Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 5

Slides adapted from Chenhao Tan, Chris Ketelsen, Jordan Boyd-Graber

Learning objectives

- Revisiting bias-variance tradeoff
- Understand K-nearest neighbor classifiers

Outline

Bias-variance tradeoff

K-nearest neighbors

- Overview
- Weighted KNN

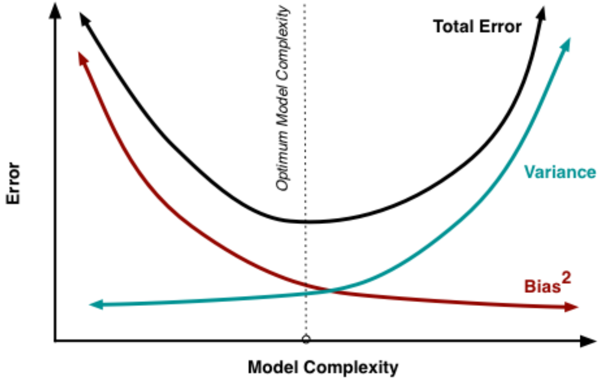
Outline

Bias-variance tradeoff

K-nearest neighbors
- Overview
- Weighted KNN

Bias-Variance Tradeoff

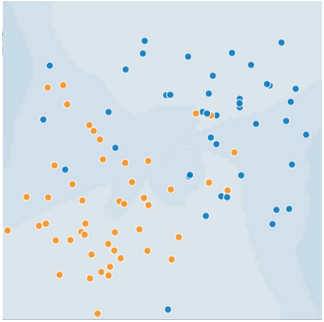
$$\text{Generailization error} \quad = \quad \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}$$



http://scott.fortmann-roe.com/docs/BiasVariance.html

Overfitting Example

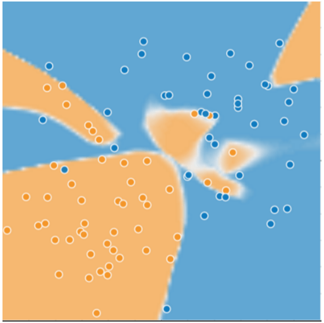
Assume a binary classification model, classifies orange vs. blue



Training examples are given above

Overfitting Example

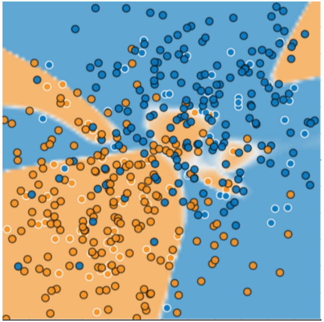
Assume a binary classification model, classifies orange vs. blue



Is this a “good” model?

Overfitting Example

Assume a binary classification model, classifies orange vs. blue



Let's see by adding unseen examples

Bias-Variance Tradeoff

Assume

- a simple model $y = f(x) + \epsilon$,
- noise ϵ is $\mathrm{E}(\epsilon) = 0, \mathrm{Var}(\epsilon) = \sigma_\epsilon^2$,
- use squared error $(y - h(x))^2$ as the loss function,
- h is a function learned by the model from training samples

$$
\begin{aligned}
\epsilon_{\text{general}}(x) &= \mathrm{E}[(y - h(x))^2] \\
&= \sigma_\epsilon^2 + [\mathrm{E}h(x) - f(x)]^2 + \mathrm{E}[h(x) - \mathrm{E}h(x)]^2 \\
&= \sigma_\epsilon^2 + \mathrm{Bias}^2(h(x)) + \mathrm{Var}(h(x)) \\
&= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}
\end{aligned}
$$

Outline

K-nearest neighbors (K-NN)

- Suppose we have an input data x that we want to classify.
- Look in training set for K examples that are **nearest** to x
- Classify x by the label of those K points.

Question: What does **nearest** mean?

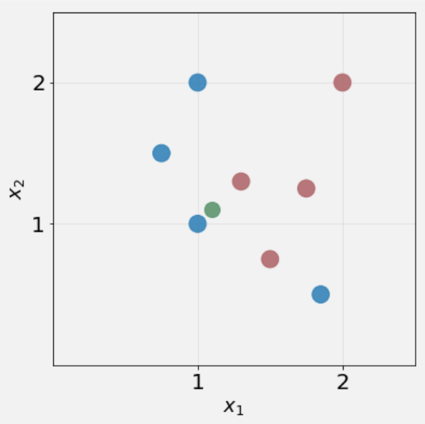
K-nearest neighbors (K-NN)

- Suppose we have an input data x that we want to classify.
- Look in training set for K examples that are **nearest** to x
- Classify x by the label of those K points.

