Cheat Sheet – Regression Analysis

What is Regression Analysis?
Fitting a function f(.) to datapoints y_i=f(x_i) under some error function. Based on the estimated function and error, we have the following types of regression

1. Linear Regression:
Fits a line minimizing the sum of mean-squared error for each datapoint.

$$min_\beta \sum_i \|y_i - f_\beta^{linear}(x_i)\|^2$$
$$f_\beta^{linear}(x_i) = \beta_0 + \beta_1 x_i$$

2. Polynomial Regression:
Fits a polynomial of order k (k+1 unknowns) minimizing the sum of mean-squared error for each datapoint.

$$min_\beta \sum_{i=0}^{m} \|y_i - f_\beta^{poly}(x_i)\|^2$$
$$f_\beta^{poly}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_k x_i^k$$

3. Bayesian Regression:
For each datapoint, fits a gaussian distribution by minimizing the mean-squared error. As the number of data points x_i increases, it converges to point estimates i.e. $n \to \infty, \sigma^2 \to 0$

$$min_\beta \sum_i \|y_i - \mathcal{N}\left(f_\beta(x_i), \sigma^2\right)\|^2$$
$$f_\beta(x_i) = f_\beta^{poly}(x_i) \text{ or } f_\beta^{linear}(x_i)$$
$$\mathcal{N}(\mu, \sigma^2) \to \text{Gaussian with mean } \mu \text{ and variance } \sigma^2$$

4. Ridge Regression:
Can fit either a line, or polynomial minimizing the sum of mean-squared error for each datapoint and the weighted L2 norm of the function parameters beta.

$$min_\beta \sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{k} \beta_j^2$$
$$f_\beta(x_i) = f_\beta^{poly}(x_i) \text{ or } f_\beta^{linear}(x_i)$$

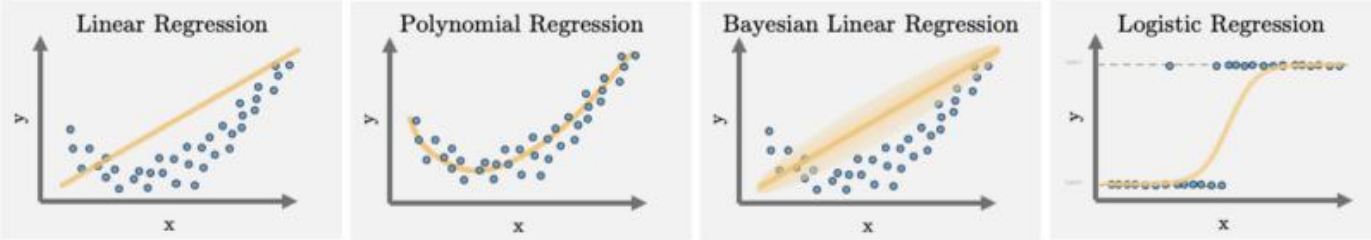
5. LASSO Regression:
Can fit either a line, or polynomial minimizing the the sum of mean-squared error for each datapoint and the weighted L1 norm of the function parameters beta.

$$min_\beta \sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{k} |\beta_j|$$
$$f_\beta(x_i) = f_\beta^{poly}(x_i) \text{ or } f_\beta^{linear}(x_i)$$

6. Logistic Regression (NOT regression, but classification):
Can fit either a line, or polynomial with sigmoid activation minimizing the sum of mean-squared error for each datapoint. The labels y are binary class labels.

$$min_\beta \sum_i \|y_i - \sigma(f_\beta(x_i))\|^2$$
$$f_\beta(x_i) = f_\beta^{poly}(x_i) \text{ or } f_\beta^{linear}(x_i)$$
$$\sigma(t) = \frac{1}{1+e^{-t}}$$

Visual Representation:



Summary:

	What does it fit?	Estimated function	Error Function
Linear	A line in n dimensions	$f_\beta^{linear}(x_i) = \beta_0 + \beta_1 x_i$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2$
Polynomial	A polynomial of order k	$f_\beta^{poly}(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2$
Bayesian Linear	Gaussian distribution for each point	$\mathcal{N}\left(f_\beta(x_i), \sigma^2\right)$	$\sum_i \|y_i - \mathcal{N}\left(f_\beta(x_i), \sigma^2\right)\|^2$
Ridge	Linear/polynomial	$f_\beta^{poly}(x_i)$ or $f_\beta^{linear}(x_i)$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{n} \beta_j^2$
LASSO	Linear/polynomial	$f_\beta^{poly}(x_i)$ or $f_\beta^{linear}(x_i)$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2 + \sum_{j=0}^{n} \|\beta_j\|$
Logistic	Linear/polynomial with sigmoid	$\sigma(f_\beta(x_i))$	$\sum_{i=0}^{m} \|y_i - f_\beta(x_i)\|^2$

Source: https://www.cheatsheets.aqeel-anwar.com

Cheat Sheet – Bias-Variance Tradeoff

What is Bias? $bias = \mathbb{E}[f'(x)] - f(x)$

- Error between average model prediction and ground truth
- The bias of the estimated function tells us the capacity of the underlying model to predict the values

What is Variance? $variance = \mathbb{E}\left[\left(f'(x) - \mathbb{E}[f'(x)]\right)^2\right]$

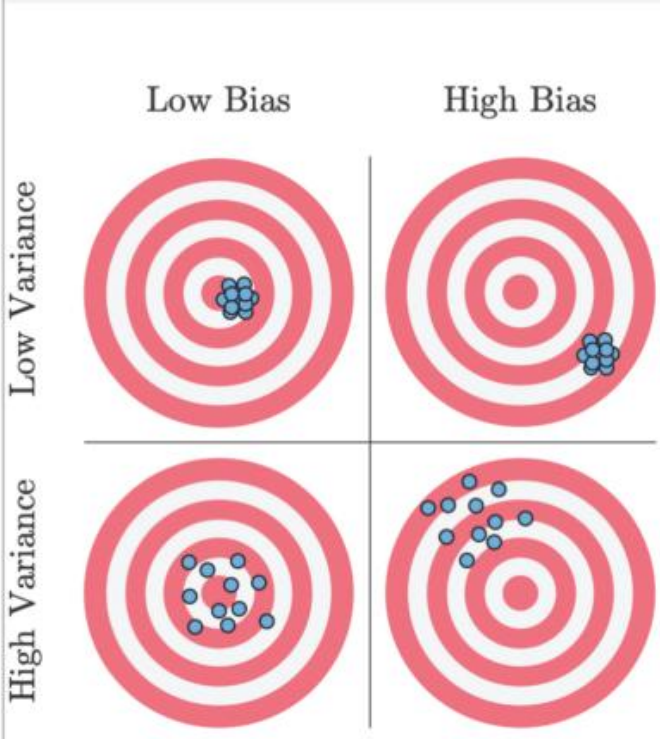
- Average variability in the model prediction for the given dataset
- The variance of the estimated function tells you how much the function can adjust to the change in the dataset

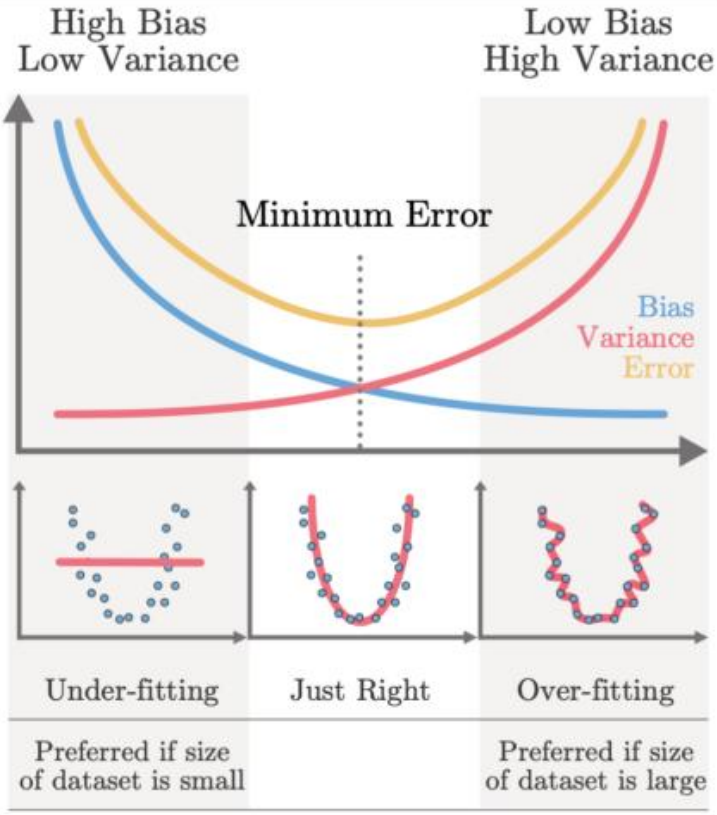
High Bias

→ Overly-simplified Model
→ Under-fitting
→ High error on both test and train data

High Variance

→ Overly-complex Model
→ Over-fitting
→ Low error on train data and high on test
→ Starts modelling the noise in the input





Bias variance Trade-off

- Increasing bias reduces variance and vice-versa
- Error = bias^2 + variance +irreducible error
- The best model is where the error is reduced.
- Compromise between bias and variance

Cheat Sheet – Regularization in ML

What is Regularization in ML?

- Regularization is an approach to address over-fitting in ML.
- Overfitted model fails to generalize estimations on test data
- When the underlying model to be learned is low bias/high variance, or when we have small amount of data, the estimated model is prone to over-fitting.
- Regularization reduces the variance of the model

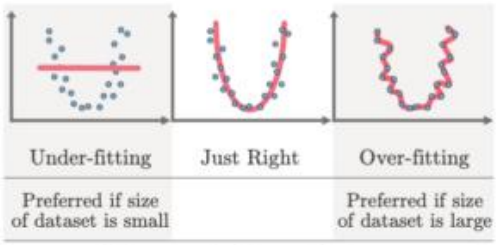


Figure 1. Overfitting

Types of Regularization:

1. Modify the loss function:

- **L2 Regularization:** Prevents the weights from getting too large (defined by L2 norm). Larger the weights, more complex the model is, more chances of overfitting.

$$loss = error(y, \hat{y}) + \lambda \sum_j \beta_j^2 \quad \lambda \geq 0,\ \lambda \propto model\ bias,\ \ \lambda \propto \frac{1}{model\ variance}$$

- **L1 Regularization:** Prevents the weights from getting too large (defined by L1 norm). Larger the weights, more complex the model is, more chances of overfitting. L1 regularization introduces sparsity in the weights. It forces more weights to be zero, than reducing the the average magnitude of all weights

$$loss = error(y, \hat{y}) + \lambda \sum_j |\beta_j| \quad \lambda \geq 0,\ \lambda \propto model\ bias,\ \ \lambda \propto \frac{1}{model\ variance}$$

- **Entropy:** Used for the models that output probability. Forces the probability distribution towards uniform distribution.

$$loss = error(p, \hat{p}) - \lambda \sum_i \hat{p}_i log(\hat{p}_i) \quad \lambda \geq 0,\ \lambda \propto model\ bias,\ \ \lambda \propto \frac{1}{model\ variance}$$

2. Modify data sampling:

- **Data augmentation:** Create more data from available data by randomly cropping, dilating, rotating, adding small amount of noise etc.
- **K-fold Cross-validation:** Divide the data into k groups. Train on (k-1) groups and test on 1 group. Try all k possible combinations.

3. Change training approach:

- **Injecting noise:** Add random noise to the weights when they are being learned. It pushes the model to be relatively insensitive to small variations in the weights, hence regularization
- **Dropout:** Generally used for neural networks. Connections between consecutive layers are randomly dropped based on a dropout-ratio and the remaining network is trained in the current iteration. In the next iteration, another set of random connections are dropped.

