Machine Learning Cheatsheet

Summary of Machine Learning Algorithms descriptions, advantages and use cases. Inspired by the very good book and articles of *MachineLearningMastery*, with added math, and *ML Pros & Cons* of *HackingNote*. Design inspired by *The Probability Cheatsheet* of W. Chen. Written by Rémi Canard.

General

Definition

We want to learn a target function ***f*** that maps input variables ***X*** to output variable ***Y***, with an error ***e***:

$$Y = f(X) + e$$

Linear, Nonlinear

Different algorithms make different assumptions about the shape and structure of *f*, thus the need of testing several methods. Any algorithm can be either:

- **Parametric** (or **Linear**): simplify the mapping to a known linear combination form and learning its coefficients.

- **Non parametric** (or **Nonlinear**): free to learn any functional form from the training data, while maintaining some ability to generalize.

Linear algorithms are usually simpler, faster and requires less data, while Nonlinear can be are more flexible, more powerful and more performant.

Supervised, Unsupervised

Supervised learning methods learn to predict Y from X given that the data is labeled.

Unsupervised learning methods learn to find the inherent structure of the unlabeled data.

Bias-Variance trade-off

In supervised learning, the prediction error ***e*** is composed of the **bias**, the **variance** and the **irreducible** part.

Bias refers to **simplifying assumptions** made to learn the target function easily.

Variance refers to sensitivity of the model to changes in the training data.

The **goal of parameterization** is to achieve a **low bias** (underlying pattern not too simplified) and **low variance** (not sensitive to specificities of the training data) **tradeoff**.

Underfitting, Overfitting

In statistics, ***fit*** refers to how well the target function is approximated.

Underfitting refers to poor inductive learning from training data and poor generalization.

Overfitting refers to learning the training data detail and noise which leads to poor generalization. It can be **limited** by using resampling and defining a validation dataset.

Optimization

Almost every machine learning method has an optimization algorithm at its core.

Gradient Descent

Gradient Descent is used to **find the coefficients** of *f* that **minimizes a cost function** (for example MSE, SSR).

Procedure:

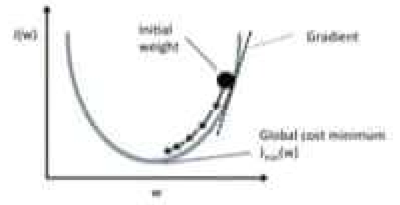
→ Initialization $\theta = 0$ (coefficients to 0 or random)

→ Calculate cost $J(\theta) = evaluate(f(coefficients))$

→ Gradient of cost $\frac{\partial}{\partial \theta_j} J(\theta)$ we know the uphill direction

→ Update coeff $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ we go downhill

The cost updating process is repeated until convergence (minimum found).



Batch Gradient Descend does summing/averaging of the cost over all the observations.

Stochastic Gradient Descent apply the procedure of parameter updating for each observation.

Tips:

- Change **learning rate** α ("size of jump" at each iteration)

- Plot *Cost vs Time* to assess learning rate performance

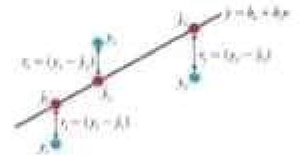
- Rescaling the input variables

- Reduce passes through training set with SGD

- Average over 10 or more updated to observe the learning trend while using SGD

Ordinary Least Squares

OLS is used to find the estimator $\hat{\beta}$ that **minimizes the sum of squared residuals:** $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{m}\beta_j x_{ij})^2 = y - X\hat{\beta}$



Using linear algebra such that we have $\hat{\beta} = (X^T X)^{-1} X^T y$

Maximum Likelihood Estimation

MLE is used to find the estimators that **minimizes the likelihood function:**

$\mathcal{L}(\theta|x) = f_\theta(x)$ density function of the data distribution

Linear Algorithms

All linear Algorithms assume a linear relationship between the input variables *X* and the output variable *Y*.

Linear Regression

Representation:

A LR model representation is a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i$$

β_0 is usually called intercept or **bias** coefficient. The dimension of the hyperplane of the regression is its **complexity**.

