Data Science Cheatsheet 2.0

Last Updated October 26, 2025

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 → can't capture underlying patterns → underfit

Variance - sensitive to fluctuations when training→ can't generalize on unseen data → 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

Let: y_i a label in dataset $\hat{y}$ a prediction $\bar{y}$ mean of labels

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 No	Predict Yes
True No	True Negative $(1-\alpha)$ (TN)	False Positive (α) (FP)
True Yes	False Negative (β) (FN)	True Positive $(1-\beta)$ (TP)

- Accuracy $= \frac{TP+TN}{TP+FP+FP+FN}$, percent correct. Not to use when having imbalanced dataset.
- 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

Ranking / Retrieval

Setup. For each query $q \in Q$, predicted ranking $\pi_q = (d_{q,1}, \ldots, d_{q,n_q})$; relevance $\mathrm{rel}_q(d) \in \{0, 1, 2, 3, \ldots\}$; $r_{q,i} = \mathrm{rel}_q(d_{q,i})$; relevant set $R_q = \{d : \mathrm{rel}_q(d) > 0\}$. Use indicator $\mathbf{1}\{\cdot\}$.

Precision@k (per q) $= \frac{1}{k}\sum_{i=1}^{\min(k,n_q)} \mathbf{1}\{r_{q,i} > 0\}$ (defined for $k \geq 1$, divide by k even if $n_q < k$).

Mean Precision@k $= \frac{1}{|Q|}\sum_{q \in Q} \mathrm{Precision@}k(q)$.

Average Precision (AP). $\mathrm{P@}i(q) = \frac{1}{i}\sum_{j=1}^{i} \mathbf{1}\{r_{q,j} > 0\}$.

$$\mathrm{AP}(q) = \begin{cases} \frac{1}{|R_q|}\sum_{i=1}^{n_q} \mathrm{P@}i(q)\,\mathbf{1}\{r_{q,i} > 0\}, & |R_q| > 0 \\ 0, & \text{otherwise} \end{cases}.$$

MAP $= \frac{1}{|Q|}\sum_{q \in Q} \mathrm{AP}(q)$.

Recall@k (per q) $= \frac{\sum_{i=1}^{\min(k,n_q)} \mathbf{1}\{r_{q,i} > 0\}}{|R_q|}$, with the convention $\mathrm{Recall@}k(q) = 0$ if $|R_q| = 0$.

Mean Recall@k $= \frac{1}{|Q|}\sum_{q \in Q} \mathrm{Recall@}k(q)$.

Reciprocal Rank (RR). $i_q^\star = \min\{\, i \geq 1 : r_{q,i} > 0 \,\}$ if it exists; $\mathrm{RR}(q) = \begin{cases} 1/i_q^\star, & \text{if exists} \\ 0, & \text{otherwise} \end{cases}.$

MRR $= \frac{1}{|Q|}\sum_{q \in Q} \mathrm{RR}(q)$.

DCG@k. $\mathrm{DCG@}k(q) = \sum_{i=1}^{\min(k,n_q)} \frac{2^{r_{q,i}} - 1}{\log_2(i+1)}$; let $\mathrm{IDCG@}k(q)$ be DCG@k of the ideal ranking.

$$\textbf{NDCG@k} = \begin{cases} \frac{\mathrm{DCG@}k(q)}{\mathrm{IDCG@}k(q)}, & \mathrm{IDCG@}k(q) > 0 \\ 0, & \text{otherwise} \end{cases}.$$

Ex. (@5). Ground truth $G_q = (A, B, C, D, E)$, predicted $\pi_q = (C, Z, A, E, B)$. Grades from G_q: $\mathrm{rel}(A, \ldots, E) = (5, 4, 3, 2, 1)$ (others 0). With gain $2^r - 1$: $\mathrm{DCG@5} \approx 28.73$, $\mathrm{IDCG@5} \approx 45.64 \Rightarrow \mathrm{nDCG@5} \approx 0.630$.

Mean NDCG@k $= \frac{1}{|Q|}\sum_{q \in Q} \mathrm{NDCG@}k(q)$.

Binary case. If $r_{q,i} \in \{0, 1\}$, then $\mathrm{DCG@}k(q) = \sum_{i=1}^{\min(k,n_q)} \frac{\mathbf{1}\{r_{q,i}=1\}}{\log_2(i+1)}$.