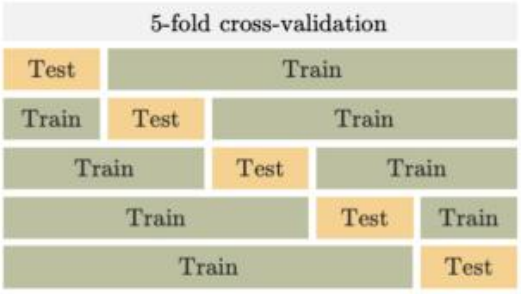


Figure 2. K-fold CV

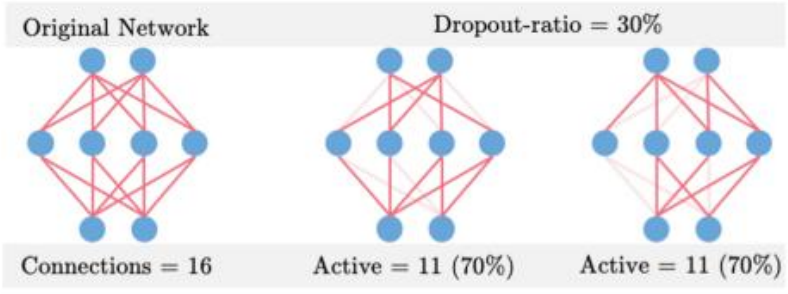


Figure 3. Drop-out

Source: https://www.cheatsheets.aqeel-anwar.com

Cheat Sheet – PCA Dimensionality Reduction

What is PCA?

- Based on the dataset find a new set of orthogonal feature vectors in such a way that the data spread is maximum in the direction of the feature vector (or dimension)
- Rates the feature vector in the decreasing order of data spread (or variance)
- The datapoints have maximum variance in the first feature vector, and minimum variance in the last feature vector
- The variance of the datapoints in the direction of feature vector can be termed as a measure of information in that direction.

Steps

1. Standardize the datapoints $X_{new} = \frac{X - mean(X)}{std(X)}$
2. Find the covariance matrix from the given datapoints $C[i,j] = cov(x_i, x_j)$
3. Carry out eigen-value decomposition of the covariance matrix $C = V\Sigma V^{-1}$
4. Sort the eigenvalues and eigenvectors $\Sigma_{sort} = sort(\Sigma)$ $V_{sort} = sort(V, \Sigma_{sort})$

Dimensionality Reduction with PCA

- Keep the first m out of n feature vectors rated by PCA. These m vectors will be the best m vectors preserving the maximum information that could have been preserved with m vectors on the given dataset

Steps:

1. Carry out steps 1-4 from above
2. Keep first m feature vectors from the sorted eigenvector matrix $V_{reduced} = V[:, 0:m]$
3. Transform the data for the new basis (feature vectors) $X_{reduced} = X_{new} \times V_{reduced}$
4. The importance of the feature vector is proportional to the magnitude of the eigen value

Figure 1

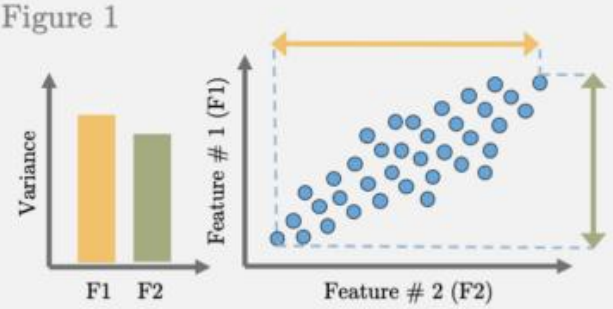


Figure 2

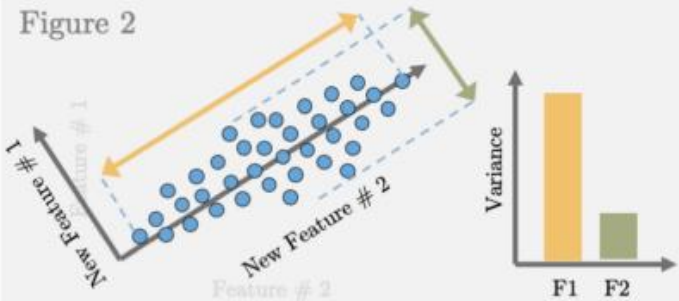


Figure 3

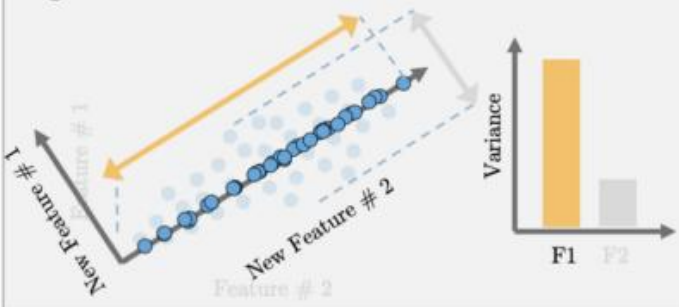


Figure 1: Datapoints with feature vectors as x and y-axis
Figure 2: The cartesian coordinate system is rotated to maximize the standard deviation along any one axis (new feature # 2)
Figure 3: Remove the feature vector with minimum standard deviation of datapoints (new feature # 1) and project the data on new feature # 2

Source: https://www.cheatsheets.aqeel-anwar.com

Cheat Sheet – Bayes Theorem and Classifier

What is Bayes' Theorem?

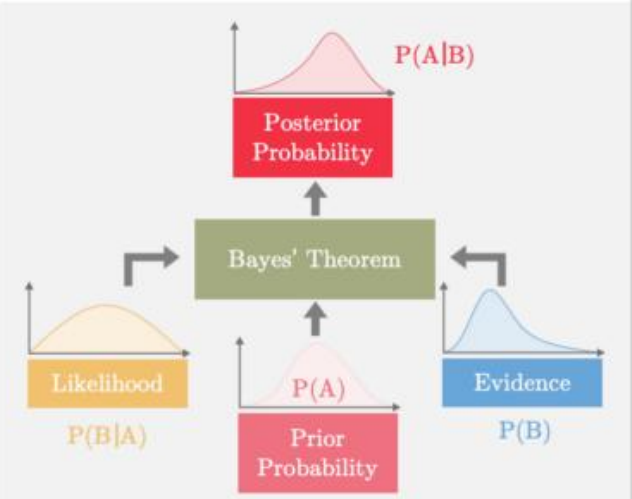
- Describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

$$P(A|B) = \frac{P(B|A)(likelihood) \times P(A)(prior)}{P(B)(prior)}$$

- How the probability of an event changes when we have knowledge of another event

P(A) ⟶ P(A|B)

↳ Usually a better estimate than P(A)



Example

- Probability of fire P(F) = 1%
- Probability of smoke P(S) = 10%
- Prob of smoke given there is a fire P(S|F) = 90%
- What is the probability that there is a fire given we see a smoke P(F|S)?

$$P(F|S) = \frac{P(S|F) \times P(F)}{P(S)} = \frac{0.9 \times 0.01}{0.1} = 9\%$$

Maximum Apriori Probability (MAP) Estimation

The MAP estimate of the random variable y, given that we have observed iid (x_1, x_2, x_3, …), is given by. We try to accommodate our prior knowledge when estimating.

$$\hat{y}_{MAP} = argmax_y \; P(y) \prod_i P(x_i|y)$$

y that maximizes the product of prior and likelihood

Maximum Likelihood Estimation (MLE)

The MAP estimate of the random variable y, given that we have observed iid (x_1, x_2, x_3, …), is given by. We assume we don't have any prior knowledge of the quantity being estimated.

$$\hat{y}_{MLE} = argmax_y \prod_i P(x_i|y)$$

y that maximizes only the likelihood

MLE is a special case of MAP where our prior is uniform (all values are equally likely)

Naïve Bayes' Classifier (Instantiation of MAP as classifier)

Suppose we have two classes, $y=y_1$ and $y=y_2$. Say we have more than one evidence/features (x_1, x_2, x_3, …), using Bayes' theorem

$$P(y|x_1, x_2, x_3, \ldots) = \frac{P(x_1, x_2, x_3, \ldots|y) \times P(y)}{P(x_1, x_2, x_3, \ldots)}$$

Bayes' theorem assumes the features (x_1, x_2, x_3, …) are i.i.d. i.e $P(x_1, x_2, x_3, \ldots|y) = \prod_i P(x_i|y)$

$$P(y|x_1, x_2, x_3, \ldots) = \prod_i P(x_i|y) \frac{P(y)}{P(x_1, x_2, x_3, \ldots)}$$

$$\hat{y} = y_1 \; if \; \frac{P(y_1|x_1, x_2, x_3, \ldots)}{P(y_2|x_1, x_2, x_3, \ldots)} > 1 \quad else \; \hat{y} = y_2$$

Cheat Sheet – Imbalanced Data in Classification

Classifier that always predicts label blue yields prediction accuracy of 90%

Accuracy doesn't always give the correct insight about your trained model

Accuracy: %age correct prediction	Correct prediction over total predictions	One value for entire network
Precision: Exactness of model	From the detected cats, how many were actually cats	Each class/label has a value
Recall: Completeness of model	Correctly detected cats over total cats	Each class/label has a value
F1 Score: Combines Precision/Recall	Harmonic mean of Precision and Recall	Each class/label has a value

Performance metrics associated with Class 1

Predicted Labels \ Actual Labels	1	0
1	True Positive	False Positive
0	False Negative	True Negative

(Is your prediction correct?) (What did you predict)

True Negative

(Your prediction is correct) (You predicted 0)

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F1 score} = 2x \frac{(\text{Prec x Rec})}{(\text{Prec} + \text{Rec})}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{False +ve rate} = \frac{FP}{TN + FP}$$

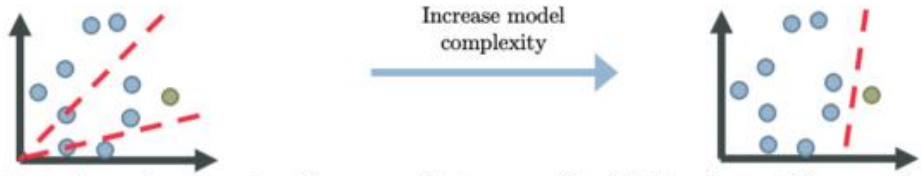
$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{Recall, Sensitivity} = \text{True +ve rate} = \frac{TP}{TP + FN}$$

Possible solutions

1. **Data Replication:** Replicate the available data until the number of samples are comparable
2. **Synthetic Data:** Images: Rotate, dilate, crop, add noise to existing input images and create new data
3. **Modified Loss:** Modify the loss to reflect greater error when misclassifying smaller sample set
4. **Change the algorithm:** Increase the model/algorithm complexity so that the two classes are perfectly separable (Con: Overfitting)

$$loss = a * loss_{green} + b * loss_{blue} \qquad a > b$$



No straight line (y=ax) passing through origin can perfectly separate data. **Best solution:** line y=0, predict all labels blue

Straight line (y=ax+b) can perfectly separate data. Green class will no longer be predicted as blue

