Data Science Cheatsheet 2.0

Last Updated June 19, 2021

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

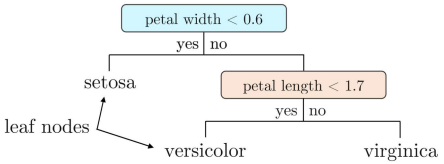
Odds - output probability can be transformed using $Odds(Y = 1) = \frac{P(Y=1)}{1-P(Y=1)}$, where $P(\frac{1}{3}) = 1{:}2$ odds

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 $\sim$63% of the data, so the out-of-bag 37% can estimate prediction error without resorting to CV.

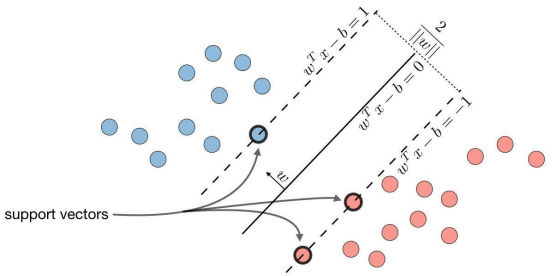
Deep trees may overfit, but adding more trees does not cause overfitting. Model bias is always equal to one of its individual trees.

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

Aaron Wang

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:

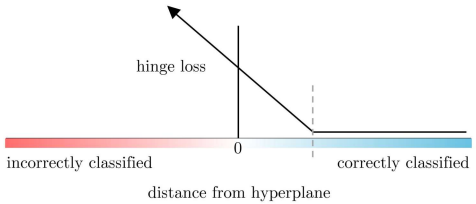


Support Vector Classifiers - account for outliers through the regularization parameter C, which penalizes misclassifications in the margin by a factor of $C > 0$

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

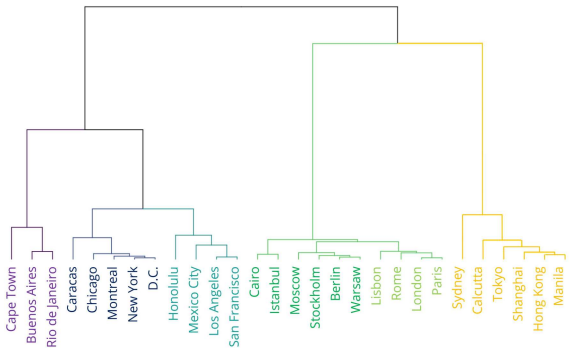
1. Each observation starts in its own cluster
2. Iteratively combine the most similar cluster pairs
3. Continue until all points are in the same cluster

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

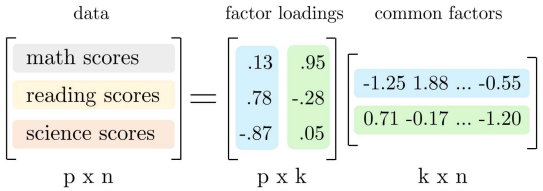
Note, the number of components is at most $C_1 - 1$

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.

Aaron Wang

Natural Language Processing

Transforms human language into machine-usable code
Processing Techniques

- Tokenization - splits text into individual words (tokens)
- Lemmatization - reduces words to its base form based on dictionary definition (*am, are, is* $\rightarrow$ *be*)
- Stemming - reduces words to its base form without context (*ended* $\rightarrow$ *end*)
- Stop words - removes common and irrelevant words (*the, is*)

Markov Chain - stochastic and memoryless process that predicts future events based only on the current state
***n*-gram** - predicts the next term in a sequence of n terms based on Markov chains
Bag-of-words - represents text using word frequencies, without context or order
tf-idf - measures word importance for a document in a collection (corpus), by multiplying the term frequency (occurrences of a term in a document) with the inverse document frequency (penalizes common terms across a corpus)
Cosine Similarity - measures similarity between vectors, calculated as $\cos(\theta) = \frac{A \cdot B}{||A|| ||B||}$, which ranges from 0 to 1

Word Embedding

Maps words and phrases to numerical vectors
word2vec - trains iteratively over local word context windows, places similar words close together, and embeds sub-relationships directly into vectors, such that $king - man + woman \approx queen$
Relies on one of the following:

- Continuous bag-of-words (CBOW) - predicts the word given its context
- skip-gram - predicts the context given a word

GloVe - combines both global and local word co-occurence data to learn word similarity
BERT - accounts for word order and trains on subwords, and unlike word2vec and GloVe, BERT outputs different vectors for different uses of words (*cell* phone vs. blood *cell*)

Sentiment Analysis

Extracts the attitudes and emotions from text
Polarity - measures positive, negative, or neutral opinions

- Valence shifters - capture amplifiers or negators such as '*really* fun' or '*hardly* fun'