Linear Regression

Models linear relationships between a continuous response and explanatory variables

Normal equation : $X^T(Y - X\Theta) = 0$

Ordinary Least Squares - find $\hat{\beta}$ for $\hat{y} = \hat{\beta_0} + \hat{\beta}X + \epsilon$ by solving $\hat{\beta} = (X^TX)^{-1}X^TY$ 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

CART (Clas. & Regr.) w/ Iris-style example

Data. $\mathcal{D} = \{(pw_i, pl_i, y_i)\}_{i=1}^{12}$ with classes $y \in \{\text{setosa}, \text{versicolor}, \text{virginica}\}$. Toy counts: 4 setosa (all $pw < 0.6$), 4 versicolor ($pw \geq 0.6$, $pl < 1.7$), 4 virginica ($pw \geq 0.6$, $pl \geq 1.7$).

Impurity & gain. For node t with class probs $p_{t,c}$: $G(t) = 1 - \sum_c p_{t,c}^2$, $\Delta I = G(t) - \frac{n_L}{n_t} G(t_L) - \frac{n_R}{n_t} G(t_R)$.

Root (before split). Class mix 4/12 each $\Rightarrow G(\text{root}) = 1 - 3(1/3)^2 = 2/3$.

Best split 1 (numeric, $pw < 0.6$). Left: $n_L = 4$ (all setosa) $\Rightarrow G(L) = 0$; Right: $n_R = 8$ (4 versicolor, 4 virginica) $\Rightarrow G(R) = 1 - 2(1/2)^2 = 0.5$.

$$\Delta I_{\text{root}} = \tfrac{2}{3} - \tfrac{4}{12} \cdot 0 - \tfrac{8}{12} \cdot 0.5 = \tfrac{2}{3} - \tfrac{1}{3} = \boxed{\tfrac{1}{3}}.$$

Best split 2 (right child, $pl < 1.7$). Left: $n = 4$ (pure versicolor) $G = 0$; Right: $n = 4$ (pure virginica) $G = 0$.

$$\Delta I_R = 0.5 - 1 \cdot 0 = \boxed{0.5}.$$

Training rule (greedy). For each node: sort thresholds per feature; evaluate ΔI via one pass (prefix counts); split if $\Delta I \geq cp$; recurse; prune by $R_\alpha(T) = \sum_\ell G(\ell) + \alpha|\mathcal{L}(T)|$.

Picking the best ΔI. *Not exponential.* Numeric features: sort distinct values once; test only cuts between consecutive values. Let ordered targets be $y_{(1)}, \ldots, y_{(n)}$.

$$S_k = \sum_{i \leq k} y_{(i)},\ p_L = S_k/k,\ G_L = 1 - p_L^2 - (1-p_L)^2,$$

$$p_R = (S_n - S_k)/(n-k),\ G_R = 1 - p_R^2 - (1-p_R)^2.$$

$$T_k = \sum_{i \leq k} y_{(i)},\ Q_k = \sum_{i \leq k} y_{(i)}^2,$$

$$\text{SSE}_L = Q_k - T_k^2/k,\ \text{SSE}_R = (Q_n - Q_k) - (T_n - T_k)^2/(n-k).$$

Scan $k = 1, \ldots, n-1$; take max ΔI. Cost per feature: sort $O(n \log n)$ (or $O(n)$ with presorted indices) + linear scan $O(n)$.

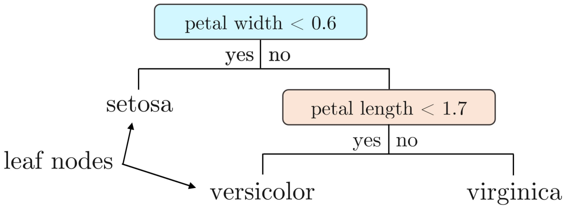
Categoricals (K levels): *Binary y:* compute $\hat{p}(a) = P(Y=1 \mid A=a)$, sort levels by $\hat{p}(a)$, test $K-1$ contiguous cuts (reduces to numeric). *Multi-class:* order by one-vs-rest or argmax class rate (or target-encode with CV); test $K-1$ cuts. Hence $O(K \log K + K)$, not $2^{K-1}-1$.

Large scale (GBDT/RF): histogram/quantile binning with B bins $\Rightarrow$ per node $O(n + dB)$; feature subsampling and min-gain/min-samples further cap work.

