Machine Learning: The Art and Science of Algorithms that Make Sense of Data

Tim Lawson

November 2, 2023

These notes are based on Flach 2012.

Contents

Theorems

5 Tree models

Trees A tree is an undirected connected acyclic graph. A rooted tree is a tree in which a node is designated the root. The edges of a rooted tree may be directed away from or towards the root. The nodes of an m-ary tree have at most m children.

Tree models

- A *tree model* is represented by a directed rooted tree. The branch nodes represent features and the leaf nodes represent instance space segments.
- A *feature tree* is represented by a binary directed rooted tree. There are two edges directed away from each branch node, each of which represents a mutually exclusive proposition about the value of the feature.
- A *decision tree* is represented by an m-ary directed rooted tree where $m \geq 2$. There are m_i edges directed away from each branch node i, each of which represents a possible value of the feature.

TODO:

- Disjunctive normal form
- Distributive equivalence $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
- De Morgan laws $\neg(A \vee B) \equiv \neg A \wedge \neg B$

Expressivity A decision tree represents a set of mutually exclusive logical expressions, which can be written in different equivalent forms. The expressions represented by a decision tree may not be equivalent to *conjunctive* expressions of individual features[1]. Because any logical expression may be written in disjunctive normal form, decision trees are maximally *expressive*, i.e., they can separate any data that is consistently labelled.

However, expressive hypothesis languages are prone to overfitting and one way to prevent overfitting is to choose a restrictive hypothesis language. Learning algorithms in expressive hypothesis spaces typically have an *inductive bias* towards simpler hypotheses, either implicitly by the search procedure or explicitly by a term in the loss function.

Bias and variance Low-bias models are more likely to overfit to the training data. Low-variance models change by a small amount when the training data changes by a small amount.

Low-variance, high-bias models are preferable when there is limited training data and overfitting is a concern. High-variance, low-bias models are preferable when there is plenty of training data but underfitting is a concern.

[1] But they may be equivalent to conjunctive expressions of *conjunctive features*. This is called *constructive induction*.

Learning algorithms A feature tree represents conjunctive concepts in the hypothesis space. The learning problem is to choose the best conjunctive concepts to solve a task.

Algorithm 5.1 (pseudo-code) is a generic learning procedure. It is a *divide-and-conquer* algorithm: it splits the data into subsets, learns a tree for each subset, and combines them. It is also a *greedy* algorithm: it always chooses the best feature values to split the data at a given step, which may be sub-optimal. An optimal but more computationally expensive alternative is to search for the best feature values to split the data over all steps.

Algorithm 5.1.

```
# Returns true if data can be given a single label.
def is_homogeneous(data)

# Returns the best label for data.
def label(data)

# Returns the best feature values to split data.
def find_feature_values(data, features)

# Returns the subsets of data for each feature value.
def find_split(data, feature, values)

def grow(tree, data, features):
  if is_homogeneous(data):
    tree.add_leaf(label(data))

  feature, values = find_feature_values(data, features)

  for subset in find_split(data, feature, values):
    if len(subset) > 0:
      grow(tree[feature], subset, features)
    else:
      tree[feature].add_leaf(label(subset))
```

5.1 Decision trees

5.1.1 Purity of a leaf

For a classification task, a set of instances is *homogeneous* if the instances belong to the same class. Therefore, **def** label returns the majority class. The *purity* of a set of instances is the proportion of instances that belong to the majority class. It is proportional to the *empirical probability* $\dot{p}$.

5.1.2 Impurity of a leaf

def find_feature_values returns the feature values that maximise the purity (minimise the impurity) of the subsets. In terms of $\dot{p}$, the impurity f must obey the following constraints:

- $f(\dot{p}) = 0 : \dot{p} \in 0, 1$, i.e., it is zero if the subset is homogeneous

- $f(\dot{p}) = f(1-\dot{p})$, i.e., it is symmetric about $\dot{p} = \frac{1}{2}$
- $\arg\max_{\dot{p}} f = \frac{1}{2}$, i.e., it is maximal when $\dot{p} = \frac{1}{2}$

Some examples of impurity functions are:

- *Minority class* $\min(\dot{p}, 1-\dot{p})$

  The error rate, i.e., the proportion of instances that are labelled incorrectly if they are labelled with the majority class.

- *Gini index* $2\dot{p}(1-\dot{p})$

  The expected error if we label instances randomly.

- *Entropy* $-\dot{p}\log_2 \dot{p} - (1-\dot{p})\log_2(1-\dot{p})$

  The expected number of bits encoded by the class of a random instance.

5.1.3 Impurity of a tree

The impurity of a set of mutually exclusive leaves, i.e., a decision tree, is the weighted average of the impurities of the leaves:

$$f(\{D_i \mid i \in 1, ..., n\}) = \frac{1}{|D|}\sum_{i=1}^{n} |D_i| f(\dot{p}_i) \tag{1}$$

For binary classification, we can find $f(\{D_+, D_-\})$ from $f(\dot{p}_+)$ and $f(\dot{p}_-)$ geometrically: First, we draw a straight line between $(\dot{p}_+, f(\dot{p}_+))$ and $(\dot{p}_-, f(\dot{p}_-))$. The line represents the possible weighted averages of $f(\dot{p}_+)$ and $f(\dot{p}_-)$. Given that $\dot{p} = \frac{|D_+|}{|D|}\dot{p}_+ + \frac{|D_-|}{|D|}\dot{p}_-$, $f(\dot{p})$ is the point on the line that corresponds to $\dot{p}$.