Question: What does **nearest** mean?

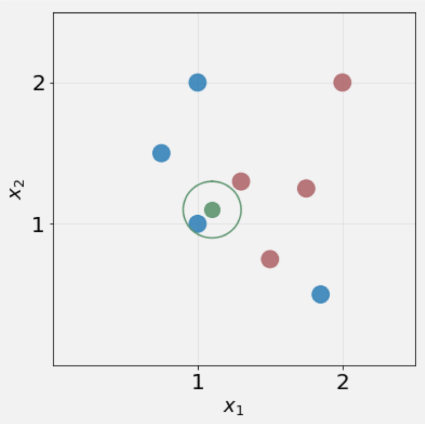
One answer: Euclidean distance $||\vec{x}_1 - \vec{x}_2||_2$

Example: Will the Green Point Classified as Blue or Red?



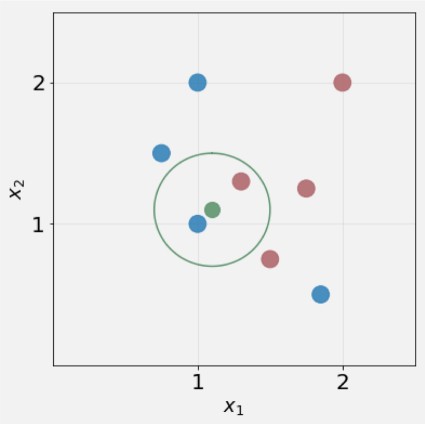
- 1-NN predicts:
- 2-NN predicts:
- 5-NN predicts:

Example: Will the Green Point Classified as Blue or Red?



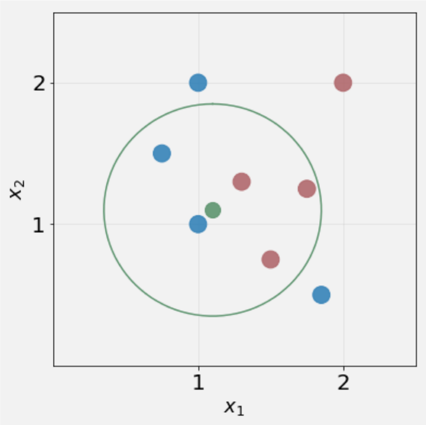
- 1-NN predicts: blue
- 2-NN predicts:
- 5-NN predicts:

Example: Will the Green Point Classified as Blue or Red?



- 1-NN predicts: blue
- 2-NN predicts: unclear
- 5-NN predicts:

Example: Will the Green Point Classified as Blue or Red?



- 1-NN predicts: blue
- 2-NN predicts: unclear
- 5-NN predicts: red

K-nearest neighbors

Find the K-nearest neighbors of x in training data and predict the majority label of those K points.

K-nearest neighbors

Find the K-nearest neighbors of x in training data and predict the majority label of those K points.

$$h(x) = \arg \max_{y \in \{+1,-1\}} \sum_{(x',y') \in \mathrm{NN}(\mathrm{x}, \mathrm{S}_{\mathrm{train}}, \mathrm{k})} I(y = y')$$

where

- I is the identity function i.e., $I(y = y') = 1$, $I(y \neq y') = 0$,
- $NN(x, S_{\text{train}}, k)$ is the k nearest neighbors of x in the training data S_{train}

K-nearest neighbors

Find the K-nearest neighbors of x in training data and predict the majority label of those K points.

$$h(x) = \arg \max_{y \in \{+1,-1\}} \sum_{(x',y') \in \mathrm{NN}(\mathrm{x}, \mathrm{S}_{\mathrm{train}}, \mathrm{k})} I(y = y')$$

where

- I is the identity function i.e., $I(y = y') = 1$, $I(y \neq y') = 0$,
- $NN(x, S_{\mathrm{train}}, k)$ is the k nearest neighbors of x in the training data S_{train}

Assumptions in the algorithm: nearby instances share similar labels.

