Machine Learning Quick Guide

2022

Distributions

Discrete

Binomial - x successes in n events, each with p probability $\rightarrow \binom{n}{x} p^x q^{n-x}$, with $\mu = np$ and $\sigma^2 = npq$
- If n = 1, this is a Bernoulli distribution

Geometric - first success with p probability on the n^{th} trial $\rightarrow q^{n-1}p$, with $\mu = 1/p$ and $\sigma^2 = \frac{1-p}{p^2}$

Negative Binomial - number of failures before r successes

Hypergeometric - x successes in n draws, no replacement, from a size N population with X items of that feature $\rightarrow \frac{\binom{X}{x}\binom{N-X}{n-x}}{\binom{N}{n}}$, with $\mu = \frac{nX}{N}$

Poisson - number of successes x in a fixed time interval, where success occurs at an average rate $\lambda \rightarrow \frac{\lambda^x e^{-\lambda}}{x!}$, with $\mu = \sigma^2 = \lambda$

Continuous

Uniform - all values between a and b are equally likely $\rightarrow \frac{1}{b-a}$ with $\mu = \frac{a+b}{2}$ and $\sigma^2 = \frac{(b-a)^2}{12}$ or $\frac{n^2-1}{12}$ if discrete

Normal/Gaussian $N(\mu, \sigma)$, Standard Normal $Z \sim N(0,1)$
- Central Limit Theorem - sample mean of i.i.d. data approaches normal distribution
- Empirical Rule - 68%, 95%, and 99.7% of values lie within one, two, and three standard deviations of the mean
- Normal Approximation - discrete distributions such as Binomial and Poisson can be approximated using z-scores when np, nq, and λ are greater than 10

Exponential - memoryless time between independent events occurring at an average rate $\lambda \rightarrow \lambda e^{-\lambda x}$, with $\mu = \frac{1}{\lambda}$

Gamma - time until n independent events occurring at an average rate λ

Concepts

Prediction Error = Bias^2 + Variance + Irreducible Noise

Bias - wrong assumptions when training $\rightarrow$ can't capture underlying patterns $\rightarrow$ underfit

Variance - sensitive to fluctuations when training$\rightarrow$ can't generalize on unseen data $\rightarrow$ overfit

The bias-variance tradeoff attempts to minimize these two sources of error, through methods such as:
- Cross validation to generalize to unseen data
- Dimension reduction and feature selection

In all cases, as variance decreases, bias increases.

ML models can be divided into two types:
- Parametric - uses a fixed number of parameters with respect to sample size
- Non-Parametric - uses a flexible number of parameters and doesn't make particular assumptions on the data

Cross Validation - validates test error with a subset of training data, and selects parameters to maximize average performance
- k-fold - divide data into k groups, and use one to validate
- leave-p-out - use p samples to validate and the rest to train

Model Evaluation

Regression

Mean Squared Error (MSE) $= \frac{1}{n}\sum(y_i - \hat{y})^2$

Sum of Squared Error (SSE) $= \sum(y_i - \hat{y})^2$

Total Sum of Squares (SST) $= \sum(y_i - \bar{y})^2$

$\boldsymbol{R^2} = 1 - \frac{SSE}{SST}$, the proportion of explained y-variability

Note, negative R^2 means the model is worse than just predicting the mean. R^2 is not valid for nonlinear models, as $SS_{residual} + SS_{error} \neq SST$.

Adjusted $\boldsymbol{R^2} = 1 - (1 - R^2)\frac{N-1}{N-p-1}$, which changes only when predictors affect R^2 above what would be expected by chance

Classification

	Predict Yes	Predict No
Actual Yes	True Positive $(1-\beta)$	False Negative (β)
Actual No	False Positive (α)	True Negative $(1-\alpha)$

- Precision $= \frac{TP}{TP+FP}$, percent correct when predict positive
- Recall, Sensitivity $= \frac{TP}{TP+FN}$, percent of actual positives identified correctly (True Positive Rate)
- Specificity $= \frac{TN}{TN+FP}$, percent of actual negatives identified correctly, also 1 - FPR (True Negative Rate)
- $F_1 = 2\frac{precision \cdot recall}{precision+recall}$, useful when classes are imbalanced

ROC Curve - plots TPR vs. FPR for every threshold α. Area Under the Curve measures how likely the model differentiates positives and negatives (perfect AUC = 1, baseline = 0.5).