5.1.4 Multi-class classification

Impurity functions can be generalized to $k > 2$ classes, e.g.:

- *k-class Gini index* $\sum_{i=1}^{k} \dot{p}_i(1-\dot{p}_i)$
- *k-class entropy* $\sum_{i=1}^{k} -\dot{p}_i \log_2 \dot{p}_i$

5.1.5 Purity and information gain

To split a parent node D into children $\{D_i \mid i = 1, ..., n\}$, we typically choose the feature that maximises the *purity gain*:

$$f(D) - f(\{D_i \mid i = 1, ..., n\}) \tag{2}$$

If $f(D)$ is the entropy, this is called the *information gain*. It measures the increase in information about the class gained by including the feature. A 'best split' algorithm finds the feature that minimises $f(\{D_i \mid i = 1, ..., n\})$.

5.2 Ranking and probability estimation trees

A grouping classifier can be used to rank instances by learning an ordering on its instance-space segments. Decision trees can access the class distributions (empirical probabilities) of the segments, from which an ordering can be derived that is optimal for the training data. This is not possible for some other grouping classifiers.

The ordering is optimal because it produces a convex ROC curve. The ROC curve is convex because its segments are sorted in decreasing order of slope. The slope of a segment is $\frac{\dot{p}}{1-\dot{p}}$ and, because the slope is a monotonic function of $\dot{p}$, sorting the segments in decreasing order of $\dot{p}$ is equivalent to sorting them in decreasing order of slope.

The empirical probability of a parent node is a weighted average of the empirical probabilities of its children (see 1):

$$\dot{p} = \frac{1}{|D|} \sum_{i=1}^{n} |D_i| \dot{p}_i \tag{3}$$

But this does not constrain the empirical probabilities of a parent's children, so we cannot find the ordering of segments from the tree structure.

TODO:

- Interpretation of splits in terms of coverage curves. To add a split: split the line segment of the ROC curve into $k > 2$ segments and re-sort the segments in decreasing order of slope. Sorting the segments ensures that the ROC curve is convex.
- Turning a feature tree into a decision tree (classifier), ranking tree, or probability estimation tree.
  - Decision tree (classifier): choose the operating conditions and find the optimal point under those conditions.
  - Ranking tree: order the segments in decreasing order of empirical probability.
  - Probability estimation tree: predict the empirical probabilities of the segments (applying smoothing).
- Pruning trees.
  - Merging all leaves in a subtree
  - Only recommended for classification and when you can define the operating conditions
  - E.g., reduced-error pruning with a pruning set
  - Never improves accuracy over the training data
- Sensitivity to imbalanced classes.
  - Oversampling the minority class. Applies to any model without changing the model itself. But increases training time and may not change the model(!).
- Relative impurity.

  - The relative impurity of a child node is its weighted impurity in proportion to its parent node's impurity.
  - Some impurity measures are invariant with respect to the class distribution. E.g., the square root of the Gini index, which minimises the relative impurity.
  - Impurity measures that vary with the class distribution produce splitting criteria that emphasise child nodes with more instances. E.g., the Gini index and entropy.

How to train a decision tree:

1. Prioritise ranking performance
2. Use an impurity measure that is invariant with respect to the class distribution. Otherwise, oversample the minority class to balance the class distribution.
3. Apply Laplace or add-k smoothing to the empirical probabilities.
4. Given the operating conditions, select the best operating point on the ROC curve.
5. Optionally, prune subtrees whose leaves are homogeneous.

5.3 Tree learning as variance reduction

TODO:

- Adapting decision trees to regression and clustering tasks.
- Variance of a Bernoulli distribution.
- Overfitting, pruning, and model trees.
- Cluster and split dissimilarity.

7 Linear models

- Linear models are defined in terms of the geometry of the instance space.
- Real-valued features are not generally intrinsically geometric.
- However, we can use geometric concepts to structure the instance space (e.g., lines and planes) and represent similarity by distance.

Linear models are simple:

- They are parametric: they have a fixed structure that is defined by numeric parameters that are learned from the training data. By contrast, tree and rule models are non-parametric: their structure is not fixed prior to learning.
- They are stable (have low variance): small variations in the training data have a small effect on the learned model. Tree models have high variance.
- They are unlikely to overfit the training data because they have relatively few parameters (have high bias). However, they sometimes underfit the training data.

7.1 The least-squares method

The least-squares method can be used to learn linear models for classification and regression. It finds a function estimator that minimises the sum of squared residuals (differences between the actual and estimated values).