Final tree (matches figure).

$$\boxed{pw < 0.6} \Rightarrow \text{setosa}; \quad pw \geq 0.6\ \&\ \boxed{pl < 1.7} \Rightarrow \text{versicolor; else virginica.}$$

Inference (examples). $(pw{=}0.55, pl{=}1.3) \Rightarrow$ setosa; $(pw{=}0.8, pl{=}1.5) \Rightarrow$ versicolor; $(pw{=}1.2, pl{=}2.0) \Rightarrow$ virginica.



Random Forest

Trains an ensemble of trees with a subset of **features** (columns) with replacement that vote for the final prediction

$$f_{ens} = \frac{1}{T} \sum_t f_t(x)$$

Bootstrapping : **sampling** (rows) 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

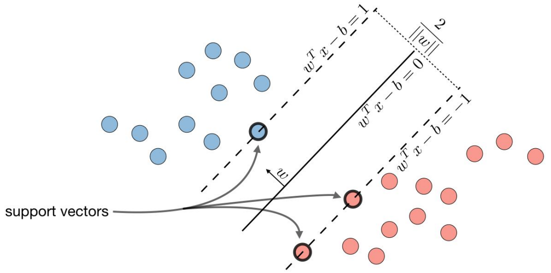
Separates data between two classes by maximizing the margin between the hyperplane and the nearest data points of any class. Relies on the following:

Matematically,it solves the problem:

$$\min \frac{1}{2}||w||^2 + C \sum_i \xi_i$$

s.t

$$y_i(wx_i + b) \geq 1 - \xi_i, \forall x_i, \xi_i \geq 0$$

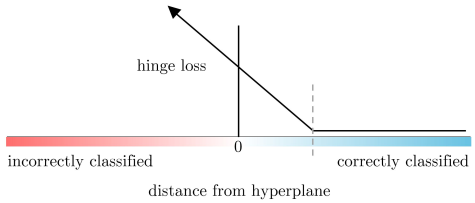


Support Vector Classifiers - account for outliers through the regularization parameter C, which penalizes misclassifications in the margin by a factor of $C > 0$

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

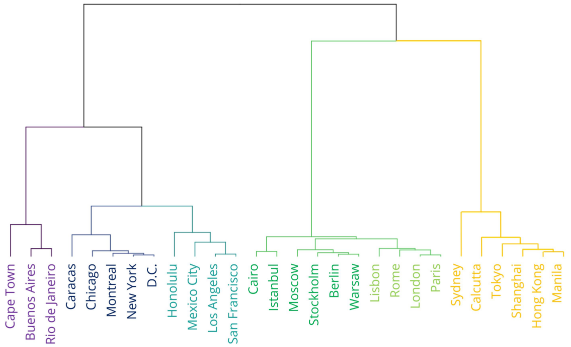
Clusters data into groups using a predominant hierarchy
Agglomerative Approach

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

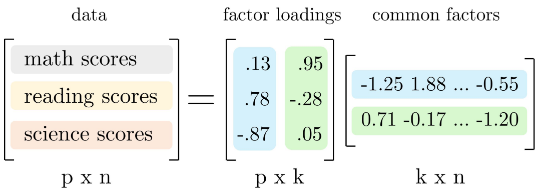
1. Compute the mean and variance of each independent variable for every class C_i
2. Calculate the within-class (σ_w^2) and between-class (σ_b^2) variance
3. Find the matrix $W = (\sigma_w^2)^{-1}(\sigma_b^2)$ that maximizes Fisher's signal-to-noise ratio
4. Rank the discriminant components by their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


data

factor loadings

common factors

math scores

reading scores

science scores

=

.13 .95

.78 -.28

-.87 .05

-1.25 1.88 ... -0.55

0.71 -0.17 ... -1.20

p x n

p x k

k x n

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.

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* → *be*)
- Stemming - reduces words to its base form without context (*ended* → *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 o to 1

Setup/Notation. Corpus $\mathcal{D} = \{d_i\}_{i=1}^N$; vocabulary $\mathcal{V}$ (after tokenization/normalization). Each document $d = (w_1, \ldots, w_{|d|})$ with tokens $w_j \in \mathcal{V}$. Query $q = (u_1, \ldots, u_{|q|})$, $u_k \in \mathcal{V}$. Average length $\overline{|d|} = \frac{1}{N}\sum_{i=1}^{N} |d_i|$.

