Distance-based Learning

Based on Raymond J. Mooney’s slides and
Peter Flach book

Distance-based Learning

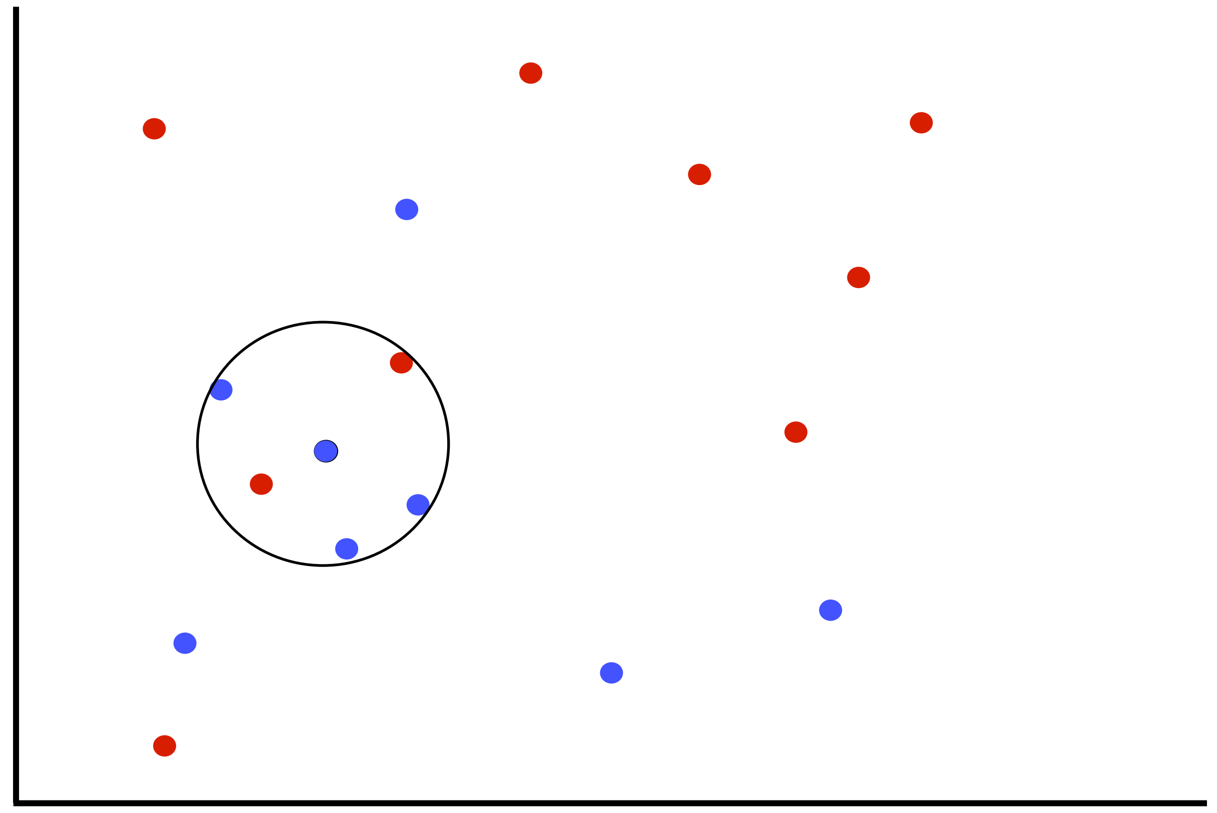
- Supervised: Instance-based learning (k-nearest neighbors)

- Unsupervised: Clustering

Instance-Based Learning

- Unlike other learning algorithms, does not involve construction of an explicit abstract generalization but classifies new instances based on direct comparison and similarity to known training instances.
- Training can be very easy, just memorizing training instances.
- Testing can be very expensive, requiring detailed comparison to all past training instances.
- Also known as:
  - Case-based
  - Exemplar-based
  - Nearest Neighbor
  - Memory-based
  - Lazy Learning

Example



Similarity/Distance Metrics

- Instance-based methods assume a function for determining the similarity or distance between any two instances.
- For continuous feature vectors, Euclidian distance is the generic choice:

$$d(x_i, x_j) = \sqrt{\sum_{p=1}^{n} (a_p(x_i) - a_p(x_j))^2}$$

Where $a_p(x)$ is the value of the p th feature of instance x.