Source: https://www.cheatsheets.aqeel-anwar.com

Machine Learning Algorithms - Regression

Key Attributes Table

Popular Algorithms	Type	Key Concepts	Feature Range Standardization [0, 1]	Results Interpretable?	Key Parameters
Linear Regression	Linear	Simplest method - Uses OLS to reduce error and find the	Not Required	Yes - Model terms indicate magnitude / direction of each features contribution	N/A
GLM (Generalized Linear Model) LASSO, Ridge Regression, Elastic Net	Linear	Linear method that penalizes irrelevant features using a concept called "Regularization", where the weight of irrelevant features is reduced to make their effect on the model lower. L1 Regularization - Called LASSO regression L2 Regularization - Called Ridge Regression Elastic Net: Combines L1 & L2 Regularization	Required (but see Application Note below). **Application Note**: In practice, some algorithms (i.e. R's glmnet::glmnet()) implement standardization internally and re-scale prior to returning term estimates and predictions. This means that features need not be scaled prior to use.	Yes, if standardization is performed internally to algorithm. Model terms indicate magnitude / direction of each features contribution	Penalty (alpha) - How much to penalize the parameters Mixture (L1 Ratio) - Ratio between L1 and L2 Regularization
Decision Tree	Tree-Based (Non-Linear)	A decision tree is a set of decision rules. Each rule is considered a node with a split being a binary decision. The decisions terminate at a leaf.	Not Required	Yes - Decision Tree Plots show rule-based decisions that show how to arrive at model prediction	Max Tree Depth - How many splits for the longest tree Min Samples Per Leaf / Node - How many samples in each end node (leaf) Cost Complexity (Cp) / Min Impurity - Instructs when to stop (create a leaf) if additional information gain is not above a Cp threshold
Random Forest	Tree-Based (Non-Linear)	Ensemble learning method where many trees are created on sub-samples of data set and combined using averaging. This process controls overfitting, typicallying leading to a more accurate model. However, because the models are combined, the decision rules become incomprehensible. This process is often called "Bagging".	Not Required	No (see Application Note) **Application Note:** Feature importance can be obtained with additional methods for global (Variable Importance) and local (e.g. LIME) model understanding.	See Decision Tree Key Parameters, and: Replacement - whether or not to draw samples with replacement Number of Features - How many columns to use when sampling Number of Trees - How many trees to average
GBM (Gradient Boosted Machine) XGBoost	Tree-Based (Non-Linear)	Implements a technique called "Boosting" to build decision trees of weak prediction models and generalizes using a loss function. The weak learners converge to a strong learner.	Not Required	No (see Application Note) **Application Note:** Feature importance can be obtained with additional methods for global (Variable Importance) and local (e.g. LIME) model understanding.	See RandomForest Key Parameters, and: Learning Rate (eta) - The rate that the boosting algorithm adapts Loss Reduction (gamma) - The loss function to use during splitting Sample Size - The proportion of data exposed to the model during each iteration
SVM (Support Vector Machine)	Kernel Basis (Polynomial or Radial) (Non-Linear)	An algorithm that uses a kernel to transform the feature space to linearly seperable boundaries, and then applies a margin penalizing points that are incorrectly measured outside of the margin. The kernel transformation (i.e. radial, polynomial) makes it possible to perform linear separations within non-linear data.	Required (but see Application Note below). **Application Note**: In practice, some algorithms (i.e. R's kernlabs::ksvm()) implement standardization internally and re-scale prior to returning term estimates and predictions. This means that features need not be scaled prior to use.	No (see Application Note) **Application Note:** Feature importance can be obtained with additional methods for global (Variable Importance) and local (e.g. LIME) model understanding.	Kernel - Polynomial or Radial Basis Function Cost / Regularization - Cost of predicting sample on wrong side of the SVM margin Margin (Epsilon) - Specifies region where no penalty is applied Degree (Polynomial) - Degree of Polynomial. Use 1 for linear, 2 or more for flexible (quadratic) Scale Factor (Polynomial) - Factor to adjust bias/variance Gamma or Sigma (Radial) - Factor to adjust bias/variance
Deep Learning (Neural Network)	Neural Network (Non-Linear)	Learning algorithms with input and output and layers in between where the model parameters are learned. The user develops the architecture of the neural network, and the algorithm learns the model through iteratively seeking to minimize a cost function.	Required	No (see Application Note) **Application Note:** Feature importance can be obtained with additional methods for global (Variable Importance) and local (e.g. LIME) model understanding.	Many tuning parameters & architecture decisions

Data Science Cheatsheet

Compiled by Maverick Lin (http://mavericklin.com)

Last Updated August 13, 2018

What is Data Science?

Multi-disciplinary field that brings together concepts from computer science, statistics/machine learning, and data analysis to understand and extract insights from the ever-increasing amounts of data.

Two paradigms of data research.

1. **Hypothesis-Driven:** Given a problem, what kind of data do we need to help solve it?
2. **Data-Driven:** Given some data, what interesting problems can be solved with it?

The heart of data science is to always ask questions. Always be curious about the world.

1. What can we learn from this data?
2. What actions can we take once we find whatever it is we are looking for?

Types of Data

Structured: Data that has predefined structures. e.g. tables, spreadsheets, or relational databases.
Unstructured Data: Data with no predefined structure, comes in any size or form, cannot be easily stored in tables. e.g. blobs of text, images, audio
Quantitative Data: Numerical. e.g. height, weight
Categorical Data: Data that can be labeled or divided into groups. e.g. race, sex, hair color.
Big Data: Massive datasets, or data that contains greater *variety* arriving in increasing *volumes* and with ever-higher *velocity* (3 Vs). Cannot fit in the memory of a single machine.

Data Sources/Fomats

Most Common Data Formats CSV, XML, SQL, JSON, Protocol Buffers
Data Sources Companies/Proprietary Data, APIs, Government, Academic, Web Scraping/Crawling

Main Types of Problems

Two problems arise repeatedly in data science.
Classification: Assigning something to a discrete set of possibilities. e.g. spam or non-spam, Democrat or Republican, blood type (A, B, AB, O)
Regression: Predicting a numerical value. e.g. someone's income, next year GDP, stock price

Probability Overview

Probability theory provides a framework for reasoning about likelihood of events.

Terminology

Experiment: procedure that yields one of a possible set of outcomes e.g. repeatedly tossing a die or coin
Sample Space S: set of possible outcomes of an experiment e.g. if tossing a die, S = (1,2,3,4,5,6
Event E: set of outcomes of an experiment e.g. event that a roll is 5, or the event that sum of 2 rolls is 7
Probability of an Outcome s or P(s): number that satisfies 2 properties

1. for each outcome s, $0 \leq \text{P(s)} \leq 1$
2. $\sum \text{p(s)} = 1$

Probability of Event E: sum of the probabilities of the outcomes of the experiment: $\text{p(E)} = \sum_{s \subset E} \text{p(s)}$
Random Variable V: numerical function on the outcomes of a probability space
Expected Value of Random Variable V: $\text{E(V)} = \sum_{s \subset S} \text{p(s)} * \text{V(s)}$

Independence, Conditional, Compound

Independent Events: A and B are independent iff:

$$\text{P(A} \cap \text{B)} = \text{P(A)P(B)}$$
$$\text{P}(A|B) = \text{P(A)}$$
$$\text{P}(B|A) = \text{P(B)}$$

Conditional Probability: $\text{P}(A|B) = \text{P(A,B)/P(B)}$
Bayes Theorem: $\text{P}(A|B) = \text{P}(B|A)\text{P(A)/P(B)}$
Joint Probability: $\text{P(A,B)} = \text{P}(B|A)\text{P(A)}$
Marginal Probability: P(A)

Probability Distributions

Probability Density Function (PDF) Gives the probability that a rv takes on the value x: $p_X(x) = P(X = x)$
Cumulative Density Function (CDF) Gives the probability that a random variable is less than or equal to x: $F_X(x) = P(X \leq x)$
Note: The PDF and the CDF of a given random variable contain exactly the same information.

Descriptive Statistics

Provides a way of capturing a given data set or sample. There are two main types: **centrality** and **variability** measures.

Centrality

Arithmetic Mean Useful to characterize symmetric distributions without outliers $\mu_X = \frac{1}{n}\sum x$
Geometric Mean Useful for averaging ratios. Always less than arithmetic mean $= \sqrt[n]{a_1 a_2 ... a_3}$
Median Exact middle value among a dataset. Useful for skewed distribution or data with outliers.
Mode Most frequent element in a dataset.

Variability

Standard Deviation Measures the squares differences between the individual elements and the mean

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}}$$

Variance $\text{V} = \sigma^2$

Interpreting Variance

Variance is an inherent part of the universe. It is impossible to obtain the same results after repeated observations of the same event due to random noise/error. Variance can be explained away by attributing to sampling or measurement errors. Other times, the variance is due to the random fluctuations of the universe.

Correlation Analysis

Correlation coefficients r(X,Y) is a statistic that measures the degree that Y is a function of X and vice versa. Correlation values range from -1 to 1, where 1 means fully correlated, -1 means negatively-correlated, and 0 means no correlation.
Pearson Coefficient Measures the degree of the relationship between linearly related variables

$$\text{r} = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)}$$

Spearman Rank Coefficient Computed on ranks and depicts monotonic relationships

Note: Correlation does not imply causation!

Data Cleaning

Data Cleaning is the process of turning raw data into a clean and analyzable data set. "Garbage in, garbage out." Make sure garbage doesn't get put in.

Errors vs. Artifacts

1. **Errors:** information that is lost during acquisition and can never be recovered e.g. power outage, crashed servers
2. **Artifacts:** systematic problems that arise from the data cleaning process. these problems can be corrected but we must first discover them

Data Compatibility

Data compatibility problems arise when merging datasets. Make sure you are comparing "apples to apples" and not "apples to oranges". Main types of conversions/unifications:

- **units** (metric vs. imperial)
- **numbers** (decimals vs. integers),
- **names** (John Smith vs. Smith, John),
- **time/dates** (UNIX vs. UTC vs. GMT),
- **currency** (currency type, inflation-adjusted, dividends)

Data Imputation

Process of dealing with missing values. The proper methods depend on the type of data we are working with. General methods include:

- Drop all records containing missing data
- Heuristic-Based: make a reasonable guess based on knowledge of the underlying domain
- Mean Value: fill in missing data with the mean
- Random Value
- Nearest Neighbor: fill in missing data using similar data points
- Interpolation: use a method like linear regression to predict the value of the missing data

Outlier Detection

Outliers can interfere with analysis and often arise from mistakes during data collection. It makes sense to run a "sanity check".

Miscellaneous

Lowercasing, removing non-alphanumeric, repairing, unidecode, removing unknown characters

Note: When cleaning data, always maintain both the raw data and the cleaned version(s). The raw data should be kept intact and preserved for future use. Any type of data cleaning/analysis should be done on a copy of the raw data.

Feature Engineering

Feature engineering is the process of using domain knowledge to create features or input variables that help machine learning algorithms perform better. Done correctly, it can help increase the predictive power of your models. Feature engineering is more of an art than science. FE is one of the most important steps in creating a good model. As Andrew Ng puts it:

"Coming up with features is difficult, time-consuming, requires expert knowledge. 'Applied machine learning' is basically feature engineering."

Continuous Data

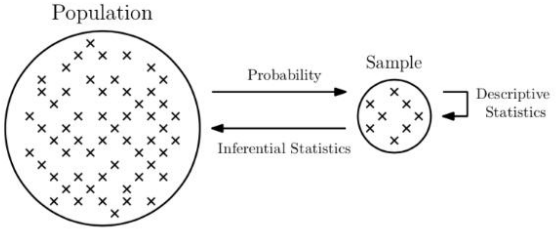
Raw Measures: data that hasn't been transformed yet
Rounding: sometimes precision is noise; round to nearest integer, decimal etc..
Scaling: log, z-score, minmax scale
Imputation: fill in missing values using mean, median, model output, etc..
Binning: transforming numeric features into categorical ones (or binned) e.g. values between 1-10 belong to A, between 10-20 belong to B, etc.
Interactions: interactions between features: e.g. subtraction, addition, multiplication, statistical test
Statistical: log/power transform (helps turn skewed distributions more normal), Box-Cox
Row Statistics: number of NaN's, 0's, negative values, max, min, etc
Dimensionality Reduction: using PCA, clustering, factor analysis etc

Discrete Data

Encoding: since some ML algorithms cannot work on categorical data, we need to turn categorical data into numerical data or vectors
Ordinal Values: convert each distinct feature into a random number (e.g. [r,g,b] becomes [1,2,3])
One-Hot Encoding: each of the m features becomes a vector of length m with containing only one 1 (e.g. [r, g, b] becomes [[1,0,0],[0,1,0],[0,0,1]])
Feature Hashing Scheme: turns arbitrary features into indices in a vector or matrix
Embeddings: if using words, convert words to vectors (word embeddings)

Statistical Analysis

Process of statistical reasoning: there is an underlying population of possible things we can potentially observe and only a small subset of them are actually sampled (ideally at random). Probability theory describes what properties our sample should have given the properties of the population, but ***statistical inference*** allows us to deduce what the full population is like after analyzing the sample.



Sampling From Distributions

Inverse Transform Sampling Sampling points from a given probability distribution is sometimes necessary to run simulations or whether your data fits a particular distribution. The general technique is called *inverse transform sampling* or Smirnov transform. First draw a random number p between [0,1]. Compute value x such that the CDF equals p: $F_X(x) = p$. Use x as the value to be the random value drawn from the distribution described by $F_X(x)$.

Monte Carlo Sampling In higher dimensions, correctly sampling from a given distribution becomes more tricky. Generally want to use Monte Carlo methods, which typically follow these rules: define a domain of possible inputs, generate random inputs from a probability distribution over the domain, perform a deterministic calculation, and analyze the results.