For a term $t \in \mathcal{V}$:

$$tf_{t,d} = \sum_{j=1}^{|d|} \mathbf{1}\{w_j = t\} \qquad \text{(TF: term frequency)}$$

$$df_t = \sum_{i=1}^{N} \mathbf{1}\{\exists j \le |d_i| : w_j^{(i)} = t\} \qquad \text{(DF: document frequency)}$$

$$qtf_t = \sum_{k=1}^{|q|} \mathbf{1}\{u_k = t\} \qquad \text{(QTF: query term frequency)}$$

Note $tf_{t,d} \in \mathbb{N}$, $df_t \in \{0, \ldots, N\}$, $qtf_t \in \{0, \ldots, |q|\}$ and $\sum_{t \in \mathcal{V}} tf_{t,d} = |d|$.
BM25 (Okapi).

$$s(d,q) = \sum_{t \in q} \mathrm{IDF}(t) \, \frac{(k_1 + 1)\, tf_{t,d}}{tf_{t,d} + k_1\left(1 - b + b\,\frac{|d|}{\overline{|d|}}\right)} \cdot \frac{(k_3 + 1)\, qtf_t}{k_3 + qtf_t} \,.$$

Commonly $k_3 \to \infty \Rightarrow \frac{(k_3+1)\, qtf_t}{k_3 + qtf_t} \approx 1$.
IDF (inverse document frequency)(RSJ).

$$\mathrm{IDF}(t) = \log \frac{N - df_t + \frac{1}{2}}{df_t + \frac{1}{2}} \,. \quad \text{Smoothed: } \log\left(1 + \frac{N - df_t + \frac{1}{2}}{df_t + \frac{1}{2}}\right).$$

Terms & params. $k_1 > 0$ (TF saturation, e.g. 1.2–2.0), $b \in [0,1]$ (length norm, e.g. 0.5–0.9), k_3 (query TF).
Intuition. TF saturation: $\frac{(k_1+1)tf}{tf + k_1(\cdot)}$ grows sublinearly; length norm $1 - b + b\frac{|d|}{\overline{|d|}}$ downweights long docs; IDF boosts rare terms.

BM25+ (offset).

$$s_+(d,q) = \sum_{t\in q} \mathrm{IDF}(t)\, \frac{(k_1+1)\,(tf_{t,d}+\delta)}{tf_{t,d}+k_1\left(1-b+b\,\frac{|d|}{\overline{|d|}}\right)} \text{ with } \delta{>}0.$$

BM25F (multi-field). $tf^*_{t,d} =$

$$\sum_f w_f\, \frac{tf_{t,d,f}}{1-b_f+b_f\,\frac{|d_f|}{\overline{|d_f|}}}, \quad s_F(d,q) = \sum_{t\in q} \mathrm{IDF}(t)\, \frac{(k_1+1)tf^*_{t,d}}{k_1+tf^*_{t,d}}.$$

Per-query normalization to $[0,1]$**.** $U_t = \mathrm{IDF}(t)\,(k_1+1)$;

$$s_{\mathrm{norm}}(d,q) = \frac{s(d,q)}{\sum_{t\in q} U_t} \in [0,1] \text{ (clip)}.$$

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* ≈ *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

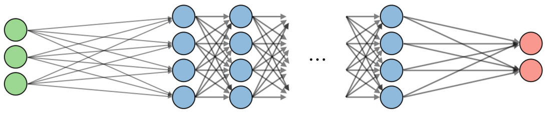
Captures the underlying themes that appear in documents

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

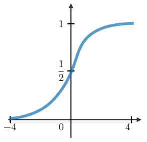
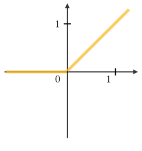
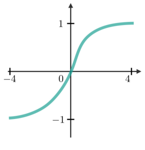
Feeds inputs through different hidden layers and relies on weights and nonlinear functions to reach an output



input hidden layers 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 → $\frac{e^{z_i}}{\sum e^z}$

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:

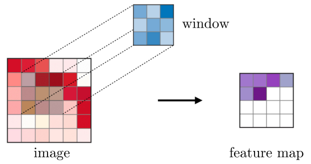
- Stopping training when validation performance drops
- Dropout - randomly drop some nodes during training to prevent over-reliance on a single node
- Embedding weight penalties into the objective function
- Batch Normalization - stabilizes learning by normalizing inputs to a layer

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.