- For discrete features, assume distance between two values is 0 if they are the same and 1 if they are different (e.g. Hamming distance for bit vectors).
- To compensate for difference in units across features, scale all continuous values to the interval [0,1].

Minkowski distance I

If $\mathscr{X} = \mathbb{R}^d$, the *Minkowski distance* of order $p > 0$ is defined as

$$\mathrm{Dis}_p(\mathbf{x},\mathbf{y}) = \left(\sum_{j=1}^{d} |x_j - y_j|^p \right)^{1/p} = ||\mathbf{x}-\mathbf{y}||_p$$

where $||\mathbf{z}||_p = \left(\sum_{j=1}^{d} |z_j|^p\right)^{1/p}$ is the *p-norm* (sometimes denoted L_p norm) of the vector $\mathbf{z}$. We will often refer to Dis_p simply as the p-norm.

☞ The *2-norm* refers to the familiar *Euclidean distance*

$$\mathrm{Dis}_2(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2} = \sqrt{(\mathbf{x}-\mathbf{y})^{\mathrm{T}}(\mathbf{x}-\mathbf{y})}$$

which measures distance ‘as the crow flies’.

Minkowski distance II

- The 1-*norm* denotes *Manhattan distance*, also called *cityblock distance*:

$$\mathrm{Dis}_1(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{d} |x_j - y_j|$$

  This is the distance if we can only travel along coordinate axes.

- If we now let p grow larger, the distance will be more and more dominated by the largest coordinate-wise distance, from which we can infer that $\mathrm{Dis}_\infty(\mathbf{x},\mathbf{y}) = \max_j |x_j - y_j|$; this is also called *Chebyshev distance*.

Minkowski distance III

- You will sometimes see references to the *0-norm* (or L_0 norm) which counts the number of non-zero elements in a vector. The corresponding distance then counts the number of positions in which vectors $\mathbf{x}$ and $\mathbf{y}$ differ. This is not strictly a Minkowski distance; however, we can define it as

$$\mathrm{Dis}_0(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{d} (x_j - y_j)^0 = \sum_{j=1}^{d} I[x_j = y_j]$$

  under the understanding that $x^0 = 0$ for $x = 0$ and 1 otherwise.

- If $\mathbf{x}$ and $\mathbf{y}$ are binary strings, this is also called the *Hamming distance*. Alternatively, we can see the Hamming distance as the number of bits that need to be flipped to change $\mathbf{x}$ into $\mathbf{y}$.

- For non-binary strings of unequal length this can be generalised to the notion of *edit distance* or *Levenshtein distance*.

Means and distances I

Theorem (The arithmetic mean minimises squared Euclidean distance)

The arithmetic mean $\boldsymbol{\mu}$ of a set of data points D in a Euclidean space is the unique point that minimises the sum of squared Euclidean distances to those data points.

Proof.

We will show that $\arg\min_{\mathbf{y}} \sum_{\mathbf{x} \in D} ||\mathbf{x}-\mathbf{y}||^2 = \boldsymbol{\mu}$, where $||\cdot||$ denotes the 2-norm. We find this minimum by taking the gradient (the vector of partial derivatives with respect to y_i) of the sum and setting it to the zero vector:

$$\nabla_{\mathbf{y}} \sum_{\mathbf{x} \in D} ||\mathbf{x}-\mathbf{y}||^2 = -2 \sum_{\mathbf{x} \in D} (\mathbf{x}-\mathbf{y}) = -2 \sum_{\mathbf{x} \in D} \mathbf{x} + 2|D|\mathbf{y} = \mathbf{0}$$

from which we derive $\mathbf{y} = \frac{1}{|D|} \sum_{\mathbf{x} \in D} \mathbf{x} = \boldsymbol{\mu}$. □

Means and distances II

- ☞ You may wonder what happens if we drop the square here: wouldn't it be more natural to take the point that minimises total Euclidean distance as exemplar?
- ☞ This point is known as the *geometric median*, as for univariate data it corresponds to the median or 'middle value' of a set of numbers. However, for multivariate data there is no closed-form expression for the geometric median, which needs to be calculated by successive approximation.
- ☞ In certain situations it makes sense to restrict an exemplar to be one of the given data points. In that case, we speak of a *medoid*, to distinguish it from a *centroid* which is an exemplar that doesn't have to occur in the data.
- ☞ Finding a medoid requires us to calculate, for each data point, the total distance to all other data points, in order to choose the point that minimises it. Regardless of the distance metric used, this is an $O(n^2)$ operation for n points.