Univariate linear regression Let $\{(x_i, y_i) \mid i \in 1..n\}$ be a set of instances. Approximate the true function $f(x_i) = y_i$ by a linear function $f'(x_i) = a + bx_i$. Univariate linear regression finds a, b such that the sum of squared residuals $\sum_{i=1}^{n}(y_i - (a + bx_i))^2$ is minimized.

When the sum of squared residuals is minimized, its partial derivatives with respect to a and b are zero:

$$\frac{\partial}{\partial a}\sum_{i=1}^{n}(y_i - (a + bx_i))^2 = -2\sum_{i=1}^{n}(y_i - (a + bx_i)) \quad = 0 \tag{4}$$

$$\frac{\partial}{\partial b}\sum_{i=1}^{n}(y_i - (a + bx_i))^2 = -2\sum_{i=1}^{n}(y_i - (a + bx_i))x_i = 0 \tag{5}$$

TODO

- Translation does not affect the regression coefficient, only the intercept. We can zero-centre the x-values by subtracting the mean $\bar{x}$.
- If we normalize x to have unit variance, then the regression coefficient is the covariance between the normalized x and y.

The least-squares solution is equivalent to the maximum likelihood estimate given the assumptions that the true function is linear but normally-distributed noise is added to the instance y-values. If noise is added to only the y-values, then it is called *ordinary* least squares, which has a unique solution. If noise is added to both x- and y-values, then it is called *total* least squares, which does not necessarily have a unique solution.

Zero-centred matrix, scatter matrix, covariance matrix.

Multivariate linear regression Matrix form and homogeneous coordinates. Transformation to decorrelate, centre and normalize features. If the features are assumed to be uncorrelated, a multivariate linear regression problem decomposes into a set of univariate linear regression problems. It is computationally expensive to invert the scatter (covariance) matrix.

Regularization Least-squares regression can be unstable. Instability demonstrates a tendency to overfit. Regularization helps to avoid overfitting by constraining the weight vector.

- Shrinkage: makes the average magnitude of the weights small. This adds a scalar parameter to the diagonal of the scatter matrix, which improves the numerical stability of matrix inversion. Least-squares regression with shrinkage is called ridge regression.

- Lasso (least absolute shrinkage and selection operator): This adds the sum of the absolute weights (L_1 regularization). This makes the magnitude of some weights smaller but sets others to zero, i.e., it favours sparse solutions.

7.2 The perceptron

7.3 Support vector machines

7.4 Obtaining probabilities from linear classifiers

7.5 Notes

When to favour models with different characteristics? E.g., the quantity and quality training data.

Normalization (zero-centre, unit variance) Write up the equivalencies between correlation coefficients etc.

Regularization Relation to, e.g., Bayesian priors Technically, e.g., sparsity (Occam's razor). Regularization changes the optimal solution (it's included in the loss)

Correlation (in the extreme case, two copies of the same feature) — your problem is underspecified, i.e., there are infinitely many solutions (which are combinations of the two). 'Spikiness of the fitness landscape' (high variance). Regularization: decreasing dependence on the data, i.e., increasing bias and decreasing variance.

When to choose ridge (L2) vs lasso (L1) regularization? An elastic net uses a weighted combination of the two where the weight is a hyperparameter that you can tune.

Why does lasso produce sparse solutions? With Euclidean distance, the set of points at equal distance is a circle. If you change the exponent, e.g., Minkowski at d = 3, that changes shape. E.g. d = 1 'pulls you' towards a solution on one of the axes, i.e., towards one or the other feature instead of a combination of both (i.e., sparsity). Vector field analysis.

LP-norm where p is an integer.

Multivariate.

8 Distance-based models

A distance-based model is generally comprised of:

- a distance metric (section 8.1);
- a set of exemplars (section 8.2.2); and
- a distance-based decision rule (e.g., section 8.2.3).

8.1 Distance metrics

Definition 8.1 (Metric)**.** *A metric is a function $d : M \times M \to \mathbb{R}$, where M is a set of points, such that:*

1. *$d(x, x) = 0 \ \forall \ x \in M$ (the distance from a point to itself is zero)*
2. *$d(x, y) > 0 \ \forall \ x, y \in M, x \neq y$ (positivity)*
3. *$d(x, y) = d(y, x) \ \forall \ x, y \in M$ (symmetry)*
4. *$d(x, z) \leq d(x, y) + d(y, z) \ \forall \ x, y, z \in M$ (triangle inequality)*

Definition 8.2 (Pseudo-metric)**.** *A pseudo-metric is a metric where the condition of positivity is replaced by non-negativity, i.e., $d(x, y) \geq 0 \ \forall \ x, y \in M$.*

8.1.1 Norms

Definition 8.3 (p-norm, L_p norm)**.** *The p-norm of a vector $\vec{x} \in \mathbb{R}^n$ is:*

$$\|\vec{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \tag{6}$$

Definition 8.4 (0-“norm”, L_0 “norm”)**.** *The 0-“norm” of a vector $\vec{x} \in \mathbb{R}^n$ is the number of non-zero elements in $\vec{x}$:*

$$\|\vec{x}\|_0 = \sum_{i=1}^{n} |x_i|^0 \tag{7}$$

The 0-“norm” is not a norm because it is not *homogeneous*, i.e., $f(ax) \neq af(x) \ \forall \ a \in \mathbb{R}, x \in X$.