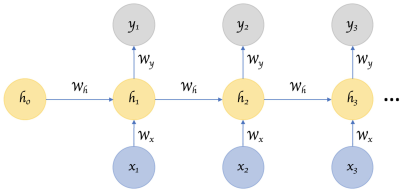
Pooling - downsamples convolution layers to reduce dimensionality and maintain spatial invariance, allowing detection of features even if they have shifted slightly. Common techniques return the max or average value in the pooling window.

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.

Attention and Transformers

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

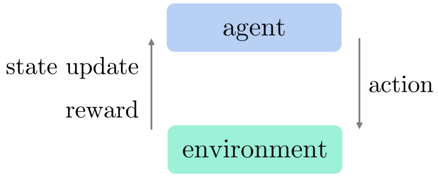
Model-based Approaches - predict ratings of unrated items, through methods such as Bayesian networks, SVD, and clustering. Handles sparse data better than memory-based approaches.

- 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

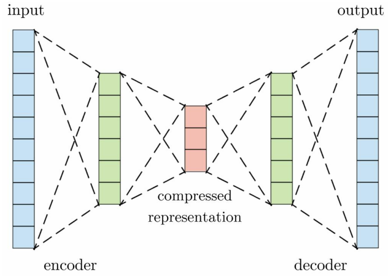
Anomaly detection techniques spans a wide range, including methods based on:
Statistics - relies on various statistical methods to identify outliers, such as Z-tests, boxplots, interquartile ranges, and variance comparisons
Density - useful when data is grouped around dense neighborhoods, measured by 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.

Time Series

Extracts characteristics from time-sequenced data, which may exhibit the following characteristics:

- Stationarity - statistical properties such as mean, variance, and auto correlation are constant over time
- Trend - long-term rise or fall in values
- Seasonality - variations associated with specific calendar times, occurring at regular intervals less than a year
- Cyclicality - variations without a fixed time length, occurring in periods of greater or less than one year
- Autocorrelation - degree of linear similarity between current and lagged values

CV must account for the time aspect, such as for each fold F_x:

- Sliding Window - train F_1, test F_2, then train F_2, test F_3
- Forward Chain - train F_1, test F_2, then train F_1, F_2, test F_3

Exponential Smoothing - uses an exponentially decreasing weight to observations over time, and takes a moving average. The time t output is $s_t = \alpha x_t + (1-\alpha)s_{t-1}$, where $0 < \alpha < 1$.

Double Exponential Smoothing - applies a recursive exponential filter to capture trends within a time series

$$\begin{aligned} s_t &= \alpha x_t + (1-\alpha)(s_{t-1} + b_{t-1}) \\ b_t &= \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1} \end{aligned}$$

Triple exponential smoothing adds a third variable γ that accounts for seasonality.

ARIMA - models time series using three parameters (p, d, q):

- Autoregressive - the past p values affect the next value
- Integrated - values are replaced with the difference between current and previous values, using the difference degree d (0 for stationary data, and 1 for non-stationary)
- Moving Average - the number of lagged forecast errors and the size of the moving average window q

SARIMA - models seasonality through four additional seasonality-specific parameters: P, D, Q, and the season length s

Prophet - additive model that uses non-linear trends to account for multiple seasonalities such as yearly, weekly, and daily. Robust to missing data and handles outliers well.
Can be represented as: $y(t) = g(t) + s(t) + h(t) + \epsilon(t)$, with four distinct components for the growth over time, seasonality, holiday effects, and error. This specification is similar to a generalized additive model.

Generalized Additive Model - combine predictive methods while preserving additivity across variables, in a form such as $y = \beta_0 + f_1(x_1) + \cdots + f_m(x_m)$, where functions can be non-linear. GAMs also provide regularized and interpretable solutions for regression and classification problems.

Naive Bayes

Setup. Data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with classes $y_i \in \{1, \ldots, C\}$ and features $x_i = (x_{i1}, \ldots, x_{id})$. Class prior $\pi_c = P(Y{=}c)$. Naive assumption: $(X_1, \ldots, X_d) \perp\!\!\!\perp$ given Y.