Precision-Recall Curve - focuses on the correct prediction of the minority class, useful when data is imbalanced

Linear Regression

Models linear relationships between a continuous response and explanatory variables

Ordinary Least Squares - find $\hat{\beta}$ for $\hat{y} = \hat{\beta}_0 + \hat{\beta}X + \epsilon$ by solving $\hat{\beta} = (X^T X)^{-1} X^T Y$ which minimizes the SSE

Assumptions
- Linear relationship and independent observations
- Homoscedasticity - error terms have constant variance
- Errors are uncorrelated and normally distributed
- Low multicollinearity

Variance Inflation Factor - measures the severity of multicollinearity $\rightarrow \frac{1}{1-R_i{}^2}$, where $R_i{}^2$ is found by regressing X_i against all other variables (a common VIF cutoff is 10)

Regularization

Add a penalty λ for large coefficients to the cost function, which reduces overfitting. Requires normalized data.

Subset (L_0): $\lambda||\hat{\beta}||_0 = \lambda(number\ of\ non{-}zero\ variables)$
- Computationally slow, need to fit 2^k models
- Alternatives: forward and backward stepwise selection

LASSO (L_1): $\lambda||\hat{\beta}||_1 = \lambda\sum|\hat{\beta}|$
- Shrinks coefficients to zero, and is robust to outliers

Ridge (L_2): $\lambda||\hat{\beta}||_2 = \lambda\sum(\hat{\beta})^2$
- Reduces effects of multicollinearity

Combining LASSO and Ridge gives Elastic Net

Logistic Regression

Predicts probability that y belongs to a binary class. Estimates β through maximum likelihood estimation (MLE) by fitting a logistic (sigmoid) function to the data. This is equivalent to minimizing the cross entropy loss. Regularization can be added in the exponent.

$$P(Y=1) = \frac{1}{1+e^{-(\beta_0+\beta x)}}$$

The threshold a classifies predictions as either 1 or 0

Assumptions
- Linear relationship between X and log-odds of Y
- Independent observations
- Low multicollinearity

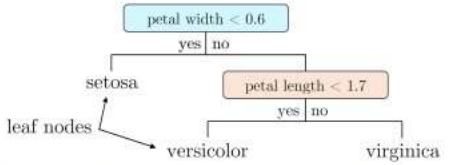
Odds - output probability can be transformed using $Odds(Y=1) = \frac{P(Y=1)}{1-P(Y=1)}$, where $P(\frac{1}{3}) = 1{:}2$ odds

Coefficients are linearly related to odds, such that a one unit increase in x_1 affects odds by e^{β_1}

Decision Trees

Classification and Regression Tree

CART for regression minimizes SSE by splitting data into sub-regions and predicting the average value at leaf nodes. The complexity parameter cp only keeps splits that reduce loss by at least cp (small $cp \rightarrow$ deep tree)



CART for classification minimizes the sum of region impurity, where $\hat{p_i}$ is the probability of a sample being in category i. Possible measures, each with a max impurity of 0.5.
- Gini Impurity $= 1 - \sum(\hat{p_i})^2$
- Cross Entropy $= -\sum(\hat{p_i})log_2(\hat{p_i})$

At each leaf node, CART predicts the most frequent category, assuming false negative and false positive costs are the same. The splitting process handles multicollinearity and outliers. Trees are prone to high variance, so tune through CV.

Random Forest

Trains an ensemble of trees that vote for the final prediction

Bootstrapping - sampling with replacement (will contain duplicates), until the sample is as large as the training set

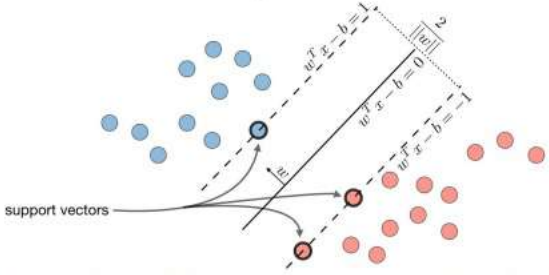
Bagging - training independent models on different subsets of the data, which reduces variance. Each tree is trained on ~63% of the data, so the out-of-bag 37% can estimate prediction error without resorting to CV.

Deep trees may overfit, but adding more trees does not cause overfitting. Model bias is always equal to one of its individual trees.