8.1.2 Distances

Definition 8.5 (Minkowski distance)**.** *The Minkowski distance of order $p \in \mathbb{N}_1$ between two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ is:*

$$D_p(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} = \|\vec{x} - \vec{y}\|_p \tag{8}$$

Definition 8.6 (Manhattan distance)**.** *The Manhattan distance between two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ is the Minkowski distance of order $p = 1$:*

$$D_1(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{9}$$

Definition 8.7 (Euclidean distance)**.** *The Euclidean distance between two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ is the Minkowski distance of order $p = 2$:*

$$D_2(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{10}$$

Definition 8.8 (Chebyshev distance)**.** *The Chebyshev distance between two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ is the Minkowski distance of order $p \to \infty$:*

$$D_\infty(\vec{x}, \vec{y}) = \lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} = \max_{i=1}^{n} |x_i - y_i| \tag{11}$$

Minkowski distances are translationally invariant but not scale-invariant. Euclidean distance is the only Minkowski distance that is rotationally invariant.

Definition 8.9 (Hamming distance)**.** *The Hamming distance between two binary strings $\vec{x}, \vec{y}$ of length n is the number of bits in which they differ:*

$$D_0(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|^0 = \sum_{i=1}^{n} \mathbb{I}(x_i \neq y_i) \tag{12}$$

The edit or *Levenshtein distance* generalises the Hamming distance to non-binary strings of different lengths.

Definition 8.10 (Mahalanobis distance)**.** *The Mahalanobis distance between two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$, where Σ is the covariance matrix, is:*

$$D_M(\vec{x}, \vec{y} \mid \Sigma) = \sqrt{(\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y})} \tag{13}$$

Euclidean distance is the Mahalanobis distance where the covariance matrix is the identity matrix.

8.2 Neighbours and exemplars

8.2.1 Means and medians

Minimising the sum of squared Euclidean distances is equivalent to minimising the *average* squared Euclidean distance.

Theorem 8.11 (Arithmetic mean minimises squared Euclidean distance)**.** *The arithmetic mean $\vec{\mu}$ of a set of points $X \in \mathbb{R}^n$ is the point with the minimum sum of squared Euclidean distances to the points in X:*

$$\arg\min_{\vec{y}} \sum_{\vec{x} \in X} \|\vec{x} - \vec{y}\|_2^2 = \vec{\mu} \tag{14}$$

Proof. The gradient of the sum of squared Euclidean distances is:

$$\begin{aligned} \nabla_{\vec{y}} \sum_{\vec{x} \in X} \|\vec{x} - \vec{y}\|_2^2 &= -2 \sum_{\vec{x} \in X} (\vec{x} - \vec{y}) \\ &= -2 \sum_{\vec{x} \in X} \vec{x} + 2|X|\vec{y} \end{aligned}$$

If the gradient is the zero vector, then:

$$\vec{y} = \frac{1}{|X|} \sum_{\vec{x} \in X} \vec{x} = \vec{\mu}$$

□

The *geometric median* minimises the sum of Euclidean distances. However, there is no closed-form expression for the geometric median of multivariate data.

8.2.2 Centroids and medoids

A *centroid* is an exemplar that is not necessarily an instance, whereas a *medoid* must be an instance. An algorithm to find the medoid of a set of n instances has time complexity $O(n^2)$. This is because the distance between every pair of instances must be computed.

8.2.3 Binary linear classifiers

A binary linear classifier finds the exemplars that minimise the sum of squared Euclidean distances to the instances in each class. Its decision boundary is the perpendicular bisector of the line segment that connects the exemplars. Alternatively, it applies the *decision rule* that an instance belongs to the class with the nearest exemplar.

8.2.4 Multi-class linear classifiers

A distance-based interpretation of the binary linear classifer generalises to $k > 2$ classes. With k exemplars, each decision region is bounded by $k - 1$ line segments. Dependent on the distance metric, some decision regions become closed cells as the number of exemplars increases. This is called *Voronoi tesselation.* Generally, the number of exemplars is greater than the number of classes.

8.3 Nearest-neighbour classifiers

Paragraph 8.2.4 generalised the binary linear classifier to $k > 2$ classes. The *nearest-neighbour classifier* is simpler: it takes each instance to be an exemplar. Its decision regions are sets of Voronoi cells (because adjacent cells may have the same class).

8.3.1 Classifier properties

The nearest-neighbour classifier has low bias and high variance. For n instances, it is 'trained' in $O(n)$ time. However, it may also take $O(n)$ time to classify a new instance. This is because the new instance must be compared with every training instance. The classification time can be decreased at the expense of the training time by choosing the data structure that stores the instances.