The basic linear classifier is distance-based

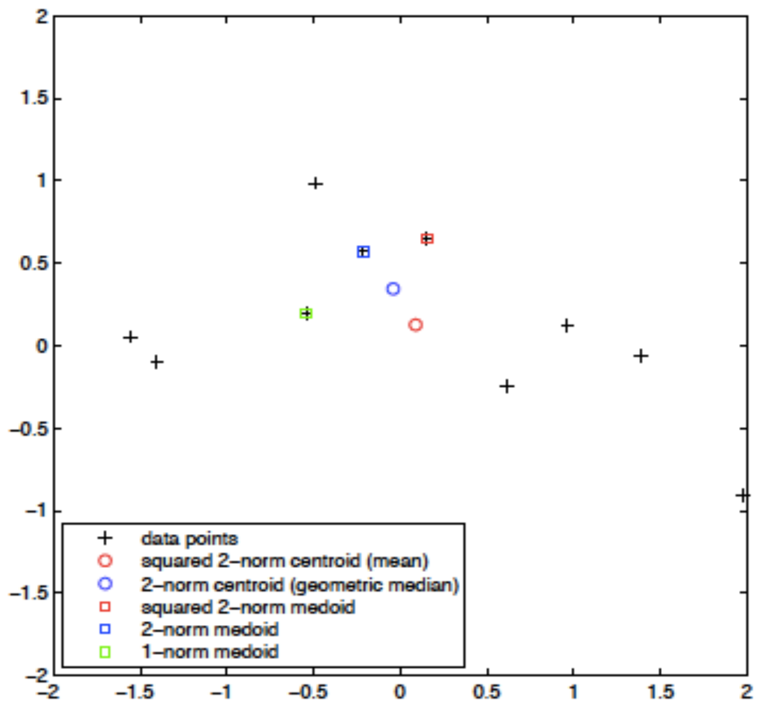
- The basic linear classifier constructs the decision boundary as the perpendicular bisector of the line segment connecting the two exemplars (one for each class).
- An alternative, distance-based way to classify instances without direct reference to a decision boundary is by the following decision rule: if $\mathbf{x}$ is nearest to $\boldsymbol{\mu}^{\oplus}$ then classify it as positive, otherwise as negative; or equivalently, classify an instance to the class of the *nearest* exemplar.
- If we use Euclidean distance as our closeness measure, simple geometry tells us we get exactly the same decision boundary (Figure 8.6 (left)).
- So the basic linear classifier can be interpreted from a distance-based perspective as constructing exemplars that minimise squared Euclidean distance within each class, and then applying a nearest-exemplar decision rule.

Other Distance Metrics

- Mahalanobis distance (→)
  - Scale-invariant metric that normalizes for variance.
- Cosine Similarity
  - Cosine of the angle between the two vectors.
  - Used in text and other high-dimensional data.
- Pearson correlation (→)
  - Standard statistical correlation coefficient.
- Edit distance
  - Used to measure distance between unbounded length strings.

Example: Centroids and medoids

Flach Fig. 8.5. p. 239



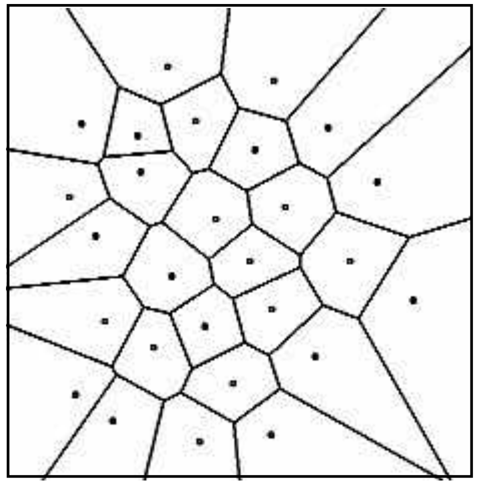
A small data set of 10 points, with circles indicating centroids and squares indicating medoids (the latter must be data points), for different distance metrics. Notice how the outlier on the bottom-right 'pulls' the mean away from the geometric median; as a result the corresponding medoid changes as well.