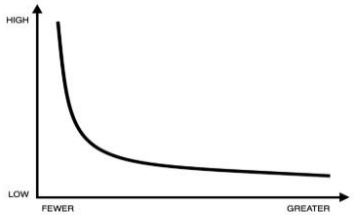
Classic Statistical Distributions

Binomial Distribution (Discrete)
Assume X is distributed Bin(n,p). X is the number of "successes" that we will achieve in n independent trials, where each trial is either a success or failure and each success occurs with the same probability p and each failure occurs with probability q=1-p.
PDF: $P(X = x) = \binom{n}{k} p^x (1-p)^{n-x}$
EV: $\mu = np$ Variance = npq

Normal/Gaussian Distribution (Continuous)
Assume X in distributed $\mathcal{N}(\mu, \sigma^2)$. It is a bell-shaped and symmetric distribution. Bulk of the values lie close to the mean and no value is too extreme. Generalization of the binomial distribution as n $\to \infty$.
PDF: $P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$
EV: μ Variance: σ^2
Implications: 68%-95%-99% rule. 68% of probability mass fall within 1σ of the mean, 95% within 2σ, and 99.7% within 3σ.

Poisson Distribution (Discrete)
Assume X is distributed Pois(λ). Poisson expresses the probability of a given number of events occurring in a fixed interval of time/space if these events occur independently and with a known constant rate λ.
PDF: $P(x) = \frac{e^{-\lambda}\lambda^x}{x!}$ EV: λ Variance = λ

Power Law Distributions (Discrete)
Many data distributions have much longer tails than the normal or Poisson distributions. In other words, the change in one quantity varies as a *power* of another quantity. It helps measure the inequality in the world. e.g. wealth, word frequency and Pareto Principle (80/20 Rule)
PDF: P(X=x) = $cx^{-\alpha}$, where α is the law's exponent and c is the normalizing constant



Modeling- Overview

Modeling is the process of incorporating information into a tool which can forecast and make predictions. Usually, we are dealing with statistical modeling where we want to analyze relationships between variables. Formally, we want to estimate a function $f(X)$ such that:

$$Y = f(X) + \epsilon$$

where X = $(X_1, X_2, ...X_p)$ represents the input variables, Y represents the output variable, and ϵ represents random error.

Statistical learning is set of approaches for estimating this $f(X)$.

Why Estimate f(X)?

Prediction: once we have a good estimate $\hat{f}(X)$, we can use it to make predictions on new data. We treat $\hat{f}$ as a black box, since we only care about the accuracy of the predictions, not why or how it works.
Inference: we want to understand the relationship between X and Y. We can no longer treat $\hat{f}$ as a black box since we want to understand how Y changes with respect to X = $(X_1, X_2, ...X_p)$

More About ϵ

The error term ϵ is composed of the reducible and irreducible error, which will prevent us from ever obtaining a perfect $\hat{f}$ estimate.

- **Reducible**: error that can potentially be reduced by using the most appropriate statistical learning technique to estimate f. The goal is to minimize the reducible error.
- **Irreducible**: error that cannot be reduced no matter how well we estimate f. Irreducible error is unknown and unmeasurable and will always be an upper bound for ϵ.

Note: There will always be trade-offs between model flexibility (prediction) and model interpretability (inference). This is just another case of the bias-variance trade-off. Typically, as flexibility increases, interpretability decreases. Much of statistical learning/modeling is finding a way to balance the two.

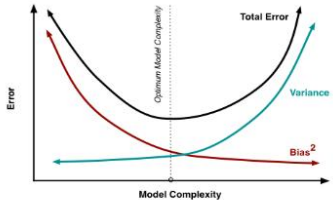
Modeling- Philosophies

Modeling is the process of incorporating information into a tool which can forecast and make predictions. Designing and validating models is important, as well as evaluating the performance of models. Note that the best forecasting model may not be the most accurate one.

Philosophies of Modeling

Occam's Razor Philosophical principle that the simplest explanation is the best explanation. In modeling, if we are given two models that predicts equally well, we should choose the simpler one. Choosing the more complex one can often result in overfitting.
Bias Variance Trade-Off Inherent part of predictive modeling, where models with lower bias will have higher variance and vice versa. Goal is to achieve low bias and low variance.

- *Bias*: error from incorrect assumptions to make target function easier to learn (high bias $\to$ missing relevant relations or underfitting)
- *Variance*: error from sensitivity to fluctuations in the dataset, or how much the target estimate would differ if different training data was used (high variance $\to$ modeling noise or overfitting)



No Free Lunch Theorem No single machine learning algorithm is better than all the others on all problems. It is common to try multiple models and find one that works best for a particular problem.

Thinking Like Nate Silver

1. Think Probabilistically Probabilistic forecasts are more meaningful than concrete statements and should be reported as probability distributions (including σ along with mean prediction μ.
2. Incorporate New Information Use live models, which continually updates using new information. To update, use Bayesian reasoning to calculate how probabilities change in response to new evidence.
3. Look For Consensus Forecast Use multiple distinct sources of evidence. Ssome models operate this way, such as boosting and bagging, which uses large number of weak classifiers to produce a strong one.

Modeling- Taxonomy

There are many different types of models. It is important to understand the trade-offs and when to use a certain type of model.

Parametric vs. Nonparametric

- **Parametric**: models that first make an assumption about a function form, or shape, of f (linear). Then fits the model. This reduces estimating f to just estimating set of parameters, but if our assumption was wrong, will lead to bad results.
- **Non-Parametric**: models that don't make any assumptions about f, which allows them to fit a wider range of shapes; but may lead to overfitting

Supervised vs. Unsupervised

- **Supervised**: models that fit input variables $x_i = (x_1, x_2, ...x_n)$ to a known output variables $y_i = (y_1, y_2, ...y_n)$
- **Unsupervised**: models that take in input variables $x_i = (x_1, x_2, ...x_n)$, but they do not have an associated output to supervise the training. The goal is understand relationships between the variables or observations.

Blackbox vs. Descriptive

- **Blackbox**: models that make decisions, but we do not know what happens "under the hood" e.g. deep learning, neural networks
- **Descriptive**: models that provide insight into *why* they make their decisions e.g. linear regression, decision trees

First-Principle vs. Data-Driven

- **First-Principle**: models based on a prior belief of how the system under investigation works, incorporates domain knowledge (ad-hoc)
- **Data-Driven**: models based on observed correlations between input and output variables

Deterministic vs. Stochastic

- **Deterministic**: models that produce a single "prediction" e.g. yes or no, true or false
- **Stochastic**: models that produce probability distributions over possible events

Flat vs. Hierarchical

- **Flat**: models that solve problems on a single level, no notion of subproblems
- **Hierarchical**: models that solve several different nested subproblems

Modeling- Evaluation Metrics

Need to determine how good our model is. Best way to assess models is out-of-sample predictions (data points your model has never seen).

Classification

	Predicted Yes	Predicted No
Actual Yes	True Positives (TP)	False Negatives (FN)
Actual No	False Positives (FP)	True Negatives (TN)

Accuracy: ratio of correct predictions over total predictions. Misleading when class sizes are substantially different. $accuracy = \frac{TP+TN}{TP+TN+FN+FP}$
Precision: how often the classifier is correct when it predicts positive: $precision = \frac{TP}{TP+FP}$
Recall: how often the classifier is correct for all positive instances: $recall = \frac{TP}{TP+FN}$
F-Score: single measurement to describe performance: $F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$
ROC Curves: plots true positive rates and false positive rates for various thresholds, or where the model determines if a data point is positive or negative (e.g. if >0.8, classify as positive). Best possible area under the ROC curve (AUC) is 1, while random is 0.5, or the main diagonal line.

Regression

Errors are defined as the difference between a prediction y′ and the actual result y.
Absolute Error: $\Delta = y' - y$
Squared Error: $\Delta^2 = (y' - y)^2$
Mean-Squared Error: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y'_i - y_i)^2$
Root Mean-Squared Error: RMSD $= \sqrt{MSE}$
Absolute Error Distribution: Plot absolute error distribution: should be symmetric, centered around 0, bell-shaped, and contain rare extreme outliers.

Modeling- Evaluation Environment

Evaluation metrics provides use with the tools to estimate errors, but what should be the process to obtain the best estimate? Resampling involves repeatedly drawing samples from a training set and refitting a model to each sample, which provides us with additional information compared to fitting the model once, such as obtaining a better estimate for the test error.

Key Concepts

Training Data: data used to fit your models or the set used for learning
Validation Data: data used to tune the parameters of a model
Test Data: data used to evaluate how good your model is. Ideally your model should never touch this data until final testing/evaluation

Cross Validation

Class of methods that estimate test error by holding out a subset of training data from the fitting process.
Validation Set: split data into training set and validation set. Train model on training and estimate test error using validation. e.g. 80-20 split
Leave-One-Out CV (LOOCV): split data into training set and validation set, but the validation set consists of 1 observation. Then repeat n-1 times until all observations have been used as validation. Test erro is the average of these n test error estimates.
k-Fold CV: randomly divide data into k groups (folds) of approximately equal size. First fold is used as validation and the rest as training. Then repeat k times and find average of the k estimates.

Bootstrapping

Methods that rely on random sampling with replacement. Bootstrapping helps with quantifying uncertainty associated with a given estimate or model.

Amplifying Small Data Sets

What can we do it we don't have enough data?

- **Create Negative Examples**- e.g. classifying presidential candidates, most people would be unqualified so label most as unqualified
- **Synthetic Data**- create additional data by adding noise to the real data

Linear Regression

Linear regression is a simple and useful tool for predicting a quantitative response. The relationship between input variables $X = (X_1, X_2, ...X_p)$ and output variable Y takes the form:

$$Y \approx \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \epsilon$$

$\beta_0 ... \beta_p$ are the unknown coefficients (parameters) which we are trying to determine. The best coefficients will lead us to the best "fit", which can be found by minimizing the *residual sum squares* (RSS), or the sum of the differences between the actual ith value and the predicted ith value. $\text{RSS} = \sum_{i=1}^{n} e_i$, where $e_i = y_i - \hat{y}_i$

How to find best fit?

Matrix Form: We can solve the closed-form equation for coefficient vector w: $w = (X^T X)^{-1} X^T Y$. X represents the input data and Y represents the output data. This method is used for smaller matrices, since inverting a matrix is computationally expensive.

Gradient Descent: First-order optimization algorithm. We can find the minimum of a *convex* function by starting at an arbitrary point and repeatedly take steps in the downward direction, which can be found by taking the negative direction of the gradient. After several iterations, we will eventually converge to the minimum. In our case, the minimum corresponds to the coefficients with the minimum error, or the best line of fit. The learning rate α determines the size of the steps we take in the downward direction.

Gradient descent algorithm in two dimensions. Repeat until convergence.

1. $w_0^{t+1} := w_0^t - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$
2. $w_1^{t+1} := w_1^t - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1)$

For non-convex functions, gradient descent no longer guarantees an optimal solutions since there may be local minimas. Instead, we should run the algorithm from different starting points and use the best local minima we find for the solution.

Stochastic Gradient Descent: instead of taking a step after sampling the *entire* training set, we take a small batch of training data at random to determine our next step. Computationally more efficient and may lead to faster convergence.

Linear Regression II

Improving Linear Regression

Subset/Feature Selection: approach involves identifying a subset of the p predictors that we believe to be best related to the response. Then we fit model using the reduced set of variables.

- Best, Forward, and Backward Subset Selection

Shrinkage/Regularization: all variables are used, but estimated coefficients are shrunken towards zero relative to the least squares estimate. λ represents the tuning parameter- as λ increases, flexibility decreases $\rightarrow$ decreased variance but increased bias. The tuning parameter is key in determining the sweet spot between under and over-fitting. In addition, while Ridge will always produce a model with p variables, Lasso can force coefficients to be equal to zero.

- Lasso (L1): min RSS + $\lambda \sum_{j=1}^{p} |\beta_j|$
- Ridge (L2): min RSS + $\lambda \sum_{j=1}^{p} \beta_j^2$

Dimension Reduction: projecting p predictors into a M-dimensional subspace, where M < p. This is achieved by computing M different linear combinations of the variables. Can use PCA.

Miscellaneous: Removing outliers, feature scaling, removing multicollinearity (correlated variables)

Evaluating Model Accuracy

Residual Standard Error (RSE): $\text{RSE} = \sqrt{\frac{1}{n-2} RSS}$. Generally, the smaller the better.

R^2: Measure of fit that represents the proportion of variance explained, or the *variability in Y that can be explained using X*. It takes on a value between 0 and 1. Generally the higher the better. $R^2 = 1 - \frac{RSS}{TSS}$, where Total Sum of Squares (TSS) = $\sum (y_i - \bar{y})^2$

Evaluating Coefficient Estimates

Standard Error (SE) of the coefficients can be used to perform hypothesis tests on the coefficients:

H_0: No relationship between X and Y, H_a: Some relationship exists. A p-value can be obtained and can be interpreted as follows: a small p-value indicates that a relationship between the predictor (X) and the response (Y) exists. Typical p-value cutoffs are around 5 or 1 %.