8.3.2 Dimensionality

High-dimensional instance spaces are sparse, i.e., the distance between any two instances is large. The effective dimensionality may be smaller: some dimensions may be irrelevant or the instances may lie on a lower-dimensional manifold. The dimensionality of the instance space can be reduced by *feature selection* or techniques such as *principal component analysis* (PCA) before applying a distance-based model.

8.3.3 *k*-nearest neighbours

If there is a way to aggregate over exemplars, then k nearest neighbours can be used (where k is distinct from the number of classes). An example of a decision rule for a k-nearest-neighbour classifier is to predict the majority class of the k nearest exemplars.

The model's properties depend on the choice of k: as it increases, the the refinement first increases and then decreases, the bias increases, and the variance decreases. The dependence on k can be lessened by applying *distance weighting* to the exemplars.

8.3.4 Regression

Nearest-neighbour approaches are agnostic with respect to the type of the target variable and can be applied to regression problems. With k-nearest-neighbours, the predicted value is typically the mean of the k nearest exemplars, which may be distance-weighted.

8.4 Clustering

For distance-based models, unsupervised learning generally refers to clustering. A predictive distance-based clustering method has the same elements as a distance-based classifier (section 8). Instead of an explicit target variable, the distance metric is taken to represent the learning target. Therefore, the aim is to find *compact* clusters with respect to the distance metric.

Definition 8.12 (Within-cluster scatter matrix)**.** *Let $X = \bigcup_{i=1}^{k} X_i$ be a set of instances partitioned into k classes. The within-cluster scatter matrix S_i is the scatter matrix of X_i.*

Definition 8.13 (Between-cluster scatter matrix)**.** *Let $X = \bigcup_{i=1}^{k} X_i$ be a set of instances partitioned into k classes. The between-cluster scatter matrix B is the scatter matrix of X where each instance is replaced by its centroid $\vec{\mu}_i$.*

Theorem 8.14 (Relation of within- and between-cluster scatter matrices)**.**

$$S = \sum_{i=1}^{k} S_i + B \tag{15}$$

$$\operatorname{Tr} S = \sum_{i=1}^{k} \operatorname{Tr} S_i + \sum_{i=1}^{k} |X_i| \|\vec{\mu}_j - \vec{\mu}\|^2 \tag{16}$$

Minimising the total within-cluster scatter is equivalent to maximising the scatter of the centroids, weighted by the number of instances in each class.

8.4.1 K-means

The *k-means* problem is to find the partition of X that minimises the total within-cluster scatter (section 8.4). In this context, k is the number of clusters instead of a number of classes. The k-means problem is NP-complete, i.e., there is no efficient way to find the global minimum. The typical heuristic algorithm is called k-means or *Lloyd's algorithm.* It iterates between partitioning the data with the nearest-centroid decision rule and recomputing the centroids from the partition.

An iteration of the k-means algorithm cannot decrease the total within-cluster scatter, so it reaches a *stationary point* (a local minimum). It converges to a stationary point in a finite number of iterations but there is no way to know whether it is optimal (the global minimum). Therefore, it is recommended to run k-means multiple times and choose the best solution.

8.4.2 K-medoids

The k-medoids algorithm uses medoids instead of centroids. Again, k is the number of clusters instead of a number of classes. An alternative is the *partitioning around medoids* (PAM) algorithm, which swaps medoids with other instances in the class. The clustering quality is the distance between the medoids and the instances in the class. Each iteration takes $O(k(n-k)^2)$ time.

8.4.3 Shape

Methods that represent clusters only by exemplars disregard the shape of the clusters. This can lead to counter-intuitive results, e.g., scale-dependence. A method that also estimates the shape of clusters must take off-diagonal entries of the scatter matrix into account.

8.4.4 Silhouettes

Definition 8.15 (Silhouette)**.** *Let:*

- $X = \bigcup_{i=1}^{k} X_i \in \mathbb{X}$ *be a set of instances partitioned into* k *clusters;*
- $a(\vec{x}_j)$ *be the average distance to the other instances in* X_i*:*

$$a(\vec{x}_j) = \frac{1}{|X_i| - 1} \sum_{\vec{x} \in X_i, \vec{x} \neq \vec{x}_j} D(\vec{x}_j, \vec{x}) \quad \forall \quad \vec{x}_j \in X_i,\ X_i \in \mathbb{X} \tag{17}$$

- $b(\vec{x}_j)$ *be the average distance to the instances in the neighbouring cluster, i.e., the cluster with the nearest centroid:*

$$b(\vec{x}_j) = \min_{X_i \in \mathbb{X}, X_i \neq X_j} \frac{1}{|X_j|} \sum_{\vec{x} \in X_j} D(\vec{x}_j, \vec{x}) \quad \forall \quad \vec{x}_j \in X_i,\ X_i \in \mathbb{X} \tag{18}$$

The silhouette of $\vec{x}_j$ *is:*

$$s(\vec{x}_j) = \frac{b(\vec{x}_j) - a(\vec{x}_j)}{\max(a(\vec{x}_j), b(\vec{x}_j))} \tag{19}$$

The silhouette of X *is a plot of* $s(\vec{x}_j)$ *against* $\vec{x}_j$*, grouped by* X_i *and sorted in descending order of* $s(\vec{x}_j)$*.*

8.5 Hierarchical clustering

8.5.1 Dendrograms

A *dendrogram* is a tree diagram defined in terms of a distance metric. It is a descriptive rather than a predictive model, because it partitions the training set but not the entire instance space, and has high variance. It requires a definition of the distance between clusters.