Hyperparameters in K-NN

- Distance function d

Hyperparameters in K-NN

- Distance function d

Discrete

$$d(x_1, x_2) = 1 - \frac{|x_1 \cap x_2|}{|x_1 \cup x_2|} \quad (1)$$

i.e., $1-$ fraction of number of shared features

Continuous

Euclidean distance

$$d(x_1, x_2) = \|\vec{x}_1 - \vec{x}_2\|_2 \quad (2)$$

Hyperparameters in K-NN

- Distance function d

Discrete

$$d(x_1, x_2) = 1 - \frac{|x_1 \cap x_2|}{|x_1 \cup x_2|} \tag{1}$$

i.e., $1-$ fraction of number of shared features

Continuous

Euclidean distance

$$d(x_1, x_2) = \|\vec{x}_1 - \vec{x}_2\|_2 \tag{2}$$

Manhattan distance

$$d(x_1, x_2) = \|\vec{x}_1 - \vec{x}_2\|_1 \tag{3}$$

Hyperparameters in K-NN

- Distance function d

Discrete

$$d(x_1, x_2) = 1 - \frac{|x_1 \cap x_2|}{|x_1 \cup x_2|} \tag{1}$$

i.e., $1-$ fraction of number of shared features

Continuous

Euclidean distance

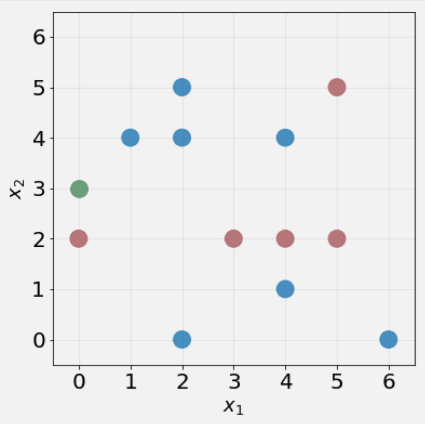
$$d(x_1, x_2) = \|\vec{x}_1 - \vec{x}_2\|_2 \tag{2}$$

Manhattan distance

$$d(x_1, x_2) = \|\vec{x}_1 - \vec{x}_2\|_1 \tag{3}$$

- Number of nearest neighbors k

Example of Eucledian vs. Manhattan Distance



Euclidean distance:

$$\|\vec{x}_1 - \vec{x}_2\|_2 = \sqrt{(x_{1,1} - x_{2,1})^2 + (x_{1,2} - x_{2,2})^2}$$

Manhattan distance:

$$\|\vec{x}_1 - \vec{x}_2\|_1 = |x_{1,1} - x_{2,1}| + |x_{1,2} - x_{2,2}|$$

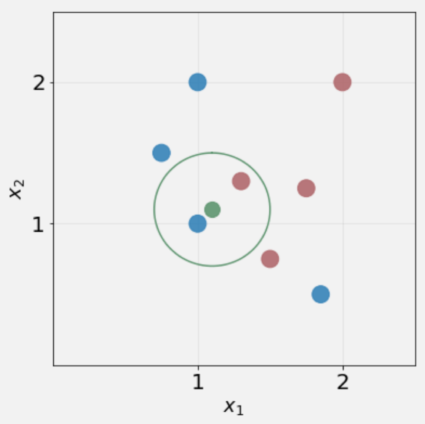
Example: Distance between the green and its closest blue point

Decision tree vs. KNN

- Decision tree uses one feature at a time, and splits the data to reduce entropy.
- KNN takes a geometry perspective and quickly finds the closest points in the training data.

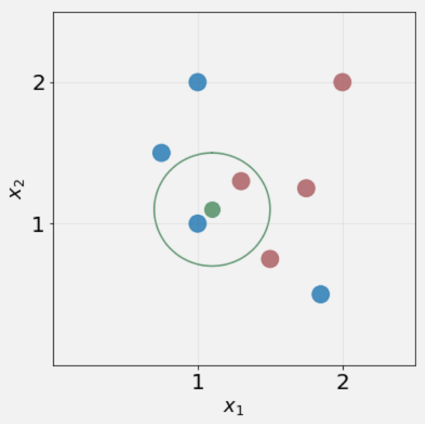
KNN Classification

What about ties?



KNN Classification

What about ties?