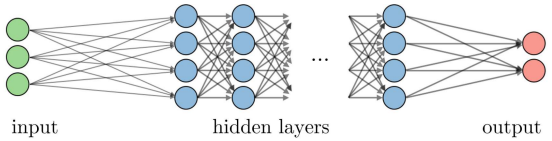
Sentiment - measures emotional states such as happy or sad
Subject-Object Identification - classifies sentences as either subjective or objective

Topic Modelling

Captures the underlying themes that appear in documents
Latent Dirichlet Allocation (LDA) - generates k topics by first assigning each word to a random topic, then iteratively updating assignments based on parameters α, the mix of topics per document, and β, the distribution of words per topic
Latent Semantic Analysis (LSA) - identifies patterns using tf-idf scores and reduces data to k dimensions through SVD

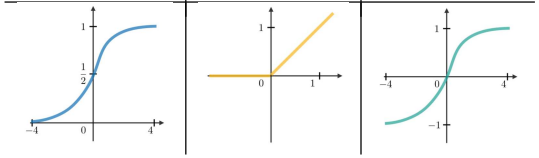
Neural Network

Feeds inputs through different hidden layers and relies on weights and nonlinear functions to reach an output



Perceptron - the foundation of a neural network that multiplies inputs by weights, adds bias, and feeds the result z to an activation function
Activation Function - defines a node's output

Sigmoid	ReLU	Tanh
$\frac{1}{1+e^{-z}}$	$\max(0, z)$	$\frac{e^z - e^{-z}}{e^z + e^{-z}}$



Softmax - given final layer outputs, provides class probabilities that sum to 1 $\rightarrow \frac{e^z i}{\sum e^z}$
If there is more than one 'correct' label, the sigmoid function provides probabilities for all, some, or none of the labels.

Loss Function - measures prediction error using functions such as MSE for regression and binary cross-entropy for probability-based classification

Gradient Descent - minimizes the average loss by moving iteratively in the direction of steepest descent, controlled by the learning rate γ (step size). Note, γ can be updated adaptively for better performance. For neural networks, finding the best set of weights involves:

1. Initialize weights W randomly with near-zero values
2. Loop until convergence:
   - Calculate the average network loss $J(W)$
   - **Backpropagation** - iterate backwards from the last layer, computing the gradient $\frac{\partial J(W)}{\partial W}$ and updating the weight $W \leftarrow W - \gamma \frac{\partial J(W)}{\partial W}$
3. Return the minimum loss weight matrix W

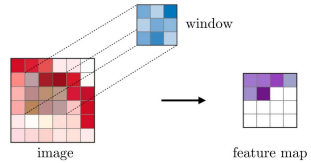
To prevent overfitting, regularization can be applied by:

- Stopping training when validation performance drops
- Dropout - randomly drop some nodes during training to prevent over-reliance on a single node
- Embedding weight penalties into the objective function
- Batch Normalization - stabilizes learning by normalizing inputs to a layer

Stochastic Gradient Descent - only uses a single point to compute gradients, leading to smoother convergence and faster compute speeds. Alternatively, mini-batch gradient descent trains on small subsets of the data, striking a balance between the approaches.

Convolutional Neural Network

Analyzes structural or visual data by extracting local features
Convolutional Layers - iterate over windows of the image, applying weights, bias, and an activation function to create feature maps. Different weights lead to different features maps.



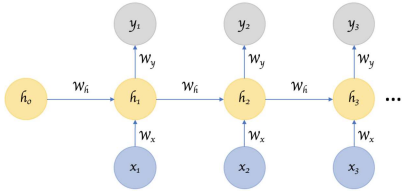
Pooling - downsamples convolution layers to reduce dimensionality and maintain spatial invariance, allowing detection of features even if they have shifted slightly. Common techniques return the max or average value in the pooling window.

The general CNN architecture is as follows:

1. Perform a series of convolution, ReLU, and pooling operations, extracting important features from the data
2. Feed output into a fully-connected layer for classification, object detection, or other structural analyses

Recurrent Neural Network

Predicts sequential data using a temporally connected system that captures both new inputs and previous outputs using hidden states



RNNs can model various input-output scenarios, such as many-to-one, one-to-many, and many-to-many. Relies on parameter (weight) sharing for efficiency. To avoid redundant calculations during backpropagation, downstream gradients are found by chaining previous gradients. However, repeatedly multiplying values greater than or less than 1 leads to:

- Exploding gradients - model instability and overflows
- Vanishing gradients - loss of learning ability

This can be solved using:

- Gradient clipping - cap the maximum value of gradients
- ReLU - its derivative prevents gradient shrinkage for $x > 0$
- Gated cells - regulate the flow of information

Long Short-Term Memory - learns long-term dependencies using gated cells and maintains a separate cell state from what is outputted. Gates in LSTM perform the following:

1. Forget and filter out irrelevant info from previous layers
2. Store relevant info from current input
3. Update the current cell state
4. Output the hidden state, a filtered version of the cell state

LSTMs can be stacked to improve performance.

Aaron Wang