Usecase example:

- Prediction of customer churn

Nonlinear Algorithms

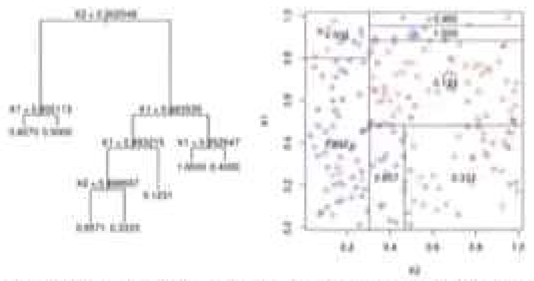
All Nonlinear Algorithms are non-parametric and more flexible. They are not sensible to outliers and do not require any shape of distribution.

Classification and Regression Trees

Also referred as CART or Decision Trees, this algorithm is the foundation of Random Forest and Boosted Trees.

Representation:

The model representation is a **binary tree**, where each **node** is an **input variable** *x* with a split point and each **leaf** contain an **output variable** *y* for prediction.



The model actually **split the input space** into (hyper) rectangles, and predictions are made according to the **area** observations *fall* into.

Learning:

Learning of a CART is done by a greedy approach called **recursive binary splitting** of the input space:

At each step, the best **predictor** X_j and the best **cutpoint** s are selected such that $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ **minimizes the cost**.

- For **regression** the cost is the **Sum of Squared Error:**

$$\sum_{i=1}^{n}(y_i - \hat{y})^2$$

- For **classification** the cost function is the **Gini index:**

$$G = \sum_{i=1}^{n} p_k(1 - p_k)$$

The Gini index is **an indication of how *pure* are the leaves**, if all observations are the same type G=0 (perfect purity), while a 50-50 split for binary would be G=0.5 (worst purity).

The most common **Stopping Criterion** for splitting is a minimum of **training observations per node**.

The simplest form of pruning is **Reduced Error Pruning:** Starting at the leaves, each node is replaced with its most popular class. If the prediction accuracy is not affected, then the change is kept

Advantages:

+ Easy to interpret and no overfitting with pruning

+ Works for both regression and classification problems

+ Can take any type of variables without modifications, and do not require any data preparation

Usecase examples:

- Fraudulent transaction classification

- Predict human resource allocation in companies

Naive Bayes Classifier

Naive Bayes is a **classification** algorithm interested in selecting the **best hypothesis *h* given data *d*** assuming there is no interaction between features.

Representation:

The representation is the based on Bayes Theorem:

$$P(h|d) = \frac{P(d|h) \times P(h)}{P(d)}$$

with naive hypothesis $P(h|d) = P(x_1|h) \times \ldots \times P(x_i|h)$

The prediction is the **Maximum A posteriori Hypothesis**:

$$MAP(h) = \max\big(P(h|d)\big) = \max\,(P(d|h) \times P(h))$$

The denominator is not kept as it is only for normalization.

Learning:

Training is **fast** because only **probabilities** need to be calculated:

$$P(h) = \frac{instances_h}{all\ instances} \text{ and } P(x|h) = \frac{count(x \wedge h)}{instances_h}$$

Variations:

Gaussian Naive Bayes can extend to numerical attributes by assuming a Gaussian distribution.

Instead of $P(x|h)$ are calculated with $P(h)$ during **learning**:

$$\mu(x) = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ and } \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu(x))^2}$$

and MAP for **prediction** is calculated using Gaussian PDF

$$f(x|\mu(x),\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Data preparation:

- Change numerical inputs to categorical (binning) or near-Gaussian inputs (remove outliers, log & boxcox transform)

- Other distributions can be used instead of Gaussian

- Log-transform of the probabilities can avoid overflow

- Probabilities can be updated as data becomes available

Advantages:

+ Fast because of the calculations

+ If the naive assumptions works can converge quicker than other models. Can be used on smaller training data.

+ Good for few categories variables

Usecase examples:

- Article classification using binary word presence

- Email spam detection using a similar technique

K-Nearest Neighbors

If you are similar to your neighbors, you are one of them.