Logistic Regression

Logistic regression is used for classification, where the response variable is categorical rather than numerical.

The model works by predicting the probability that Y belongs to a particular category by first fitting the data to a linear regression model, which is then passed to the logistic function (below). The logistic function will always produce a S-shaped curve, so regardless of X, we can always obtain a sensible answer (between 0 and 1). If the probability is above a certain predetermined threshold (e.g. P(Yes) > 0.5), then the model will predict Yes.

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + ... + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + ... + \beta_p X_p}}$$

How to find best coefficients?

Maximum Likelihood: The coefficients $\beta_0 ... \beta_p$ are unknown and must be estimated from the training data. We seek estimates for $\beta_0 ... \beta_p$ such that the predicted probability $\hat{p}(x_i)$ of each observation is a number close to one if its observed in a certain class and close to zero otherwise. This is done by maximizing the likelihood function:

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=1} (1 - p(x_i))$$

Potential Issues

Imbalanced Classes: imbalance in classes in training data lead to poor classifiers. It can result in a lot of false positives and also lead to few training data. Solutions include forcing balanced data by removing observations from the larger class, replicate data from the smaller class, or heavily weigh the training examples toward instances of the larger class.

Multi-Class Classification: the more classes you try to predict, the harder it will be for the the classifier to be effective. It is possible with logistic regression, but another approach, such as Linear Discriminant Analysis (LDA), may prove better.

Cheat Sheet – Ensemble Learning in ML

What is Ensemble Learning? Wisdom of the crowd

Combine multiple weak models/learners into one predictive model to reduce bias, variance and/or improve accuracy.

Types of Ensemble Learning: N number of weak learners

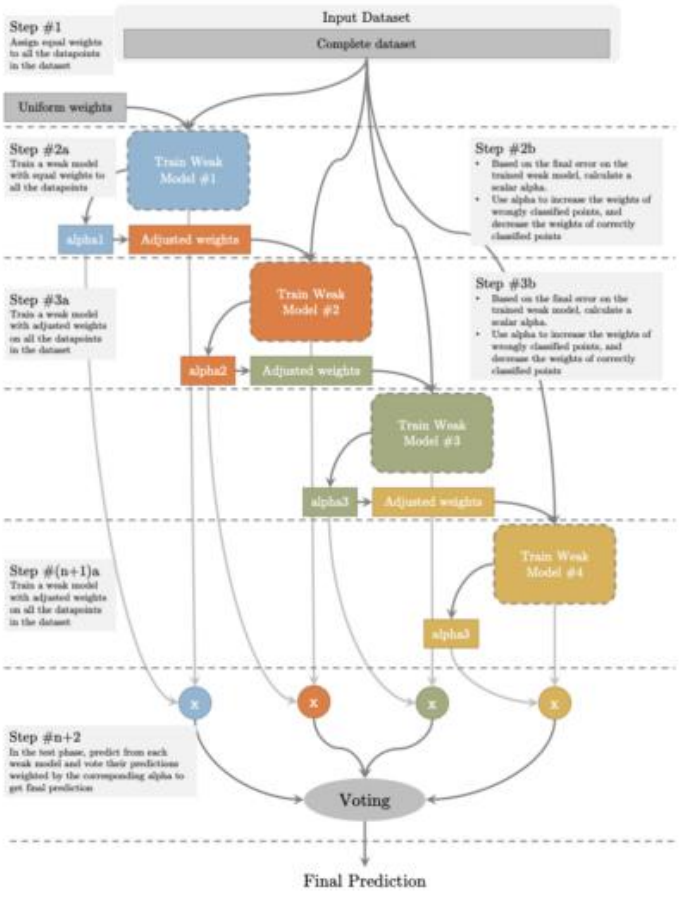
1.Bagging: Trains N different weak models (usually of same types – homogenous) with N non-overlapping subset of the input dataset in parallel. In the test phase, each model is evaluated. The label with the greatest number of predictions is selected as the prediction. Bagging methods reduces variance of the prediction

2.Boosting: Trains N different weak models (usually of same types – homogenous) with the complete dataset in a sequential order. The datapoints wrongly classified with previous weak model is provided more weights to that they can be classified by the next weak leaner properly. In the test phase, each model is evaluated and based on the test error of each weak model, the prediction is weighted for voting. Boosting methods decreases the bias of the prediction.

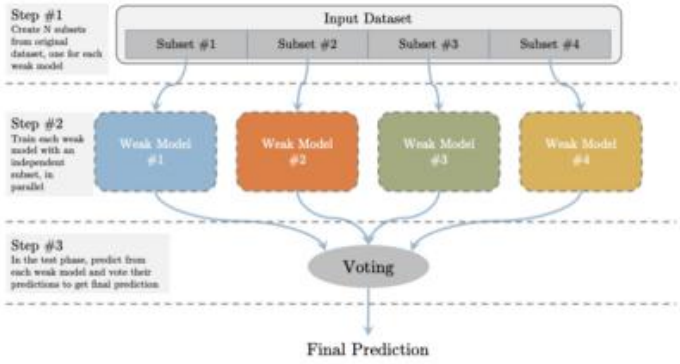
3.Stacking: Trains N different weak models (usually of different types – heterogenous) with one of the two subsets of the dataset in parallel. Once the weak learners are trained, they are used to trained a meta learner to combine their predictions and carry out final prediction using the other subset. In test phase, each model predicts its label, these set of labels are fed to the meta learner which generates the final prediction.

The block diagrams, and comparison table for each of these three methods can be seen below.

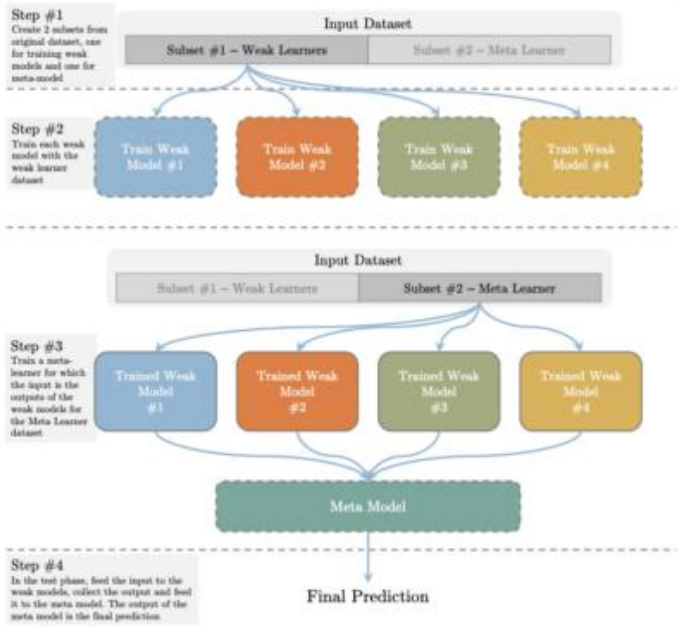
Ensemble Method – Boosting



Ensemble Method – Bagging



Ensemble Method – Stacking



Parameter	Bagging	Boosting	Stacking
Focuses on	Reducing variance	Reducing bias	Improving accuracy
Nature of weak learners is	Homogenous	Homogenous	Heterogenous
Weak learners are aggregated by	Simple voting	Weighted voting	Learned voting (meta-learner)

Source: https://www.cheatsheets.aqeel-anwar.com

Machine Learning Part I

Comparing ML Algorithms

Power and Expressibility: ML methods differ in terms of complexity. Linear regression fits linear functions while NN define piecewise-linear separation boundaries. More complex models can provide more accurate models, but at the risk of overfitting.

Interpretability: some models are more transparent and understandable than others (white box vs. black box models)

Ease of Use: some models feature few parameters/decisions (linear regression/NN), while others require more decision making to optimize (SVMs)

Training Speed: models differ in how fast they fit the necessary parameters

Prediction Speed: models differ in how fast they make predictions given a query

Method	Power of Expression	Ease of Interpretation	Ease of Use	Training Speed	Prediction Speed
Linear Regression	5	9	9	9	9
Nearest Neighbor	5	9	8	10	2
Naive Bayes	4	8	7	9	8
Decision Trees	8	8	7	7	9
Support Vector Machines	8	6	6	7	7
Boosting	9	6	6	6	6
Graphical Models	9	8	3	4	4
Deep Learning	10	3	4	3	7

Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features.

Problem: Suppose we need to classify vector $X = x_1...x_n$ into m classes, $C_1...C_m$. We need to compute the probability of each possible class given X, so we can assign X the label of the class with highest probability. We can calculate a probability using the Bayes' Theorem:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Where:

1. $P(C_i)$: the prior probability of belonging to class i
2. $P(X)$: normalizing constant, or probability of seeing the given input vector over all possible input vectors
3. $P(X|C_i)$: the conditional probability of seeing input vector X given we know the class is C_i

The prediction model will formally look like:

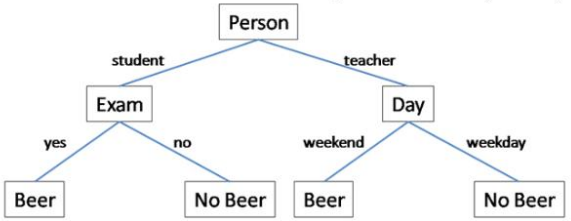
$$C(X) = argmax_{i \in classes(t)} \frac{P(X|C_i)P(C_i)}{P(X)}$$

where C(X) is the prediction returned for input X.

Machine Learning Part II

Decision Trees

Binary branching structure used to classify an arbitrary input vector X. Each node in the tree contains a simple feature comparison against some field ($x_i > 42$?). Result of each comparison is either true or false, which determines if we should proceed along to the left or right child of the given node. Also known as sometimes called classification and regression trees (CART).



Advantages: Non-linearity, support for categorical variables, easy to interpret, application to regression.

Disadvantages: Prone to overfitting, instable (not robust to noise), high variance, low bias

Note: rarely do models just use one decision tree. Instead, we aggregate many decision trees using methods like ensembling, bagging, and boosting.

Ensembles, Bagging, Random Forests, Boosting

Ensemble learning is the strategy of combining many different classifiers/models into one predictive model. It revolves around the idea of voting: a so-called "wisdom of crowds" approach. The most predicted class will be the final prediction.

Bagging: ensemble method that works by taking B bootstrapped subsamples of the training data and constructing B trees, each tree training on a distinct subsample as

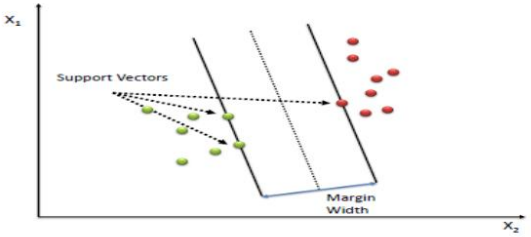
Random Forests: builds on bagging by decorrelating the trees. We do everything the same like in bagging, but when we build the trees, everytime we consider a split, a random sample of the p predictors is chosen as split candidates, not the full set (typically m $\approx \sqrt{p}$). When m = p, then we are just doing bagging.

Boosting: the main idea is to improve our model where it is not performing well by using information from previously constructed classifiers. Slow learner. Has 3 tuning parameters: number of classifiers B, learning parameter λ, interaction depth d (controls interaction order of model).

Machine Learning Part III

Support Vector Machines

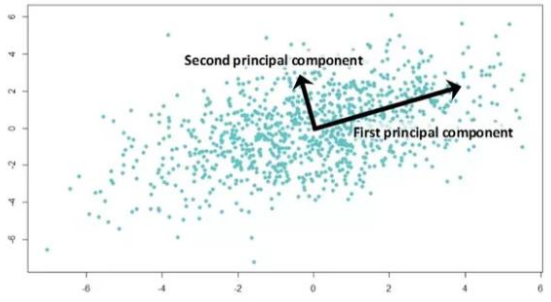
Work by constructing a hyperplane that separates points between two classes. The hyperplane is determined using the maximal margin hyperplane, which is the hyperplane that is the maximum distance from the training observations. This distance is called the margin. Points that fall on one side of the hyperplane are classified as -1 and the other +1.



Principal Component Analysis (PCA)

Principal components allow us to summarize a set of correlated variables with a smaller set of variables that collectively explain most of the variability in the original set. Essentially, we are "dropping" the least important feature variables.