Variable Importance - ranks variables by their ability to minimize error when split upon, averaged across all trees

Support Vector Machines

Separates data between two classes by maximizing the margin between the hyperplane and the nearest data points of any class. Relies on the following:

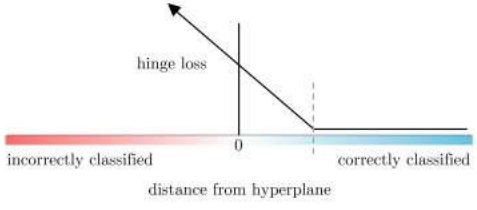


Support Vector Classifiers - account for outliers through the regularization parameter C, which penalizes misclassifications in the margin by a factor of $C > 0$

Kernel Functions - solve nonlinear problems by computing the similarity between points a, b and mapping the data to a higher dimension. Common functions:

- Polynomial $(ab + r)^d$
- Radial $e^{-\gamma(a-b)^2}$, where smaller $\gamma \rightarrow$ smoother boundaries

Hinge Loss - $\max(0, 1 - y_i(w^T x_i - b))$, where w is the margin width, b is the offset bias, and classes are labeled ± 1. Acts as the cost function for SVM. Note, even a correct prediction inside the margin gives loss > 0.



Multiclass Prediction

To classify data with 3+ classes C, a common method is to binarize the problem through:

- One vs. Rest - train a classifier for each class c_i by setting c_i's samples as 1 and all others as 0, and predict the class with the highest confidence score
- One vs. One - train $\frac{C(C-1)}{2}$ models for each pair of classes, and predict the class with the highest number of positive predictions

k-Nearest Neighbors

Non-parametric method that calculates $\hat{y}$ using the average value or most common class of its k-nearest points. For high-dimensional data, information is lost through equidistant vectors, so dimension reduction is often applied prior to k-NN.

Minkowski Distance $= (\sum |a_i - b_i|^p)^{1/p}$

- p = 1 gives Manhattan distance $\sum |a_i - b_i|$
- p = 2 gives Euclidean distance $\sqrt{\sum (a_i - b_i)^2}$

Hamming Distance - count of the differences between two vectors, often used to compare categorical variables

Clustering

Unsupervised, non-parametric methods that groups similar data points together based on distance

k-Means

Randomly place k centroids across normalized data, and assig observations to the nearest centroid. Recalculate centroids as the mean of assignments and repeat until convergence. Using the median or medoid (actual data point) may be more robust to noise and outliers. k-modes is used for categorical data.

***k*-means++** - improves selection of initial clusters

1. Pick the first center randomly
2. Compute distance between points and the nearest center
3. Choose new center using a weighted probability distribution proportional to distance
4. Repeat until k centers are chosen

Evaluating the number of clusters and performance:

Silhouette Value - measures how similar a data point is to its own cluster compared to other clusters, and ranges from 1 (best) to -1 (worst).

Davies-Bouldin Index - ratio of within cluster scatter to between cluster separation, where lower values are better

Hierarchical Clustering

Clusters data into groups using a predominant hierarchy

Agglomerative Approach

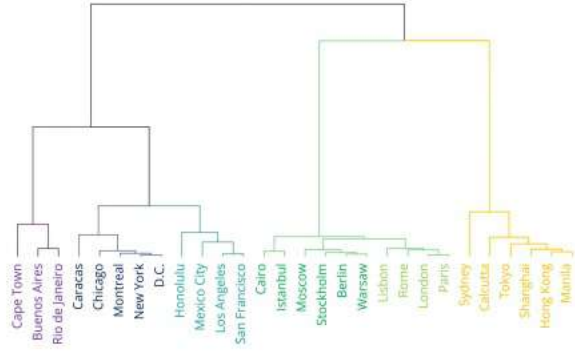
1. Each observation starts in its own cluster
2. Iteratively combine the most similar cluster pairs
3. Continue until all points are in the same cluster

Divisive Approach - all points start in one cluster and splits are performed recursively down the hierarchy

Linkage Metrics - measure dissimilarity between clusters and combines them using the minimum linkage value over all pairwise points in different clusters by comparing:

- Single - the distance between the closest pair of points
- Complete - the distance between the farthest pair of points
- Ward's - the increase in within-cluster SSE if two clusters were to be combined

Dendrogram - plots the full hierarchy of clusters, where the height of a node indicates the dissimilarity between its children



Dimension Reduction

High-dimensional data can lead to the *curse of dimensionality*, which increases the risk of overfitting and decreases the value added. The number of samples for each feature combination quickly becomes sparse, reducing model performance.