Definition 8.16 (Dendrogram). *Let X be a set of instances. A dendrogram is a binary tree where each leaf is an instance in X, each branch is a subset of instances, and the level of each node is the distance between the clusters represented by its children.*

8.5.2 Linkage functions

A *linkage function* translates pairwise distances between instances into pairwise distances between clusters.

Definition 8.17 (Linkage function). *Let X be a set of instances and $D : X \times X \to \mathbb{R}$ be a distance metric. A linkage function $L : X \times X \to \mathbb{R}$ is the distance between $X_i, X_j \subseteq X$.*

Definition 8.18 (Single linkage). *The single linkage between $X_i, X_j \subseteq X$ is the minimum distance between any two instances in X_i and X_j:*

$$L(X_i, X_j) = \min_{\vec{x} \in X_i, \vec{y} \in X_j} D(\vec{x}, \vec{y}) \tag{20}$$

Definition 8.19 (Complete linkage). *The complete linkage between $X_i, X_j \subseteq X$ is the maximum distance between any two instances in X_i and X_j:*

$$L(X_i, X_j) = \max_{\vec{x} \in X_i, \vec{y} \in X_j} D(\vec{x}, \vec{y}) \tag{21}$$

Definition 8.20 (Average linkage). *The average linkage between $X_i, X_j \subseteq X$ is the average distance between any two instances in X_i and X_j:*

$$L(X_i, X_j) = \frac{1}{|X_i||X_j|} \sum_{\vec{x} \in X_i, \vec{y} \in X_j} D(\vec{x}, \vec{y}) \tag{22}$$

Definition 8.21 (Centroid linkage). *The centroid linkage between $X_i, X_j \subseteq X$ is the distance between the centroids of X_i and X_j:*

$$L(X_i, X_j) = D\left(\frac{1}{|X_i|} \sum_{\vec{x} \in X_i} \vec{x},\ \frac{1}{|X_j|} \sum_{\vec{y} \in X_j} \vec{y} \right) \tag{23}$$

8.5.3 Hierarchical agglomerative clustering

The algorithm to build a dendrogram is *agglomerative* (it works from the bottom up). Generally, it produces different results for different linkage functions. For single linkage, it adds links until there is a path between any two instances.

Hierarchical clustering for single linkage effectively computes and sorts the pairwise distances between instances, which takes $O(n^2)$ time. For other linkage functions, it takes at least $O(n^2 \log n)$ time. It is deterministic and always produces a clustering, which may not be high-quality. While the number of clusters does not need to be chosen in advance, both a distance metric and a linkage function must be chosen.

Single and complete linkage do not take the shape of clusters into account because they are defined in terms of the distance between a pair of instances. However, centroid linkage can produce counter-intuitive results because it violates *monotonicity*.

9 Probabilistic models

Recall that:

- $P(Y \mid X)$ is the posterior probability distribution of Y given X;
- $P(Y, X)$ is the joint probability distribution of Y and X;
- $P(X \mid Y)$ is the likelihood function; and
- $P(X)$ is the prior distribution of X.

Discriminative and generative models

- A *discriminative* model describes the posterior distribution of the target given the input. It does not describe the prior distribution of the input.
- A *generative* model describes a joint distribution of the target and input. If the prior distribution of the target can be estimated, then it can be described by a likelihood function. It can be used to generate data by sampling from the joint distribution.

Generative models can do more than discriminative models. However, joint distributions are harder to learn than conditional distributions like the posterior distribution because they are described by more probability values (parameters). This may be handled by simplifying assumptions like independence but they are not always appropriate.

Uncertainty A probabilistic view treats learning as a procedure that reduces uncertainty.

9.1 Normal distributions

Univariate

Definition 9.1 (Univariate normal distribution)**.**

$$\begin{aligned} P(x \mid \mu, \sigma) &= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) && (24) \\ &= \frac{1}{E} \exp\left(-\frac{z^2}{2}\right), \ E = \sqrt{2\pi}\sigma && (25) \end{aligned}$$

- *$\mu \in \mathbb{R}$ is the mean;*
- *$\sigma \in \mathbb{R}$ is the standard deviation; and*
- *$z = \frac{x-\mu}{\sigma}$ is the z-score.*

If $\mu = 0$ and $\sigma = 1$, then it is the *standard* univariate normal distribution:

$$P(x \mid \mu = 0, \sigma = 1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{26}$$

Multivariate

Definition 9.2 (Multivariate normal distribution)**.**

$$\begin{aligned} P(\vec{x} \mid \vec{\mu}, \Sigma) = \frac{1}{E_n} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})\right), \\ E_n = (2\pi)^{n/2}\sqrt{\det \Sigma} \end{aligned} \tag{27}$$

