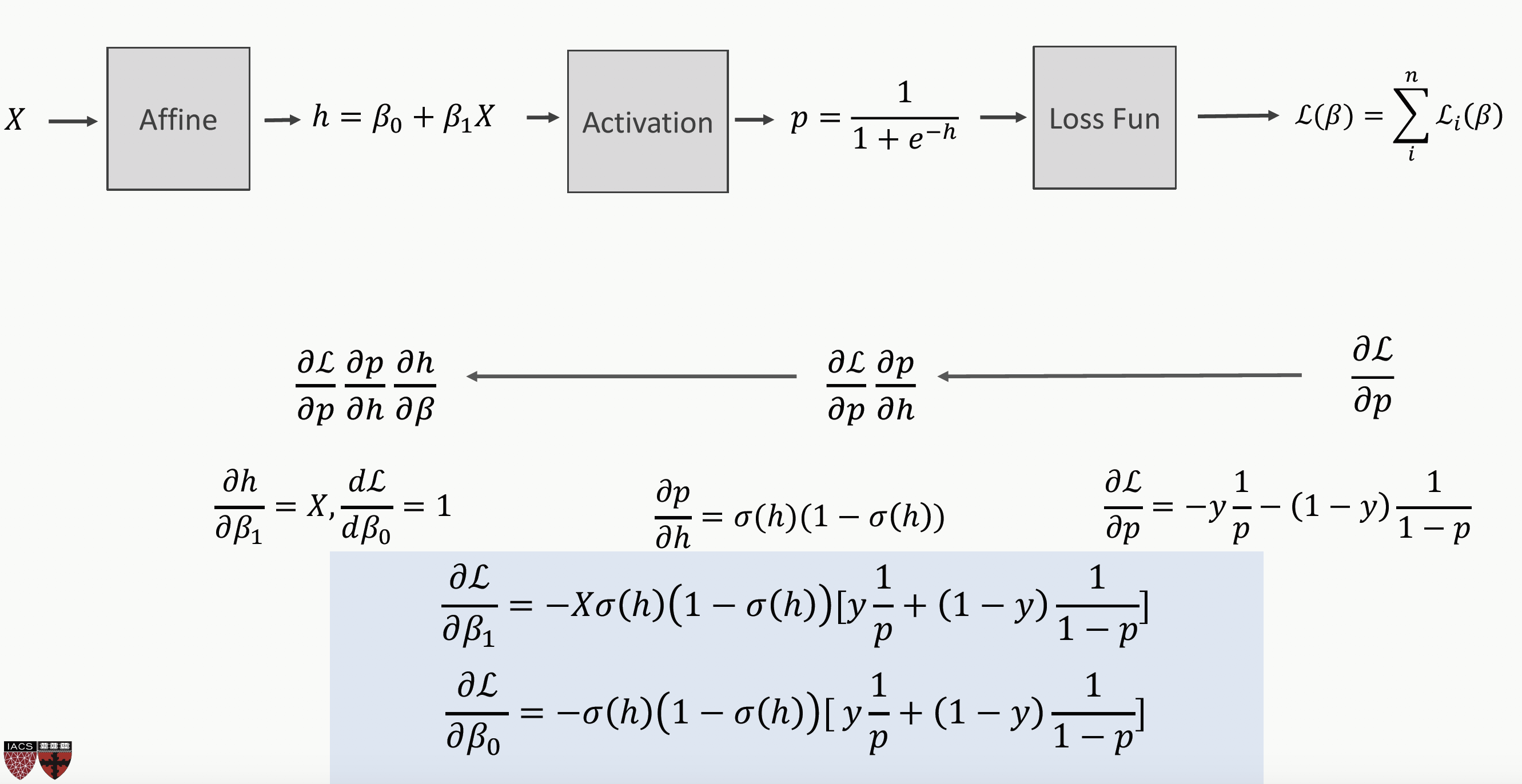
* :