Principal Component Analysis is the process by which principal components are calculated and the use of them to analyzing and understanding the data. PCA is an unsupervised approach and is used for dimensionality reduction, feature extraction, and data visualization. Variables after performing PCA are independent. Scaling variables is also important while performing PCA.



Machine Learning Part IV

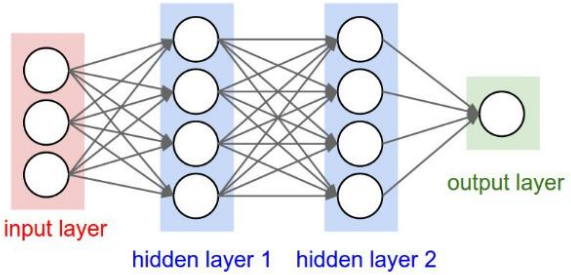
ML Terminology and Concepts

Features: input data/variables used by the ML model
Feature Engineering: transforming input features to be more useful for the models. e.g. mapping categories to buckets, normalizing between -1 and 1, removing null
Train/Eval/Test: training is data used to optimize the model, evaluation is used to asses the model on new data during training, test is used to provide the final result
Classification/Regression: regression is prediction a number (e.g. housing price), classification is prediction from a set of categories(e.g. predicting red/blue/green)
Linear Regression: predicts an output by multiplying and summing input features with weights and biases
Logistic Regression: similar to linear regression but predicts a probability
Overfitting: model performs great on the input data but poorly on the test data (combat by dropout, early stopping, or reduce # of nodes or layers)
Bias/Variance: how much output is determined by the features. more variance often can mean overfitting, more bias can mean a bad model
Regularization: variety of approaches to reduce overfitting, including adding the weights to the loss function, randomly dropping layers (dropout)
Ensemble Learning: training multiple models with different parameters to solve the same problem
A/B testing: statistical way of comparing 2+ techniques to determine which technique performs better and also if difference is statistically significant
Baseline Model: simple model/heuristic used as reference point for comparing how well a model is performing
Bias: prejudice or favoritism towards some things, people, or groups over others that can affect collection/sampling and interpretation of data, the design of a system, and how users interact with a system
Dynamic Model: model that is trained online in a continuously updating fashion
Static Model: model that is trained offline
Normalization: process of converting an actual range of values into a standard range of values, typically -1 to +1
Independently and Identically Distributed (i.i.d): data drawn from a distribution that doesn't change, and where each value drawn doesn't depend on previously drawn values; ideal but rarely found in real life
Hyperparameters: the "knobs" that you tweak during successive runs of training a model
Generalization: refers to a model's ability to make correct predictions on new, previously unseen data as opposed to the data used to train the model
Cross-Entropy: quantifies the difference between two probability distributions

Deep Learning Part I

What is Deep Learning?

Deep learning is a subset of machine learning. One popular DL technique is based on Neural Networks (NN), which loosely mimic the human brain and the code structures are arranged in layers. Each layer's input is the previous layer's output, which yields progressively higher-level features and defines a hierarchy. A Deep Neural Network is just a NN that has more than 1 hidden layer.



Recall that statistical learning is all about approximating $f(X)$. Neural networks are known as **universal approximators**, meaning no matter how complex a function is, there exists a NN that can (approximately) do the job. We can increase the approximation (or complexity) by adding more hidden layers and neurons.

Popular Architectures

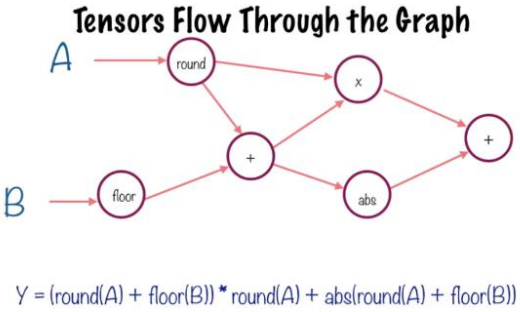
There are different kinds of NNs that are suitable for certain problems, which depend on the NN's architecture.

Linear Classifier: takes input features and combines them with weights and biases to predict output value
DNN: deep neural net, contains intermediate layers of nodes that represent "hidden features" and activation functions to represent non-linearity
CNN: convolutional NN, has a combination of convolutional, pooling, dense layers. popular for image classification.
Transfer Learning: use existing trained models as starting points and add additional layers for the specific use case. idea is that highly trained existing models know general features that serve as a good starting point for training a small network on specific examples
RNN: recurrent NN, designed for handling a sequence of inputs that have "memory" of the sequence. LSTMs are a fancy version of RNNs, popular for NLP
GAN: general adversarial NN, one model creates fake examples, and another model is served both fake example and real examples and is asked to distinguish
Wide and Deep: combines linear classifiers with deep neural net classifiers, "wide" linear parts represent memorizing specific examples and "deep" parts represent understanding high level features

Deep Learning Part II

Tensorflow

Tensorflow is an open source software library for numerical computation using data flow graphs. Everything in TF is a graph, where nodes represent operations on data and edges represent the data. Phase 1 of TF is building up a computation graph and phase 2 is executing it. It is also distributed, meaning it can run on either a cluster of machines or just a single machine.
TF is extremely popular/suitable for working with Neural Networks, since the way TF sets up the computational graph pretty much resembles a NN.



Tensors

In a graph, tensors are the edges and are multidimensional data arrays that flow through the graph. Central unit of data in TF and consists of a set of primitive values shaped into an array of any number of dimensions.
A tensor is characterized by its rank (# dimensions in tensor), shape (# of dimensions and size of each dimension), data type (data type of each element in tensor).

Placeholders and Variables

Variables: best way to represent shared, persistent state manipulated by your program. These are the parameters of the ML model are altered/trained during the training process. Training variables.
Placeholders: way to specify inputs into a graph that hold the place for a Tensor that will be fed at runtime. They are assigned once, do not change after. Input nodes

Deep Learning Part III

Deep Learning Terminology and Concepts

Neuron: node in a NN, typically taking in multiple input values and generating one output value, calculates the output value by applying an activation function (nonlinear transformation) to a weighted sum of input values
Weights: edges in a NN, the goal of training is to determine the optimal weight for each feature; if weight = 0, corresponding feature does not contribute
Neural Network: composed of neurons (simple building blocks that actually "learn"), contains activation functions that makes it possible to predict non-linear outputs
Activation Functions: mathematical functions that introduce non-linearity to a network e.g. RELU, tanh
Sigmoid Function: function that maps very negative numbers to a number very close to 0, huge numbers close to 1, and 0 to .5. Useful for predicting probabilities
Gradient Descent/Backpropagation: fundamental loss optimizer algorithms, of which the other optimizers are usually based. Backpropagation is similar to gradient descent but for neural nets
Optimizer: operation that changes the weights and biases to reduce loss e.g. Adagrad or Adam
Weights / Biases: weights are values that the input features are multiplied by to predict an output value. Biases are the value of the output given a weight of 0.
Converge: algorithm that converges will eventually reach an optimal answer, even if very slowly. An algorithm that doesn't converge may never reach an optimal answer.
Learning Rate: rate at which optimizers change weights and biases. High learning rate generally trains faster but risks not converging, whereas a lower rate trains slower
Numerical Instability: issues with very large/small values due to limits of floating point numbers in computers
Embeddings: mapping from discrete objects, such as words, to vectors of real numbers. useful because classifiers/neural networks work well on vectors of real numbers
Convolutional Layer: series of convolutional operations, each acting on a different slice of the input matrix
Dropout: method for regularization in training NNs, works by removing a random selection of some units in a network layer for a single gradient step
Early Stopping: method for regularization that involves ending model training early
Gradient Descent: technique to minimize loss by computing the gradients of loss with respect to the model's parameters, conditioned on training data
Pooling: Reducing a matrix (or matrices) created by an earlier convolutional layer to a smaller matrix. Pooling usually involves taking either the maximum or average value across the pooled area

Big Data- Hadoop Overview

Data can no longer fit in memory on one machine (monolithic), so a new way of computing was devised using a group of computers to process this "big data" (distributed). Such a group is called a cluster, which makes up server farms. All of these servers have to be coordinated in the following ways: partition data, coordinate computing tasks, handle fault tolerance/recovery, and allocate capacity to process.

Hadoop

Hadoop is an open source *distributed* processing framework that manages data processing and storage for big data applications running in clustered systems. It is comprised of 3 main components:

- **Hadoop Distributed File System (HDFS)**: a distributed file system that provides high-throughput access to application data by partitioning data across many machines
- **YARN**: framework for job scheduling and cluster resource management (task coordination)
- **MapReduce**: YARN-based system for parallel processing of large data sets on multiple machines

HDFS

Each disk on a different machine in a cluster is comprised of 1 master node and the rest are workers/data nodes. The **master node** manages the overall file system by storing the directory structure and the metadata of the files. The **data nodes** physically store the data. Large files are broken up and distributed across multiple machines, which are also replicated across multiple machines to provide fault tolerance.

MapReduce

Parallel programming paradigm which allows for processing of huge amounts of data by running processes on multiple machines. Defining a MapReduce job requires two stages: map and reduce.

- **Map**: operation to be performed in parallel on small portions of the dataset. the output is a key-value pair $< K, V >$
- **Reduce**: operation to combine the results of Map

YARN- Yet Another Resource Negotiator

Coordinates tasks running on the cluster and assigns new nodes in case of failure. Comprised of 2 subcomponents: the resource manager and the node manager. The **resource manager** runs on a single master node and schedules tasks across nodes. The **node manager** runs on all other nodes and manages tasks on the individual node.

Big Data- Hadoop Ecosystem

An entire ecosystem of tools have emerged around Hadoop, which are based on interacting with HDFS. Below are some popular ones:

Hive: data warehouse software built o top of Hadoop that facilitates reading, writing, and managing large datasets residing in distributed storage using SQL-like queries (HiveQL). Hive abstracts away underlying MapReduce jobs and returns HDFS in the form of tables (not HDFS).
Pig: high level scripting language (Pig Latin) that enables writing complex data transformations. It pulls unstructured/incomplete data from sources, cleans it, and places it in a database/data warehouses. Pig performs ETL into data warehouse while Hive queries from data warehouse to perform analysis (GCP: DataFlow).
Spark: framework for writing fast, distributed programs for data processing and analysis. Spark solves similar problems as Hadoop MapReduce but with a fast in-memory approach. It is an unified engine that supports SQL queries, streaming data, machine learning and graph processing. Can operate separately from Hadoop but integrates well with Hadoop. Data is processed using Resilient Distributed Datasets (RDDs), which are immutable, lazily evaluated, and tracks lineage. ;
Hbase: non-relational, NoSQL, column-oriented database management system that runs on top of HDFS. Well suited for sparse data sets (GCP: BigTable)
Flink/Kafka: stream processing framework. Batch streaming is for bounded, finite datasets, with periodic updates, and delayed processing. Stream processing is for unbounded datasets, with continuous updates, and immediate processing. Stream data and stream processing must be decoupled via a message queue. Can group streaming data (windows) using tumbling (non-overlapping time), sliding (overlapping time), or session (session gap) windows.
Beam: programming model to define and execute data processing pipelines, including ETL, batch and stream (continuous) processing. After building the pipeline, it is executed by one of Beam's distributed processing back-ends (Apache Apex, Apache Flink, Apache Spark, and Google Cloud Dataflow). Modeled as a Directed Acyclic Graph (DAG).
Oozie: workflow scheduler system to manage Hadoop jobs
Sqoop: transferring framework to transfer large amounts of data into HDFS from relational databases (MySQL)

TOP PREDICTION ALGORITHMS

TYPE	NAME	DESCRIPTION	ADVANTAGES	DISADVANTAGES
Linear	 Linear regression	**The "best fit" line** through all data points. Predictions are numerical.	Easy to understand -- you clearly see what the biggest drivers of the model are.	X Sometimes too simple to capture complex relationships between variables. X Tendency for the model to "overfit".
	 Logistic regression	The adaptation of **linear regression** to problems of classification (e.g., yes/no questions, groups, etc.)	Also easy to understand.	X Sometimes too simple to capture complex relationships between variables. X Tendency for the model to "overfit".
Tree-based	 Decision tree	A graph that uses a **branching method** to match all possible outcomes of a decision.	Easy to understand and implement.	X Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
	 Random Forest	Takes the average of many decision trees, each of which is made with a sample of the data. Each tree is weaker than a full decision tree, but **by combining them we get better overall performance.**	A sort of "wisdom of the crowd". Tends to result in very high quality models. Fast to train.	X Can be slow to output predictions relative to other algorithms. X Not easy to understand predictions.
	 Gradient Boosting	Uses even weaker decision trees, that are increasingly **focused on "hard" examples.**	High-performing.	X A small change in the feature set or training set can create radical changes in the model. X Not easy to understand predictions.
Neural networks	 Neural networks	Mimics the behavior of the brain. Neural networks are interconnected neurons that pass messages to each other. Deep learning uses several **layers of neural networks put one after the other.**	Can handle extremely complex tasks - no other algorithm comes close in image recognition.	X Very, very slow to train, because they have so many layers. Require a lot of power. X Almost impossible to understand predictions.