- *$\vec{\mu} \in \mathbb{R}^n$ is the mean; and*
- *$\Sigma \in \mathbb{R}^{n\times n}$ is the covariance matrix.*

If $\vec{\mu} = \vec{0}$ and $\Sigma = I$, then it is the *standard* multivariate normal distribution:

$$P(\vec{x} \mid \vec{\mu} = \vec{0}, \Sigma = I) = \frac{1}{(2\pi)^{n/2}} \exp\left(-\frac{\vec{x}\cdot\vec{x}}{2}\right) \tag{28}$$

9.1.1 Gaussian mixture models

A Gaussian mixture model is a mixture of k Gaussian distributions. For $k = 2$, i.e., binary classification, $X = \{x_i \mid i = 1..n\} = X_+ \cup X_-$.

Univariate In the univariate case, the likelihood ratio is:

$$\frac{P(X_+)}{P(X_-)} = \frac{\sigma_-}{\sigma_+} \exp\left(-\frac{1}{2}\left(\left(\frac{x-\mu_+}{\sigma_+}\right)^2 - \left(\frac{x-\mu_-}{\sigma_-}\right)^2\right)\right)$$

If $\sigma_+ = \sigma_- = \sigma$, then the likelihood ratio is:

$$\exp(\gamma(x-\mu)),\ \gamma = \frac{\mu_+ - \mu_-}{\sigma^2},\ \mu = \frac{\mu_+ + \mu_-}{2}$$

and the maximum-likelihood decision threshold, i.e., the value of x such that the likelihood ratio is 1, is μ.

If the standard deviations of the two Gaussian distributions are different, then there are two decision boundaries and a non-contiguous decision region for one of the classes.

Multivariate In the multivariate case, the likelihood ratio is:

$$\sqrt{\frac{\det \Sigma_+}{\det \Sigma_-}} \exp\left(-\frac{1}{2}\left((\vec{x}-\vec{\mu}_+)^T \Sigma_+^{-1} (\vec{x}-\vec{\mu}_+) - (\vec{x}-\vec{\mu}_-)^T \Sigma_-^{-1} (\vec{x}-\vec{\mu}_-)\right)\right)$$

If $\Sigma_+ = \Sigma_- = I$, i.e., for each class, the features are uncorrelated and have unit variance, then the likelihood ratio is:

$$\exp\left(-\frac{1}{2}\left(\|\vec{x}-\vec{\mu}_+\|^2 - \|\vec{x}-\vec{\mu}_-\|^2\right)\right)$$

and the maximum-likelihood decision boundary, i.e., the values of $\vec{x}$ such that the likelihood ratio is 1, is the hyperplane equidistant from $\vec{\mu}_+$ and $\vec{\mu}_-$.

This is the same as the decision boundary for the basic linear classifier. In other words, for uncorrelated Gaussian features with unit variance, the basic linear classifier is *Bayes-optimal.*

9.1.2 Distances and probabilities

The normal distribution demonstrates the connection between the geometric and probabilistic views of models. Effectively, it translates distances into probabilities.

Definition 9.3. *The multivariate normal distribution (definition 9.2) can be expressed in terms of the Mahalanobis distance (definition 8.10):*

$$P(\vec{x} \mid \vec{\mu}, \Sigma) = \frac{1}{E_n} \exp\left(-\frac{1}{2} D_M(\vec{x}, \vec{\mu} \mid \Sigma)^2\right) \tag{29}$$

Definition 9.4. *The negative logarithm of the Gaussian likelihood is proportional to the squared Mahalanobis distance (definition 8.10):*

$$-\ln P(\vec{x} \mid \vec{\mu}, \Sigma) = \ln E_d + \frac{1}{2} D_M(\vec{x}, \vec{\mu} \mid \Sigma)^2 \tag{30}$$

Theorem 9.5. *Let $P(\vec{x} \mid \vec{\mu}, \Sigma)$ be a multivariate normal distribution. The maximum-likelihood estimate of $\vec{\mu}$ is the point that minimises the sum of squared Mahalanobis distances to the data points $X = \{\vec{x}_i \mid i = 1..n\}$.*

Proof. The maximum-likelihood estimate is the value of $\vec{\mu}$ that maximises the joint likelihood of X:

$$\vec{\hat{\mu}} = \arg\max_{\vec{\mu}} P(X \mid \vec{\mu}, \Sigma) \tag{31}$$

Assume that the data points are independently sampled from $P(\vec{x} \mid \vec{\mu}, \Sigma)$. Then, the joint likelihood is the product of the likelihoods of the data points:

$$P(X \mid \vec{\mu}, \Sigma) = \prod_{i=1}^{n} P(\vec{x}_i \mid \vec{\mu}, \Sigma) \tag{32}$$