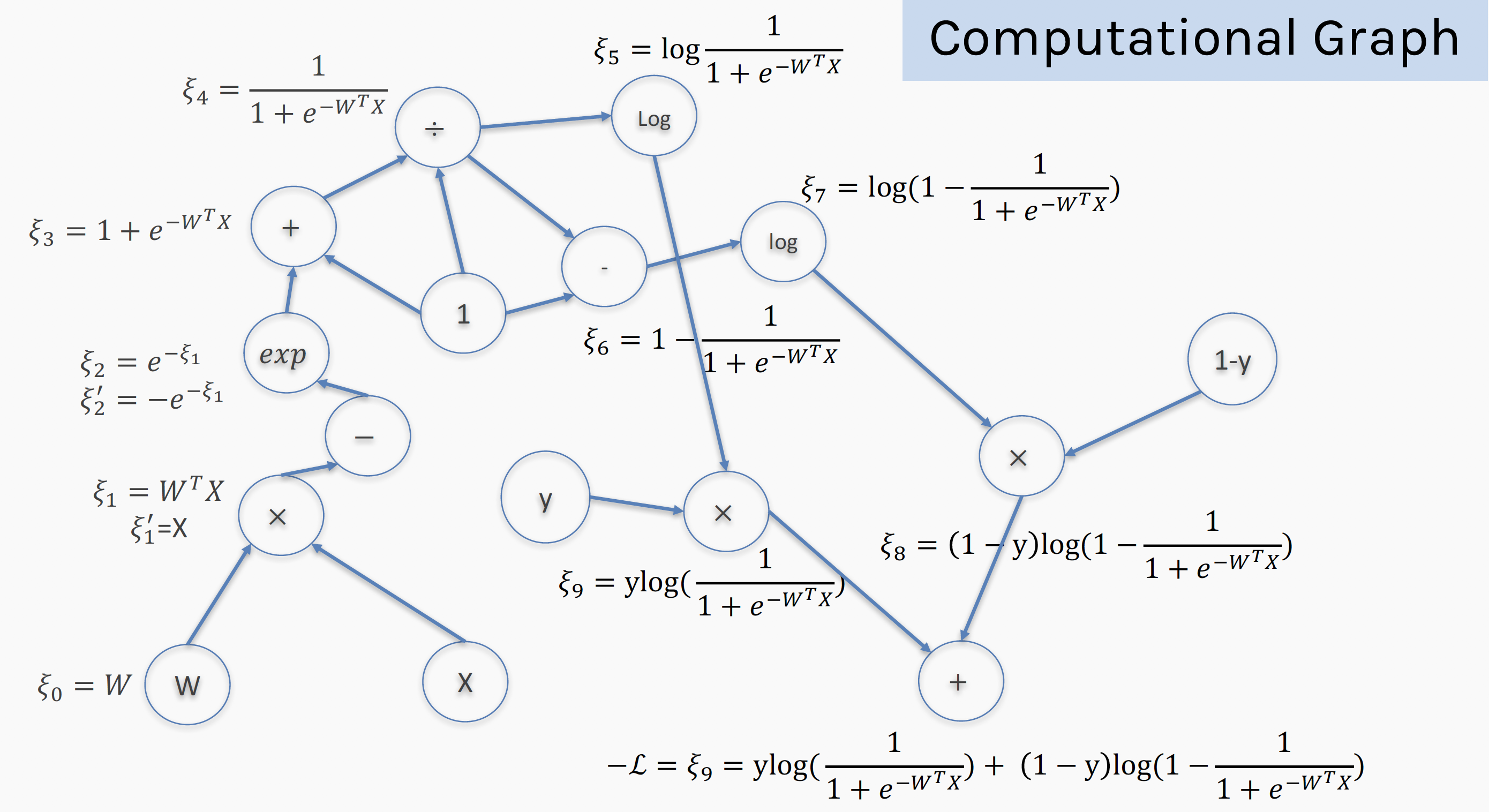
* :



* For a simple neuron:



* Derivatives need to be evaluated at some values of X, y and W, but since we have an expression, we can build a function that takes an input X, y, W and returns the derivatives and then we can use gradient descent to update. This approach works well, but it does not generalize. For example, if the network is changed, we need to write a new function to evaluate the derivatives.
* Need to find a formalism to calculate the derivatives of the loss function w.r.t. the weights that is:
  + Flexible enough that adding a node or a layer or changing something in the network won’t require to re-derive the functional form from scratch.
  + It is exact.
  + It is computationally efficient.
* Autograd: Automatic Differentiation
  + First, we specify the network structure.
  + Second, we create the computational graph like below. In Keras, when you compile a NN, Keras will generate the computational graph for you.



* + Third, at each node of the graph, we build two functions: the evaluation of the variable and its partial derivative with respect to the previous variable.
  + Now we can either go forward or backward depending on the situation. In general, forward is easier to implement and to understand. The difference is clearer when there are multiple nodes per layer.