VIP Cheatsheet: Machine Learning Tips

Afshine Amidi and Shervine Amidi

September 9, 2018

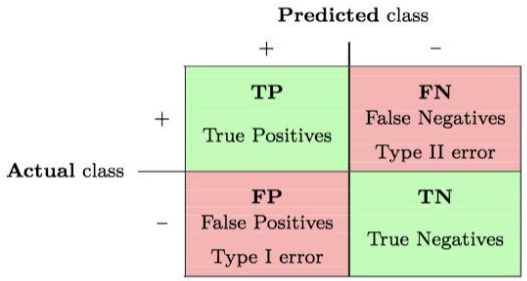
Metrics

Given a set of data points $\{x^{(1)}, ..., x^{(m)}\}$, where each $x^{(i)}$ has n features, associated to a set of outcomes $\{y^{(1)}, ..., y^{(m)}\}$, we want to assess a given classifier that learns how to predict y from x.

Classification

In a context of a binary classification, here are the main metrics that are important to track to assess the performance of the model.

❒ **Confusion matrix** – The confusion matrix is used to have a more complete picture when assessing the performance of a model. It is defined as follows:



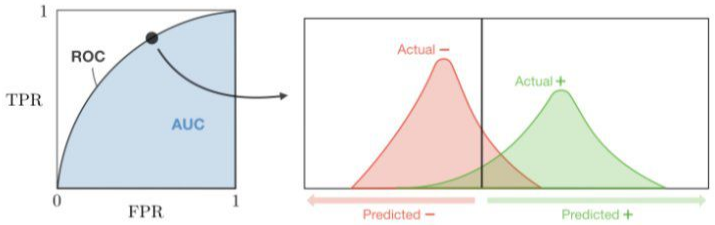
❒ **Main metrics** – The following metrics are commonly used to assess the performance of classification models:

Metric	Formula	Interpretation
Accuracy	$\frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$	Overall performance of model
Precision	$\frac{\text{TP}}{\text{TP}+\text{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{\text{TP}}{\text{TP}+\text{FN}}$	Coverage of actual positive sample
Specificity	$\frac{\text{TN}}{\text{TN}+\text{FP}}$	Coverage of actual negative sample
F1 score	$\frac{2\text{TP}}{2\text{TP}+\text{FP}+\text{FN}}$	Hybrid metric useful for unbalanced classes

❒ **ROC** – The receiver operating curve, also noted ROC, is the plot of TPR versus FPR by varying the threshold. These metrics are are summed up in the table below:

Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{\text{TP}}{\text{TP}+\text{FN}}$	Recall, sensitivity
False Positive Rate FPR	$\frac{\text{FP}}{\text{TN}+\text{FP}}$	1-specificity

❒ **AUC** – The area under the receiving operating curve, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:



Regression

❒ **Basic metrics** – Given a regression model f, the following metrics are commonly used to assess the performance of the model:

Total sum of squares	Explained sum of squares	Residual sum of squares
$\text{SS}_{\text{tot}} = \sum_{i=1}^{m}(y_i - \overline{y})^2$	$\text{SS}_{\text{reg}} = \sum_{i=1}^{m}(f(x_i) - \overline{y})^2$	$\text{SS}_{\text{res}} = \sum_{i=1}^{m}(y_i - f(x_i))^2$

❒ **Coefficient of determination** – The coefficient of determination, often noted R^2 or r^2, provides a measure of how well the observed outcomes are replicated by the model and is defined as follows:

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

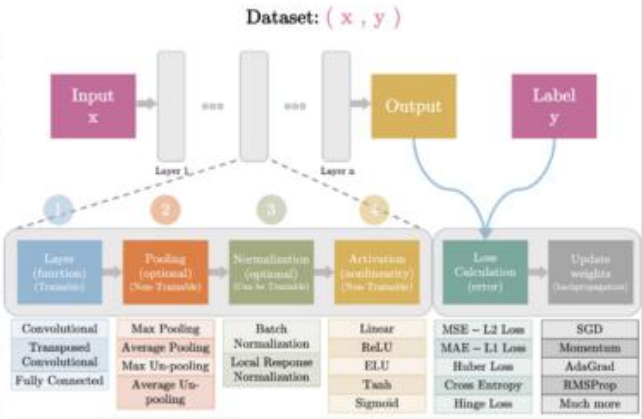
❒ **Main metrics** – The following metrics are commonly used to assess the performance of regression models, by taking into account the number of variables n that they take into consideration:

Mallow's Cp	AIC	BIC	Adjusted R^2
$\frac{\text{SS}_{\text{res}} + 2(n+1)\widehat{\sigma}^2}{m}$	$2\left[(n+2) - \log(L)\right]$	$\log(m)(n+2) - 2\log(L)$	$1 - \frac{(1-R^2)(m-1)}{m-n-1}$

Cheat Sheet – Convolutional Neural Network

Convolutional Neural Network:

The data gets into the CNN through the input layer and passes through various hidden layers before getting to the output layer. The output of the network is compared to the actual labels in terms of loss or error. The partial derivatives of this loss w.r.t the trainable weights are calculated, and the weights are updated through one of the various methods using backpropagation.



CNN Template:

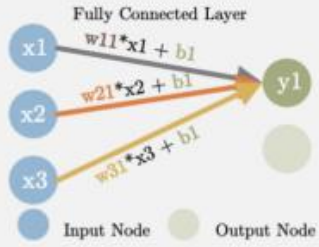
Most of the commonly used hidden layers (not all) follow a pattern

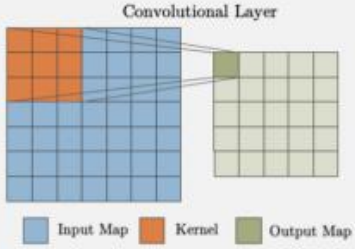
1. **Layer function:** Basic transforming function such as convolutional or fully connected layer.

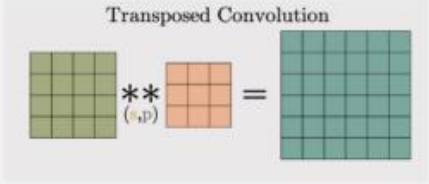
a. **Fully Connected:** Linear functions between the input and the output.

a. **Convolutional Layers:** These layers are applied to 2D (3D) input feature maps. The trainable weights are a 2D (3D) kernel/filter that moves across the input feature map, generating dot products with the overlapping region of the input feature map.

b. **Transposed Convolutional (DeConvolutional) Layer:** Usually used to increase the size of the output feature map (Upsampling) The idea behind the transposed convolutional layer is to undo (not exactly) the convolutional layer



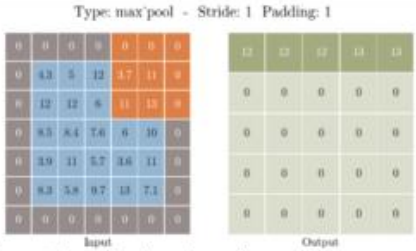




2. **Pooling:** Non-trainable layer to change the size of the feature map

a. **Max/Average Pooling:** Decrease the spatial size of the input layer based on selecting the maximum/average value in receptive field defined by the kernel

b. **UnPooling:** A non-trainable layer used to increase the spatial size of the input layer based on placing the input pixel at a certain index in the receptive field of the output defined by the kernel.



3. **Normalization:** Usually used just before the activation functions to limit the unbounded activation from increasing the output layer values too high

a. **Local Response Normalization LRN:** A **non-trainable layer** that square-normalizes the pixel values in a feature map within a local neighborhood.

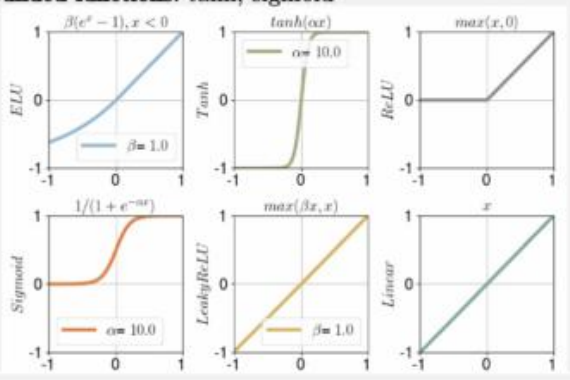
b. **Batch Normalization:** A trainable approach to normalizing the data by learning scale and shift variable during training.

3. **Activation:** Introduce non-linearity so CNN can efficiently map non-linear complex mapping.

a. **Non-parametric/Static functions:** Linear, ReLU

b. **Parametric functions:** ELU, tanh, sigmoid, Leaky ReLU

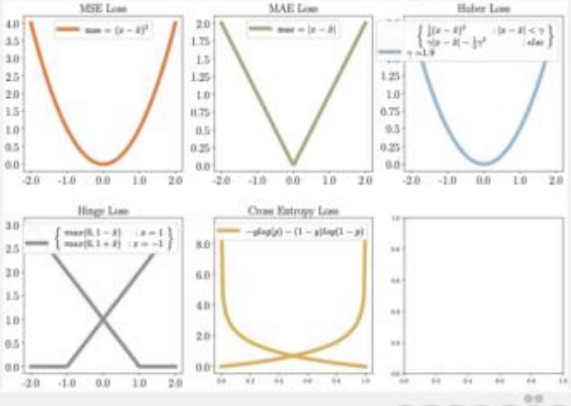
c. **Bounded functions:** tanh, sigmoid



5. **Loss function:** Quantifies how far off the CNN prediction is from the actual labels.

a. Regression Loss Functions: MAE, MSE, Huber loss

b. Classification Loss Functions: Cross entropy, Hinge loss



Source: https://www.cheatsheets.aqeel-anwar.com

Cheat Sheet – Famous CNNs

AlexNet – 2012

Why: AlexNet was born out of the need to improve the results of the ImageNet challenge.
What: The network consists of 5 Convolutional (CONV) layers and 3 Fully Connected (FC) layers. The activation used is the Rectified Linear Unit (ReLU).
How: Data augmentation is carried out to reduce over-fitting, Uses Local response localization.

AlexNet Network - Structural Details

Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool3	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
Total													62,378,344

VGGNet – 2014

Why: VGGNet was born out of the need to reduce the # of parameters in the CONV layers and improve on training time
What: There are multiple variants of VGGNet (VGG16, VGG19, etc.)
How: The important point to note here is that all the conv kernels are of size 3x3 and maxpool kernels are of size 2x2 with a stride of two.

VGG16 - Structural Details

#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3-64	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													138,423,208

ResNet – 2015

Why: Neural Networks are notorious for not being able to find a simpler mapping when it exists. ResNet solves that.
What: There are multiple versions of ResNetXX architectures where 'XX' denotes the number of layers. The most used ones are ResNet50 and ResNet101. Since the vanishing gradient problem was taken care of (more about it in the How part), CNN started to get deeper and deeper
How: ResNet architecture makes use of shortcut connections do solve the vanishing gradient problem. The basic building block of ResNet is a Residual block that is repeated throughout the network.