By definitions 9.3 and 9.4:

$$\vec{\hat{\mu}} = \arg\max_{\vec{\mu}} \prod_{i=1}^{n} \frac{1}{E_n} \exp\left(-\frac{1}{2} D_M(\vec{x}, \vec{\mu} \mid \Sigma)^2\right) \tag{33}$$

$$= \arg\min_{\vec{\mu}} \sum_{i=1}^{n} \left(\ln E_d + \frac{1}{2} D_M(\vec{x}, \vec{\mu} \mid \Sigma)^2\right) \tag{34}$$

$$= \arg\min_{\vec{\mu}} \sum_{i=1}^{n} D_M(\vec{x}, \vec{\mu} \mid \Sigma)^2 \tag{35}$$

□

Definition 9.6. *The standard normal distribution with $n = 2$ (definition 9.2) can be expressed in terms of the Euclidean distance (definition 8.7):*

$$P(\vec{x} \mid \vec{0}, I) = \frac{1}{E_2} \exp\left(-\frac{1}{2} D_2(\vec{x}, \vec{0})^2\right) \tag{36}$$

Theorem 9.7. *Let $P(\vec{x} \mid \vec{\mu}, I)$ be a multivariate normal distribution. The maximum-likelihood estimate of $\vec{\mu}$ is the point that minimises the sum of squared Euclidean distances to the data points $X = \{\vec{x}_i \mid i = 1..n\}$.*

9.1.3 Ordinary least-squares regression

Theorem 9.8. *Let $\hat{y}(x) = \alpha + \beta x$ be a univariate linear regression model and $X = \{x_i \mid i = 1..n\}$, $Y = \{y_i \mid i = 1..n\}$ be a set of data points. If the noise is normally distributed, then the maximum-likelihood estimates of α and β are equivalent to the ordinary least-squares solution.*

Proof. Assume that y_i is a noisy observation of $\hat{y}(x_i)$, i.e., $y_i = \hat{y}(x_i) + \epsilon_i$. If the noise is normally distributed, then the likelihood of y_i is:

$$P(y_i \mid x_i, \alpha, \beta, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \hat{y}(x_i))^2}{2\sigma^2}\right) \tag{37}$$

Assume that ϵ_i and y_i are independent. Then, the joint likelihood of Y is the product of the likelihoods of y_i:

$$\begin{aligned} P(Y \mid X, \alpha, \beta, \sigma) &= \prod_{i=1}^{n} P(y_i \mid x_i, \alpha, \beta, \sigma) && (38) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \hat{y}(x_i))^2}{2\sigma^2}\right) && (39) \\ &= \frac{1}{(2\pi)^{n/2}\,\sigma^n} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \hat{y}(x_i))^2\right) && (40) \end{aligned}$$

Apply the negative logarithm and substitute $\hat{y}(x_i)$:

$$-\ln P(Y \mid X, \alpha, \beta, \sigma) = \frac{n}{2}\ln 2\pi + n \ln \sigma + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (\alpha + \beta x_i))^2 \tag{41}$$

The negative log likelihood is minimised when its partial derivatives with respect to α, β, and σ^2 are zero:

$$\begin{aligned} \frac{\partial}{\partial \alpha} - \ln P(Y \mid X, \alpha, \beta, \sigma) &= \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - (\alpha + \beta x_i)) = 0 \\ &\implies \sum_{i=1}^{n}(y_i - (\alpha + \beta x_i)) = 0 \end{aligned} \tag{42}$$

$$\begin{aligned} \frac{\partial}{\partial \beta} - \ln P(Y \mid X, \alpha, \beta, \sigma) &= \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - (\alpha + \beta x_i))x_i = 0 \\ &\implies \sum_{i=1}^{n}(y_i - (\alpha + \beta x_i))x_i = 0 \end{aligned} \tag{43}$$

$$\begin{aligned} \frac{\partial}{\partial \sigma^2} - \ln P(Y \mid X, \alpha, \beta, \sigma) &= \frac{n}{2\sigma^2} - \frac{1}{2\sigma^4}\sum_{i=1}^{n}(y_i - (\alpha + \beta x_i))^2 = 0 \\ &\implies \sum_{i=1}^{n}(y_i - (\alpha + \beta x_i))^2 = n\sigma^2 \end{aligned} \tag{44}$$

□

9.2 Naïve Bayes

In the context of classification, it is assumed that a distribution that models the data X depends on the class Y. The greater the differences between the distributions for the different classes, the better the model can discriminate between them. Several decision rules can be applied:

- Maximum likelihood (ML):

$$\hat{y} = \arg\max_{y} P(X = x \mid Y = y) \tag{45}$$

- Maximum a posteriori (MAP):

$$\hat{y} = \arg\max_{y} P(X = x \mid Y = y) P(Y = y) \tag{46}$$

- Recalibrated likelihood:

$$\hat{y} = \arg\max_{y} w_y P(X = x \mid Y = y) \tag{47}$$

ML and MAP are equivalent if the prior distribution of Y is uniform. The recalibrated likelihood generalises ML and MAP by a set of weights w_y. With uncalibrated probability estimates, the recalibrated likelihood is needed.

9.3 TODO

- Categorical variables
- Categorical probability distributions
- Naïve Bayes
- Logistic regression

References

Flach, Peter (Sept. 2012). *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. 1st ed. Cambridge University Press.