K-nearest neighbor classification

K-nearest neighbors

- Training method:
  - Save the training examples (no sophisticated learning)
- At prediction (testing) time:
  - Given a test (query) example **x**, find the *K* training examples that are *closest* to **x**.

$$\mathrm{KNN}(\mathbf{x}) = \left\{\left(\mathbf{x}^{(1)\prime}, y^{(1)\prime}\right), \left(\mathbf{x}^{(2)\prime}, y^{(2)\prime}\right), ..., \left(\mathbf{x}^{(K)\prime}, y^{(K)\prime}\right)\right\}$$

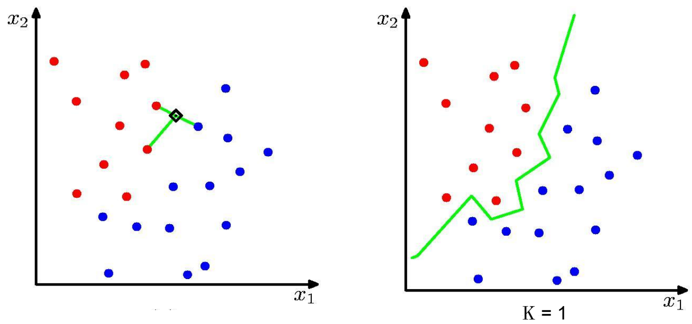
- Predict the most frequent class among all *y'*s from *KNN*(**x**).

$$h(\mathbf{x}) = \arg\max_y \sum_{(\mathbf{x}', y') \in \mathrm{kNN}(\mathbf{x})} \mathbb{I}[y' = y]$$ "majority vote"

- <u>Note: this function can be applied to regression!</u>

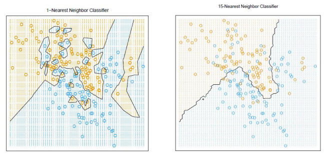
Slide credit: William Cohen

K-nearest neighbors for classification



Slide credit: Ben Kuipers

K-nearest neighbors for classification



- Larger *K* leads to a smoother decision boundary (bias-variance trade-off)
- Classification performance generally improves as *N* (training set size) increases
- For *N* ⇒ ∞, the error rate of the 1-nearest-neighbor classifier is never more than twice the optimal error (obtained from the true conditional class distributions). See ESL CH 13.3.

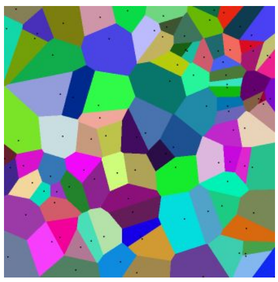
Slide credit: Ben Kuipers

Factors (hyperparameters) affecting KNN

- Distance metric *D*(**x**, **x'**)
  - How to define distance between two examples **x** and **x'**?
- The value of *K*
  - *K* determines how much we "smooth out" the prediction

What is the decision boundary?

Voronoi diagram: Euclidean (L_2) distance



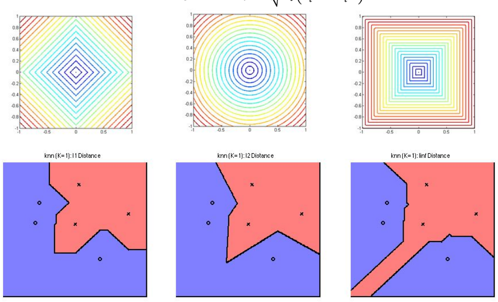
Note: Each region corresponds kNN's prediction when K=1

i.e. prediction is the same as the corresponding training sample's label in each region (training sample is visualized as dot).

Slide credit: William Cohen

Dependence on distance metric (L^q norm)

Distance between i-th and j-th example: $\sqrt[q]{\sum_l \left(x_l^{(i)} - x_l^{(j)}\right)^q}$



Slide credit: Ben Taskar

KNN: classification vs regression

- We can formulate KNN into regression/classification
- For classification, where the label *y* is categorical, we take the "majority vote" over target labels.

$$h(\mathbf{x}) = \arg\max_y \sum_{(\mathbf{x}', y') \in \mathrm{KNN}(\mathbf{x})} \mathbb{I}[y' = y]$$

- For regression, where the label *y* is real-valued numbers, we take "average" over target labels.

$$h(\mathbf{x}) = \frac{1}{k} \sum_{(\mathbf{x}', y') \in \mathrm{KNN}(\mathbf{x})} y'$$

Advantage/disadvantages of KNN methods

- Advantage:
  - Very simple and flexible (no assumption on distribution)
  - Effective (e.g. for low dimensional inputs)
- Disadvantages:
  - Expensive: need to remember (store) and search through all the training data for every prediction
  - Curse of dimensionality: in high dimensions, all points are far
  - Not robust to irrelevant features: if **x** has irrelevant/noisy features, then distance function does not reflect similarity between examples

Concept check

- How are labels represented in multiclass classification problems?
- What is the motivation for using Newton's method for optimization in logistic regression?
- What does increasing K do for the results from KNN?