Representation:

KNN uses the **entire training set, no training** is required.

Predictions are made by searching the ***k* similar instances**, according to a **distance**, and **summarizing the output**.

Learning:

Learning a LR means estimating the coefficients from the training data. Common methods include **Gradient Descent** or **Ordinary Least Squares**.

Variations:

There are extensions of LR training called **regularization** methods, that aim to **reduce the complexity** of the model:

- **Lasso Regression**: where OLS is modified to minimize the sum of the coefficients (L1 regularization)

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

- **Ridge Regression**: where OLS is modified to minimize the squared sum of the coefficients (L2 regularization)

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

where $\lambda \geq 0$ is a tuning parameter to be determined.

Data preparation:

- Transform data for linear relationship (ex: log transform for exponential relationship)
- Remove noise such as outliers
- Rescale inputs using standardization or normalization

Advantages:

+ Good regression baseline considering simplicity
+ Lasso/Ridge can be used to avoid overfitting
+ Lasso/Ridge permit feature selection in case of collinearity

Usecase examples:

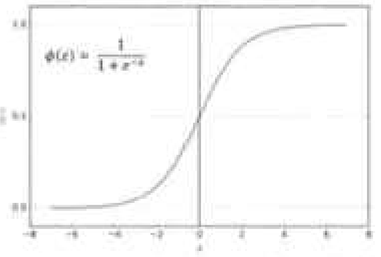
- Product sales prediction according to prices or promotions
- Call-center waiting-time prediction according to the number of complaints and the number of working agents

Logistic Regression

It is the go-to for **binary classification.**

Representation:

Logistic regression a linear method but predictions are transformed using the **logistic function** (or sigmoid):



ϕ is *S*-shaped and map real-valued number in (0,1).

The representation is an equation with binary output:

$$y = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}$$

Which actually models the probability of default class:

$$p(X) = \frac{e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}}{1+e^{\beta_0+\beta_1 x_1+\cdots+\beta_i x_i}} = p(Y=1|X)$$

Learning:

Learning the Logistic regression coefficients is done using **maximum-likelihood estimation**, to predict values close to 1 for default class and close to 0 for the other class.

Data preparation:

- Probability transformation to binary for classification
- Remove noise such as outliers

Advantages:

+ Good classification baseline considering simplicity
+ Possibility to change cutoff for precision/recall tradeoff
+ Robust to noise/overfitting with L1/L2 regularization
+ Probability output can be used for ranking

Usecase examples:

- Customer scoring with probability of purchase
- Classification of loan defaults according to profile

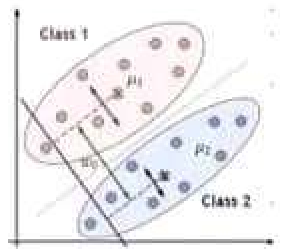
Linear Discriminant Analysis

For **multiclass classification**, LDA is the preferred linear technique.

Representation:

LDA representation consists of **statistical properties** calculated for **each class**: **means** and the **covariance matrix**:

$$\mu_k = \frac{1}{n_k}\sum_{i=1}^{n} x_i \quad \text{and} \quad \sigma^2 = \frac{1}{n-K}\sum_{i=1}^{n}(x_i - \mu_k)^2$$



LDA assumes **Gaussian** data and attributes of **same σ^2**.

Predictions are made using **Bayes Theorem:**

$$P(Y=k|X=x) = \frac{P(k)\times P(x|k)}{\sum_{l=1}^{K} P(l)\times P(x|l)}$$

to obtain a discriminate function (latent variable) for each class *k*, estimating $P(x|k)$ with a Gaussian distribution:

$$D_k(x) = x \times \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \ln(P(k))$$

The class with largest **discriminant value** is the **output class**.

Variations:

- **Quadratic DA**: Each class uses its own variance estimate
- **Regularized DA**: Regularization into the variance estimate

Data preparation:

- Review and modify univariate distributions to be Gaussian
- Standardize data to $\mu = 0, \sigma = 1$ to have same variance
- Remove noise such as outliers

Advantages:

+ Can be used for dimensionality reduction by keeping the latent variables as new variables

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

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

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!