Bayes' Theorem.

$$P(Y{=}c \mid x) = \frac{P(Y{=}c)p(x \mid c)}{p(x)}, \quad p(x) = \sum_{c'=1}^{C} \pi_{c'}\, p(x \mid c')\,.$$

Under Naive independence:

$$p(x \mid c) = p(x_1, \ldots, x_d \mid c) = \prod_{j=1}^{d} p(x_j \mid c).$$ *Proof.* By chain rule $p(x_1, \ldots, x_d \mid c) = \prod_{j=1}^{d} p(x_j \mid x_{1:j-1}, c)$ and by conditional independence $p(x_j \mid x_{1:j-1}, c) = p(x_j \mid c)$, hence the product.

MAP rule.

$$\hat{y}(x) = \arg\max_{c \in [C]} P(Y{=}c \mid x) \propto \arg\max_{c}\ \pi_c \prod_{j=1}^{d} p(x_j \mid Y{=}c)\,.$$

In log-space: $$\hat{y}(x) = \arg\max_c \left[\log \pi_c + \sum_{j=1}^{d} \log p(x_j \mid c)\right].$$

Posterior. $$P(Y{=}c \mid x) = \frac{\pi_c \prod_j p(x_j \mid c)}{\sum_{c'=1}^{C} \pi_{c'} \prod_j p(x_j \mid c')}\,.$$

Priors (imbalanced classes). Empirical/Laplace-smoothed: $\hat{\pi}_c = \dfrac{N_c + \alpha}{N + \alpha C}$ with $N_c = \sum_{i=1}^{N} \mathbf{1}\{y_i{=}c\}$.

Bernoulli NB (binary features). $x_j \in \{0, 1\}$. Likelihood: $$p(x \mid c) = \prod_{j=1}^{d} \theta_{cj}^{x_j} (1 - \theta_{cj})^{1-x_j}, \ \hat{\theta}_{cj} = \frac{N_{cj}^{(1)} + \beta}{N_c + 2\beta}$$ where $N_{cj}^{(1)} = \sum_{i:y_i=c} x_{ij}$.

Multinomial NB (counts / BoW). $x_j \in \mathbb{N}$, vocabulary size V. Likelihood (up to multinomial coeff.): $p(x \mid c) \propto \prod_{j=1}^{d} \phi_{cj}^{x_j}$, $$\hat{\phi}_{cj} = \frac{N_{cj} + \beta}{\sum_{k=1}^{d} N_{ck} + \beta V}$$ where $N_{cj} = \sum_{i:y_i=c} x_{ij}$.

Gaussian NB (continuous). $x_j \in \mathbb{R}$, $p(x_j \mid c) = \mathcal{N}(\mu_{cj}, \sigma_{cj}^2)$. Decision: $$\hat{y}(x) = \arg\max_c \left[\log \pi_c - \tfrac{1}{2} \sum_{j=1}^{d} \left(\log(2\pi\sigma_{cj}^2) + \tfrac{(x_j - \mu_{cj})^2}{\sigma_{cj}^2}\right)\right].$$

Estimates: $$\hat{\mu}_{cj} = \frac{1}{N_c} \sum_{i:y_i=c} x_{ij}, \quad \hat{\sigma}_{cj}^2 = \frac{1}{N_c} \sum_{i:y_i=c} (x_{ij} - \hat{\mu}_{cj})^2$$ (use $1/(N_c{-}1)$ for unbiased).

Notes. Use $\alpha, \beta > 0$ (Laplace/Lidstone) to avoid zeros; set π_c to reflect true prevalences in imbalanced data.

Statistics

***p*-value** - probability that an effect could have occurred by chance. If less than the significance level α, or if the test statistic is greater than the critical value, then reject the null.
Type I Error (False Positive α) - rejecting a true null
Type II Error (False Negative β) - not rejecting a false null
Decreasing Type I Error causes an increase in Type II Error
Confidence Level (1 - α) - probability of finding an effect that did not occur by chance and avoiding a Type I error
Power (1 - β) - probability of picking up on an effect that is present and avoiding a Type II Error
Confidence Interval - estimated interval that models the long-term frequency of capturing the true parameter value
***z*-test** - tests whether normally distributed population means are different, used when n is large and variances are known

- z-score - the number of standard deviations between a data point x and the mean $\to \frac{x-\mu}{\sigma}$