Principal Component Analysis

Projects data onto orthogonal vectors that maximize variance. Remember, given an $n \times n$ matrix A, a nonzero vector $\vec{x}$, and a scaler λ, if $A\vec{x} = \lambda\vec{x}$ then $\vec{x}$ and λ are an eigenvector and eigenvalue of A. In PCA, the eigenvectors are uncorrelated and represent principal components.

1. Start with the covariance matrix of standardized data
2. Calculate eigenvalues and eigenvectors using SVD or eigendecomposition
3. Rank the principal components by their proportion of variance explained $= \frac{\lambda_i}{\sum \lambda}$

Data should be linearly related, and for a p-dimensional dataset, there will be p principal components.

Note, PCA explains the variance in X, not necessarily Y.

Sparse PCA - constrains the number of non-zero values in each component, reducing susceptibility to noise and improving interpretability

Linear Discriminant Analysis

Supervised method that maximizes separation between classes and minimizes variance within classes for a labeled dataset

1. Compute the mean and variance of each independent variable for every class C_i
2. Calculate the within-class (σ_w^2) and between-class (σ_b^2) variance
3. Find the matrix $W = (\sigma_w^2)^{-1}(\sigma_b^2)$ that maximizes Fisher's signal-to-noise ratio
4. Rank the discriminant components by their signal-to-noise ratio λ

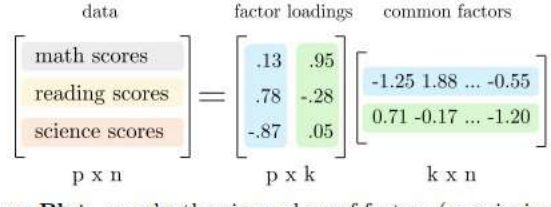
Note, the number of components is at most $C_1 - 1$

Assumptions

- Independent variables are normally distributed
- Homoscedasticity - constant variance of error
- Low multicollinearity

Factor Analysis

Describes data using a linear combination of k latent factors. Given a normalized matrix X, it follows the form $X = Lf + \epsilon$, with factor loadings L and hidden factors f.



Scree Plot - graphs the eigenvalues of factors (or principal components) and is used to determine the number of factors to retain. The 'elbow' where values level off is often used as the cutoff.

Boosting

Sequentially fits many simple models that account for the previous model's errors. As opposed to bagging, boosting trains on all the data and combines models using the learning rate α.

AdaBoost - uses sample weighting and decision 'stumps' (one-level decision trees) to classify samples

1. Build decision stumps for every feature, choosing the one with the best classification accuracy
2. Assign more weight to misclassified samples and reward trees that differentiate them, where $\alpha = \frac{1}{2} ln \frac{1-TotalError}{TotalError}$
3. Continue training and weighting decision stumps until convergence

Gradient Boost - trains sequential models by minimizing a given loss function using gradient descent at each step

1. Start by predicting the average value of the response
2. Build a tree on the errors, constrained by depth or the number of leaf nodes
3. Scale decision trees by a constant learning rate α
4. Continue training and weighting decision trees until convergence

XGBoost - fast gradient boosting method that utilizes regularization and parallelization

Recommender Systems

Suggests relevant items to users by predicting ratings and preferences, and is divided into two main types:

- Content Filtering - recommends similar items
- Collaborative Filtering - recommends what similar users like

The latter is more common, and includes methods such as:
Memory-based Approaches - finds neighborhoods by using rating data to compute user and item similarity, measured using correlation or cosine similarity

- User-User - similar users also liked...
  - Leads to more diverse recommendations, as opposed to just recommending popular items
  - Suffers from sparsity, as the number of users who rate items is often low
- Item-Item - similar users who liked this item also liked...
  - Efficient when there are more users than items, since the item neighborhoods update less frequently than users
  - Similarity between items is often more reliable than similarity between users

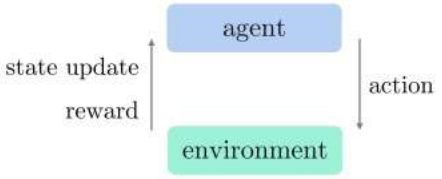
Model-based Approaches - predict ratings of unrated items, through methods such as Bayesian networks, SVD, and clustering. Handles sparse data better than memory-based approaches.