K-Nearest Neighbor

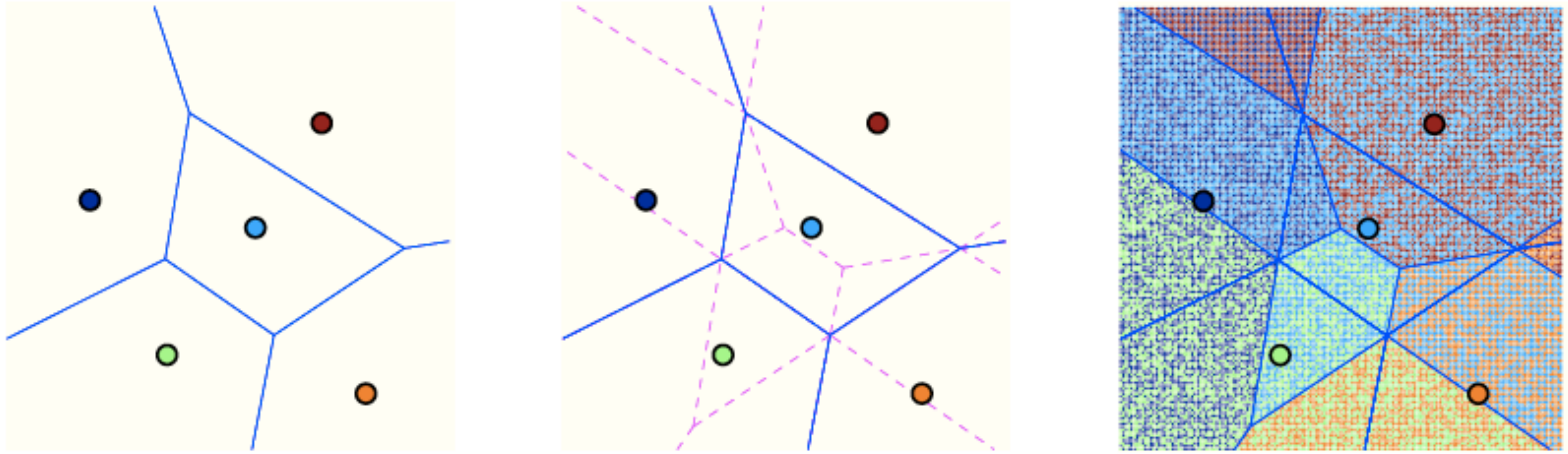
- Calculate the distance between a test point and every training instance.
- Pick the *k* closest training examples and assign the test instance to the most common category amongst these nearest neighbors.
- Voting multiple neighbors helps decrease susceptibility to noise.
- Usually use odd value for *k* to avoid ties.

Implicit Classification Function

- Although it is not necessary to explicitly calculate it, the learned classification rule is based on regions of the feature space closest to each training example.
- For 1-nearest neighbor with Euclidian distance, the **Voronoi diagram** gives the complex polyhedra segmenting the space into the regions closest to each point.



One vs. Two (and more) nearest neighbors



(left) Voronoi tesselation for five exemplars. **(middle)** Taking the two nearest exemplars into account leads to a further subdivision of each Voronoi cell. **(right)** The shading indicates which exemplars contribute to which cell.

Efficient Indexing

- Linear search to find the nearest neighbors is not efficient for large training sets.
- Indexing structures can be built to speed testing.
- For Euclidian distance, a **kd-tree** can be built that reduces the expected time to find the nearest neighbor to O(log n) in the number of training examples.
  - Nodes branch on threshold tests on individual features and leaves terminate at nearest neighbors.
- Other indexing structures possible for other metrics or string data.
  - Inverted index for text retrieval.

kd-tree

- The kd-tree is a binary tree in which every node is a k-dimensional point.
- Every non-leaf node generates a splitting hyperplane that divides the space into two subspaces.
- Points left to the hyperplane represent the left sub-tree of that node and the points right to the hyperplane by the right sub-tree.
- The hyperplane direction is chosen in the following way: every node split to sub-trees is associated with one of the k-dimensions, such that the hyperplane is perpendicular to that dimension vector.