ResNet18 - Structural Details

#	Input Image			output			Layer	Stride	Pad	Kernel		in	out	Param
1	227	227	3	112	112	64	conv1	2	1	7	7	3	64	9472
	112	112	64	56	56	64	maxpool	2	0.5	3	3	64	64	0
2	56	56	64	56	56	64	conv2-1	1	1	3	3	64	64	36928
3	56	56	64	56	56	64	conv2-2	1	1	3	3	64	64	36928
4	56	56	64	56	56	64	conv2-3	1	1	3	3	64	64	36928
5	56	56	64	56	56	64	conv2-4	1	1	3	3	64	64	36928
6	56	56	64	28	28	128	conv3-1	2	0.5	3	3	64	128	73856
7	28	28	128	28	28	128	conv3-2	1	1	3	3	128	128	147584
8	28	28	128	28	28	128	conv3-3	1	1	3	3	128	128	147584
9	28	28	128	28	28	128	conv3-4	1	1	3	3	128	128	147584
10	28	28	128	14	14	256	conv4-1	2	0.5	3	3	128	256	295168
11	14	14	256	14	14	256	conv4-2	1	1	3	3	256	256	590080
12	14	14	256	14	14	256	conv4-3	1	1	3	3	256	256	590080
13	14	14	256	14	14	256	conv4-4	1	1	3	3	256	256	590080
14	14	14	256	7	7	512	conv5-1	2	0.5	3	3	256	512	1180160
15	7	7	512	7	7	512	conv5-2	1	1	3	3	512	512	2359808
16	7	7	512	7	7	512	conv5-3	1	1	3	3	512	512	2359808
17	7	7	512	7	7	512	conv5-4	1	1	3	3	512	512	2359808
	7	7	512	1	1	512	avg_pool	1	0	7	7	512	512	0
18	1	1	512	1	1	1000	fc					512	1000	513000
Total														11,511,784

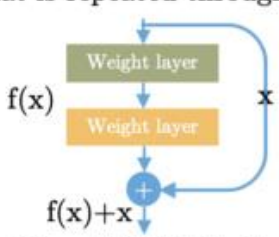


Figure 1 ResNet Block

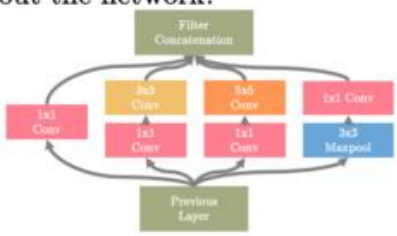
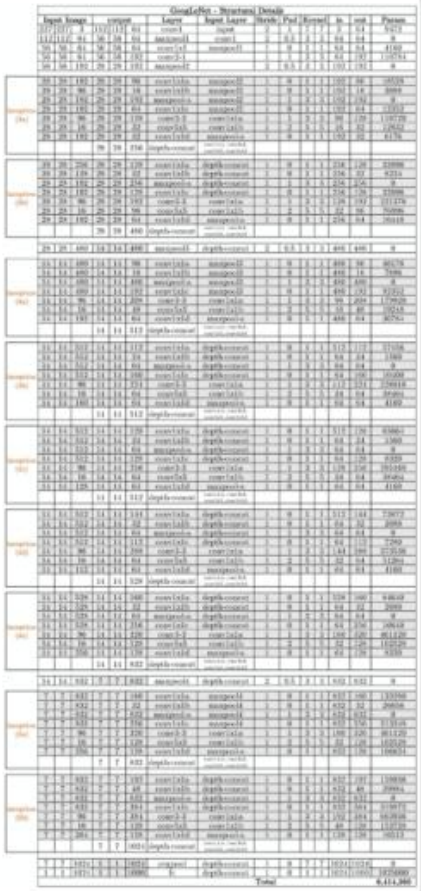


Figure 2 Inception Block

Inception – 2014

Why: Lager kernels are preferred for more global features, on the other hand, smaller kernels provide good results in detecting area-specific features. For effective recognition of such a variable-sized feature, we need kernels of different sizes. That is what Inception does.
What: The Inception network architecture consists of several inception modules of the following structure. Each inception module consists of four operations in parallel, 1x1 conv layer, 3x3 conv layer, 5x5 conv layer, max pooling
How: Inception increases the network space from which the best network is to be chosen via training. Each inception module can capture salient features at different levels.

GoogLeNet - Structural Details



Comparison

Network	Year	Salient Feature	top5 accuracy	Parameters	FLOP
AlexNet	2012	Deeper	84.70%	62M	1.5B
VGGNet	2014	Fixed-size kernels	92.30%	138M	19.6B
Inception	2014	Wider - Parallel kernels	93.30%	6.4M	2B
ResNet-152	2015	Shortcut connections	95.51%	60.3M	11B

Source: https://www.cheatsheets.aqeel-anwar.com

VIP Cheatsheet: Deep Learning

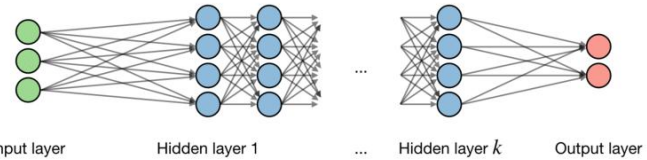
Afshine Amidi and Shervine Amidi

September 15, 2018

Neural Networks

Neural networks are a class of models that are built with layers. Commonly used types of neural networks include convolutional and recurrent neural networks.

❒ **Architecture** – The vocabulary around neural networks architectures is described in the figure below:

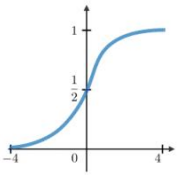
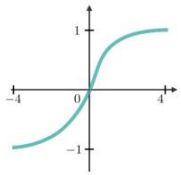
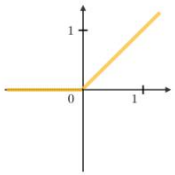
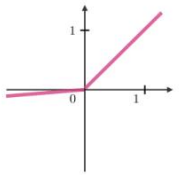


By noting i the i^{th} layer of the network and j the j^{th} hidden unit of the layer, we have:

$$\boxed{z_j^{[i]} = {w_j^{[i]}}^T x + b_j^{[i]}}$$

where we note w, b, z the weight, bias and output respectively.

❒ **Activation function** – Activation functions are used at the end of a hidden unit to introduce non-linear complexities to the model. Here are the most common ones:

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0,z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$
			

❒ **Cross-entropy loss** – In the context of neural networks, the cross-entropy loss $L(z,y)$ is commonly used and is defined as follows:

$$\boxed{L(z,y) = -\Big[y\log(z) + (1-y)\log(1-z)\Big]}$$

❒ **Learning rate** – The learning rate, often noted η, indicates at which pace the weights get updated. This can be fixed or adaptively changed. The current most popular method is called Adam, which is a method that adapts the learning rate.

❒ **Backpropagation** – Backpropagation is a method to update the weights in the neural network by taking into account the actual output and the desired output. The derivative with respect to weight w is computed using chain rule and is of the following form:

$$\boxed{\frac{\partial L(z,y)}{\partial w} = \frac{\partial L(z,y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}}$$

As a result, the weight is updated as follows:

$$\boxed{w \longleftarrow w - \eta \frac{\partial L(z,y)}{\partial w}}$$

❒ **Updating weights** – In a neural network, weights are updated as follows:

- Step 1: Take a batch of training data.
- Step 2: Perform forward propagation to obtain the corresponding loss.
- Step 3: Backpropagate the loss to get the gradients.
- Step 4: Use the gradients to update the weights of the network.

❒ **Dropout** – Dropout is a technique meant at preventing overfitting the training data by dropping out units in a neural network. In practice, neurons are either dropped with probability p or kept with probability $1-p$.

Convolutional Neural Networks

❒ **Convolutional layer requirement** – By noting W the input volume size, F the size of the convolutional layer neurons, P the amount of zero padding, then the number of neurons N that fit in a given volume is such that:

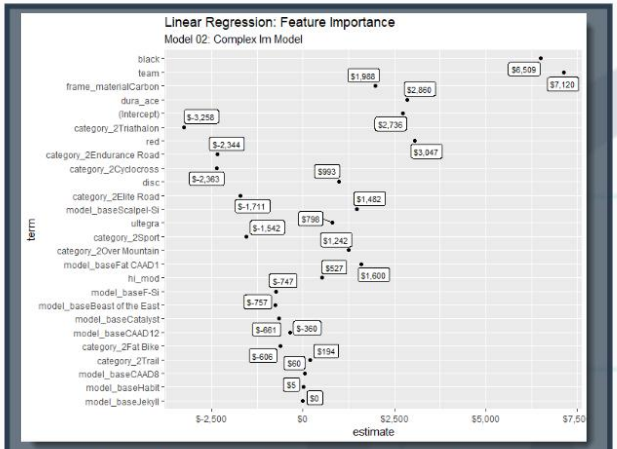
$$\boxed{N = \frac{W - F + 2P}{S} + 1}$$

❒ **Batch normalization** – It is a step of hyperparameter γ, β that normalizes the batch $\{x_i\}$. By noting μ_B, σ_B^2 the mean and variance of that we want to correct to the batch, it is done as follows:

$$\boxed{x_i \longleftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta}$$

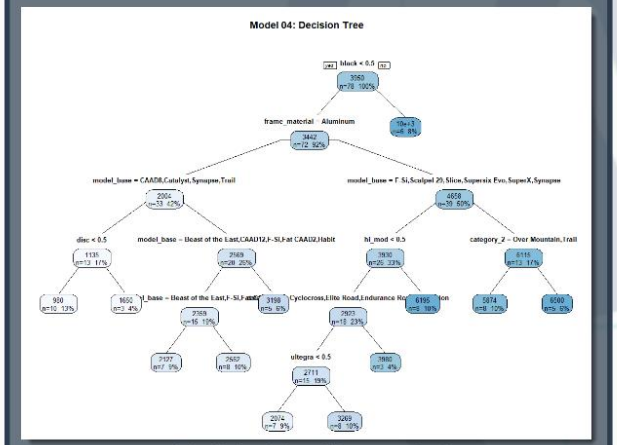
It is usually done after a fully connected/convolutional layer and before a non-linearity layer and aims at allowing higher learning rates and reducing the strong dependence on initialization.

Regression (Machine Learning)



Linear Regression with Multiple Predictors

Each predictor (term) is interpretable meaning the value (estimate) indicates an increase/decrease in the target



Decision Tree (Regression)

Decisions are based on binary rules that split the nodes and terminate at leaves. Regression trees estimate the value at each node.

Summary:

- **Common Applications in Business**: Used to predict a *numeric value* (e.g. forecasting sales, estimating prices, etc).
- **Key Concept:** Data is usually in a rectangular format (like a spreadsheet) with one column that is a *target* (e.g. price) and other columns that are *predictors* (e.g. product category)
- **Gotchas:**
  - **Preprocessing:** Knowing when to preprocess data (normalize) prior to machine learning step
  - **Feature Engineering:** Getting good features is more important than applying complex models.
- **Parameter Tuning:** Higher complexity models have many parameters that can be tuned.
- **Interpretability:** Some models are more explainable than others, meaning the estimates for each feature means something in relation to the target. Other models are not interpretable and require additional tools (e.g. LIME) to explain.

Terminology:

- **Supervised vs Unsupervised:** Regression is a supervised technique that requires training with a "**target**" (e.g. price of product or sales by month). The algorithm learns by identifying relationships between the target & the **predictors** (attributes related to the target like category of product or month of sales).
- **Classification vs Regression:** Classification aims to predict classes (either binary yes/no or multi-class categorical). Regression aims to predict a numeric value (e.g. product price = $4,233).
- **Preprocessing**: Many algorithms require preprocessing, which transforms the data into a format more suitable for the machine learning algorithm. A common example is "**standardization**" or scaling the feature to be in a range of [0,1] (or close to it).
- **Hyper Parameter & Tuning:** Machine learning algorithms have many parameters that can be adjusted (e.g. learning rate in GBM). Tuning is the process of systematically finding the optimum parameter values.
- **Cross Validation**: Machine learning algorithms should be tuned on a validation set as opposed to a test set. Cross-validation is the process of splitting the training set into multiple sets using a portion of the training set for tuning.
- **Performance Metrics (Regression)**: Common performance metrics are **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)**. These measures provide an estimate of model performance to compare models to each other.

R Cheat Sheet

Parsnip (Machine Learning):

- Model List (start here first)
- Linear Regression & GLM
- Decision Tree
- Random Forest
- Boosted Trees (XGBoost)
- SVM: Poly & Radial

Keras (Deep Learning)

H2O (ML & DL Framework)

MLR (ML Framework)

Python Cheat Sheet

Scikit-Learn (Machine Learning):

- Linear Regression
- GLM (Elastic Net)
- Decision Tree (Regressor)
- Random Forest (Regressor)
- AdaBoost (Regressor), XGBoost
- SVM (Regressor)

Keras (Deep Learning)

H2O (ML & DL Framework)

Resources

- Business Analysis With R Course (DS4B 101-R) - Modeling - Week 6
- Business Science Problem Framework
- Ultimate R Cheat Sheet | Ultimate Python Cheat Sheet

Data Science Courses for Business

Business Science University
university.business-science.io

version: 1.1