***t*-test** - used when population variances are unknown, and converges to the z-test when n is large

- t-score - uses the standard error as an estimate for population variance $\to \frac{x-\mu}{s/\sqrt{n}}$

Degrees of Freedom - the number of independent (free) dimensions needed before the parameter estimate can be determined
Chi-Square Tests - measure differences between categorical variables, using $\chi^2 = \sum \frac{observed - expected}{expected}$ to test:

- Goodness of fit - if samples of one categorical variable match the population category expectations
- Independence - if being in one category is independent of another, based off two categories
- Homogeneity - if different subgroups come from the same population, based off a single category

ANOVA - analysis of variance, used to compare 3+ samples

- F-score - compares the ratio of explained and unexplained variance $\to \frac{\text{between group variance}}{\text{within group variance}}$

Conditional Probability $P(A \mid B) = \frac{P(A \cap B)}{P(B)}$
If A and B are independent, then $P(A \cap B) = P(A)P(B)$. Note, events that are independent of themselves must have probability either 1 or 0.
Union $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
Mutually Exclusive - events cannot happen simultaneously

Expected Value $E[X] = \sum x_i p_i$, with properties

- $E[X + Y] = E[X] + E[Y]$
- $E[XY] = E[X]E[Y]$ if X and Y are independent

Variance $\text{Var}(X) = E[X^2] - E[X]^2$, with properties

- $\text{Var}(X \pm Y) = \text{Var}(X) + \text{Var}(Y) \pm 2\text{Cov}(X, Y)$
- $\text{Var}(aX \pm b) = a^2 \text{Var}(X)$

Covariance - measures the direction of the joint linear relationship of two variables $\to \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n-1}$
Correlation - normalizes covariance to provide both strength and direction of linear relationships $\to r = \frac{Cov(x,y)}{\sigma_x \sigma_y}$
Independent variables are uncorrelated, though the inverse is not necessarily true

A/B Testing

Examines user experience through randomized tests with two variants. The typical steps are:

1. Determine the evaluation metric and experiment goals
2. Select a significance level α and power threshold 1 - β
3. Calculate the required sample size per variation
4. Randomly assign users into control and treatment groups
5. Measure and analyze results using the appropriate test

The required sample size depends on α, β, and the MDE
Minimum Detectable Effect - the target relative minimum increase over the baseline that should be observed from a test

Overall Evaluation Criterion - quantitative measure of the test's objective, commonly used when short and long-term metrics have inverse relationships

Multivariate Testing - compares 3+ variants or combinations, but requires larger sample sizes
Bonferroni Correction - when conducting n tests, run each test at the $\frac{\alpha}{n}$ significance level, which lowers the false positive rate of finding effects by chance

Network Effects - changes that occur due to effect spillover from other groups. To detect group interference:

1. Split the population into distinct clusters
2. Randomly assign half the clusters to the control and treatment groups A_1 and B_1
3. Randomize the other half at the user-level and assign to control and treatment groups A_2 and B_2
4. Intuitively, if there are network effects, then the tests will have different results

To account for network effects, randomize users based on time, cluster, or location

Sequential Testing - allows for early experiment stopping by drawing statistical borders based on the Type I Error rate. If the effect reaches a border, the test can be stopped. Used to combat *peeking* (preliminarily checking results of a test), which can inflate p-values and lead to incorrect conclusions.
Cohort Analysis - examines specific groups of users based on behavior or time and can help identify whether novelty or primacy effects are present

Miscellaneous

Shapley Values - measures the marginal contribution of each variable in the output of a model, where the sum of all Shapley values equals the total value (prediction − mean prediction)
SHAP - interpretable Shapley method that utilizes both global and local importance to model variable explainability
Permutation - order matters → $\frac{n!}{(n-k)!} = {}^nP_k$
Combination - order doesn't matter
→ $\frac{n!}{k!(n-k)!} = {}^nC_k = \binom{n}{k}$
Left Skew - Mean $<$ Median $\leq$ Mode
Right Skew - Mean $>$ Median $\geq$ Mode
Probability vs Likelihood - given a situation θ and observed outcomes O, probability is calculated as $P(O|\theta)$. However, when true values for θ are unknown, O is used to estimate the θ that maximizes the likelihood function. That is, $L(\theta|O) = P(O|\theta)$.