- Matrix Factorization - decomposes the user-item rating matrix into two lower-dimensional matrices representing the users and items, each with k latent factors

Recommender systems can also be combined through ensemble methods to improve performance.

Reinforcement Learning

Maximizes future rewards by learning through state-action pairs. That is, an *agent* performs *actions* in an *environment*, which updates the *state* and provides a *reward*.



Multi-armed Bandit Problem - a gambler plays slot machines with unknown probability distributions and must decide the best strategy to maximize reward. This exemplifies the exploration-exploitation tradeoff, as the best long-term strategy may involve short-term sacrifices.

RL is divided into two types, with the former being more common:

- Model-free - learn through trial and error in the environment
- Model-based - access to the underlying (approximate) state-reward distribution

Q-Value $Q(s, a)$ - captures the expected discounted total future reward given a state and action
Policy - chooses the best actions for an agent at various states $\pi(s) = \arg\max_a Q(s, a)$

Deep RL algorithms can further be divided into two main types, depending on their learning objective
Value Learning - aims to approximate $Q(s, a)$ for all actions the agent can take, but is restricted to discrete action spaces. Can use the ϵ-greedy method, where ϵ measures the probability of exploration. If chosen, the next action is selected uniformly at random.

- Q-Learning - simple value iteration model that maximizes the Q-value using a table on states and actions
- Deep Q Network - finds the best action to take by minimizing the Q-loss, the squared error between the target Q-value and the prediction

Policy Gradient Learning - directly optimize the the policy $\pi(s)$ through a probability distribution of actions, without the need for a value function, allowing for continuous action spaces.

Actor-Critic Model - hybrid algorithm that relies on two neural networks, an actor $\pi(s, a, \theta)$ which controls agent behavior and a critic $Q(s, a, w)$ that measures how good an action is. Both run in parallel to find the optimal weights θ, w to maximize expected reward. At each step:

1. Pass the current state into the actor and critic
2. The critic evaluates the action's Q-value, and the actor updates its weight θ
3. The actor takes the next action leading to a new state, and the critic updates its weight w

Anomaly Detection

Identifies unusual patterns that differ from the majority of the data. Assumes that anomalies are:

- Rare - the minority class that occurs rarely in the data
- Different - have feature values that are very different from normal observations

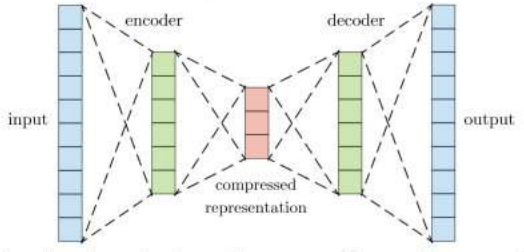
Anomaly detection techniques spans a wide range, including methods based on:
Statistics - relies on various statistical methods to identify outliers, such as Z-tests, boxplots, interquartile ranges, and variance comparisons
Density - useful when data is grouped around dense neighborhoods, measured by distance. Methods include k-nearest neighbors, local outlier factor, and isolation forest.

- Isolation Forest - tree-based model that labels outliers based on an anomaly score
  1. Select a random feature and split value, dividing the dataset in two
  2. Continue splitting randomly until every point is isolated
  3. Calculate the anomaly score for each observation, based on how many iterations it took to isolate that point.
  4. If the anomaly score is greater than a threshold, mark it as an outlier

  Intuitively, outliers are easier to isolate and should have shorter path lengths in the tree

Clusters - data points outside of clusters could potentially be marked as anomalies

Autoencoders - unsupervised neural networks that compress data through an encoder and reconstruct it using a decoder. Autoencoders do not reconstruct the data perfectly, but rather focus on capturing important features in the data.



The decoder struggles to capture anomalous patterns, and the reconstruction error acts as a score to detect anomalies.

Autoencoders can also be used for image processing, dimension reduction, and information retrieval.

Hidden Markov Model - uses observed events O to model a set of n underlying states Q using $\lambda = (A, B, \pi)$

- A - $n \times n$ matrix of transition probabilities from state i to j
- B - sequence of likelihoods of emitting o_t in state i
- π - initial probability distribution over states

HMMs can calculate $P(O|\lambda)$, find the best hidden state sequence Q, or learn the parameters A and B. Anomalies are observations that are unlikely to occur across states.

HMMs can be applied to many problems such as signal processing and part of speech tagging.