Nearest Neighbor Variations

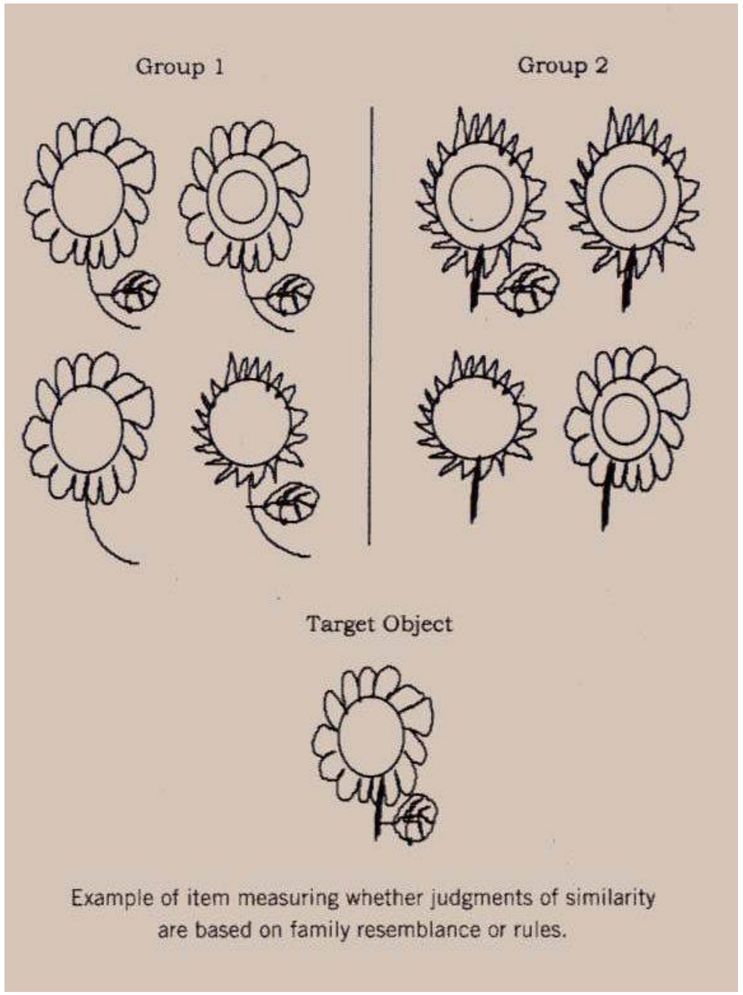
- Can be used to estimate the value of a real-valued function – regression - by taking the average function value of the *k* nearest neighbors to an input point.
- All training examples can be used to help classify a test instance by giving every training example a vote that is weighted by the inverse square of its distance from the test instance.

Feature Relevance and Weighting

- Standard distance metrics weight each feature equally when determining similarity.
  - Problematic if many features are irrelevant, since similarity along many irrelevant examples could mislead the classification.
- Features can be weighted by some measure that indicates their ability to discriminate the category of an example, such as information gain.
- Overall, instance-based methods favor global similarity over concept simplicity.

Rules and Instances in Human Learning Biases

- Psychological experiments show that people from different cultures exhibit distinct categorization biases.
- "Western" subjects favor simple rules (straight stem) and classify the target object in group 2.
- "Asian" subjects favor global similarity and classify the target object in group 1.



Example of item measuring whether judgments of similarity are based on family resemblance or rules.

Other Issues

- Can reduce storage of training instances to a small set of representative examples.
  - Support vectors in an SVM are somewhat analogous.
- Can hybridize with rule-based methods or neural-net methods.
  - Radial basis functions in neural nets and Gaussian kernels in SVMs are similar.
- Can be used for more complex relational or graph data.
  - Similarity computation is complex since it involves some sort of graph isomorphism.
- Can be used in problems other than classification.
  - Case-based planning
  - Case-based reasoning in law and business.

Conclusions

- IBL methods classify test instances based on similarity to specific training instances rather than forming explicit generalizations.
- Typically trade decreased training time for increased testing time.