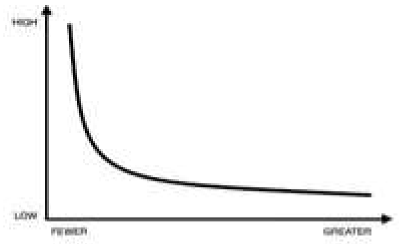
Classic Statistical Distributions

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.

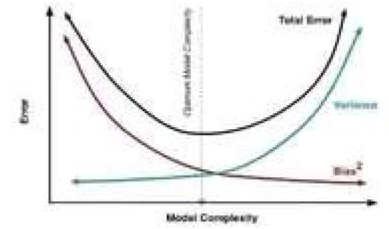
Modeling- Philosophies

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.

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:

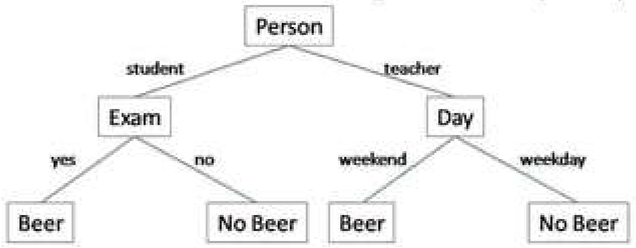
$$C(X) = argmax_{i \in classes(t)} \frac{P(X|C_i)P(C_i)}{P(X)}$$

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.

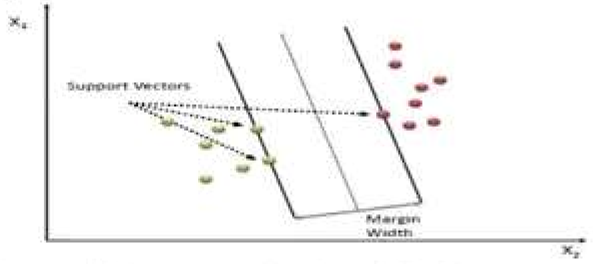
Ensembles, Bagging, Random Forests, Boosting

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

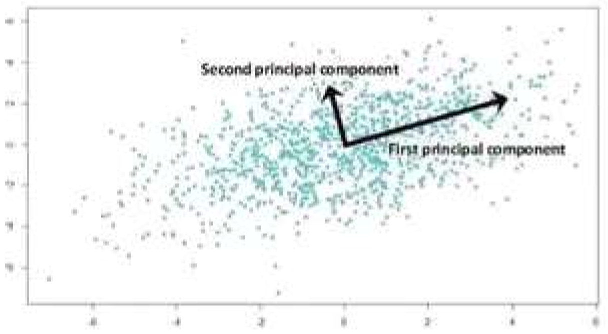
Work by constructing a hyperplane that separates points between two classes. The hyperplane is determined using the maximal margin hyperplane, which is the hyperplane that is the maximum distance from the training observations. This distance is called the margin. Points that fall on one side of the hyperplane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

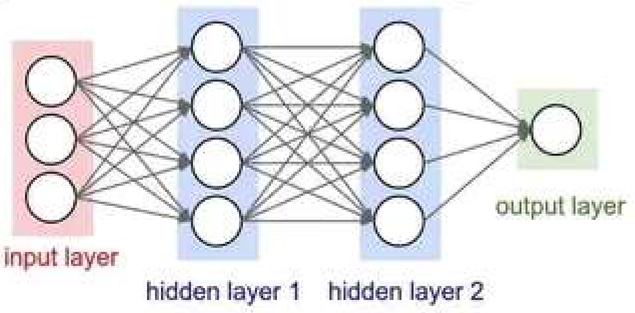
ML Terminology and Concepts

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

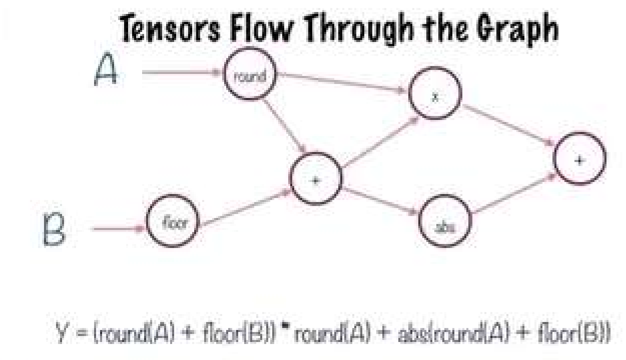
There are different kinds of NNs that are suitable for certain problems, which depend on the NN's architecture.

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 Flow Through the Graph
A
B
round
floor
+
x
abs
+
Y = (round(A) + floor(B)) * round(A) + abs(round(A) + floor(B))

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

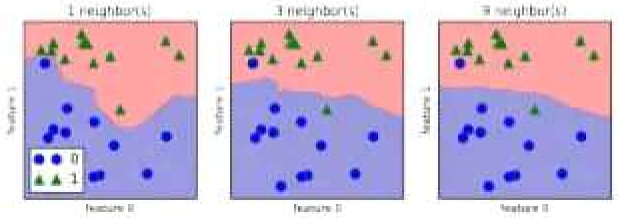
Variables: best way to represent shared, persistent state manipulated by your program. These are the parameters of the ML model are altered/trained during the training process. Training variables.
Placeholders: way to specify inputs into a graph that hold the place for a Tensor that will be fed at runtime.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

Nearest Neighbor Classification

Distance functions allow us to identify the points closest to a given target, or the *nearest neighbors (NN)* to a given point. The advantages of NN include simplicity, interpretability and non-linearity.

k-Nearest Neighbors

Given a positive integer k and a point x_0, the KNN classifier first identifies k points in the training data most similar to x_0, then estimates the conditional probability of x_0 being in class j as the fraction of the k points whose values belong to j. The optimal value for k can be found using cross validation.



KNN Algorithm

1. Compute distance D(a,b) from point b to all points
2. Select k closest points and their labels
3. Output class with most frequent labels in k points

Optimizing KNN

Comparing a query point a in d dimensions against n training examples computes with a runtime of $O(nd)$, which can cause lag as points reach millions or billions. Popular choices to speed up KNN include:

- **Vernoi Diagrams**: partitioning plane into regions based on distance to points in a specific subset of the plane
- **Grid Indexes**: carve up space into d-dimensional boxes or grids and calculate the NN in the same cell as the point
- **Locality Sensitive Hashing (LSH)**: abandons the idea of finding the exact nearest neighbors. Instead, batch up nearby points to quickly find the most appropriate bucket B for our query point. LSH

Clustering

Clustering is the problem of grouping points by similarity using distance metrics, which ideally reflect the similarities you are looking for. Often items come from logical "sources" and clustering is a good way to reveal those origins. Perhaps the first thing to do with any data set. Possible applications include: hypothesis development, modeling over smaller subsets of data, data reduction, outlier detection.

K-Means Clustering

Simple and elegant algorithm to partition a dataset into K distinct, non-overlapping clusters.

1. Choose a K. Randomly assign a number between 1 and K to each observation. These serve as initial cluster assignments
2. Iterate until cluster assignments stop changing
   (a) For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
   (b) Assign each observation to the cluster whose centroid is closest (where closest is defined using distance metric).

Since the results of the algorithm depends on the initial random assignments, it is a good idea to repeat the algorithm from different random initializations to obtain the best overall results. Can use MSE to determine which cluster assignment is better.

Hierarchical Clustering

Alternative clustering algorithm that does not require us to commit to a particular K. Another advantage is that it results in a nice visualization called a **dendrogram**. Observations that fuse at bottom are similar, where those at the top are quite different- we draw conclusions based on the location on the vertical rather than horizontal axis.

1. Begin with n observations and a measure of all the $\frac{(n)n-1}{2}$ pairwise dissimilarities. Treat each observation as its own cluster.