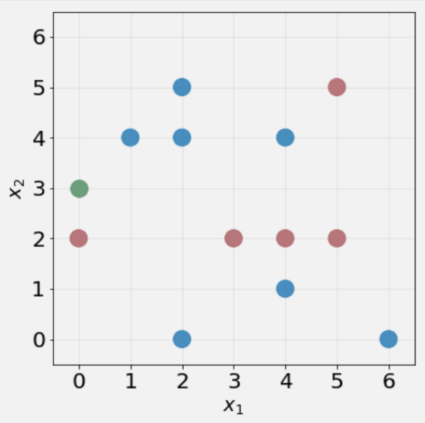


One reasonable strategy:

- try classifying with $k-1$
- repeat for $k-2, \ldots$ until the tie is broken

KNN Classification

Another example

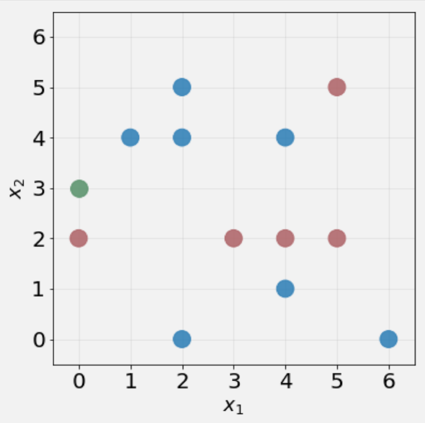


Euclidean distance: $\|\vec{x}_1 - \vec{x}_2\|_2$

$k = 1$

KNN Classification

Another example

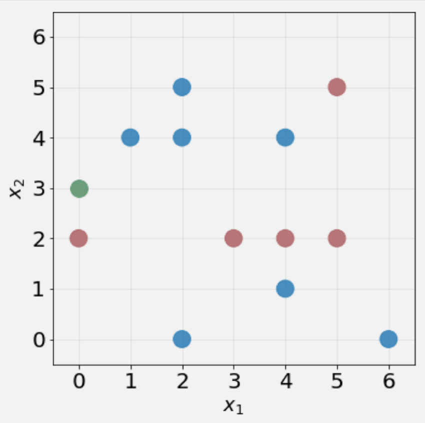


Euclidean distance: $||\vec{x}_1 - \vec{x}_2||_2$

$k = 2$

KNN Classification

Another example

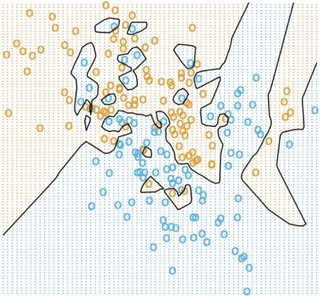


Euclidean distance: $\|\vec{x}_1 - \vec{x}_2\|_2$

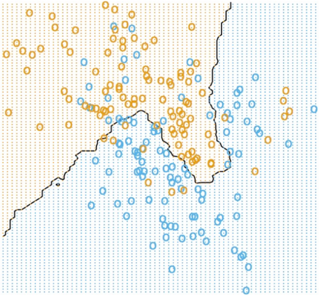
$k = 3$

K-nearest neighbors

Recall the danger of overfitting



$k = 1$

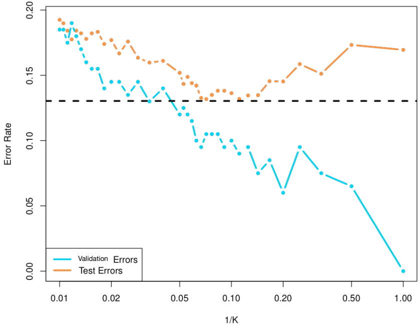


$k = 15$

K-nearest neighbors

Choose optimal k by varying k and computing error rate on the development set.

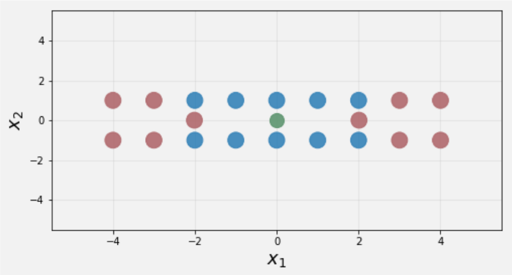
- x-axis: $\frac{1}{k}$ (degree of freedom)
- Lower k leads to more flexibility



Feature Scaling

Practical tips:

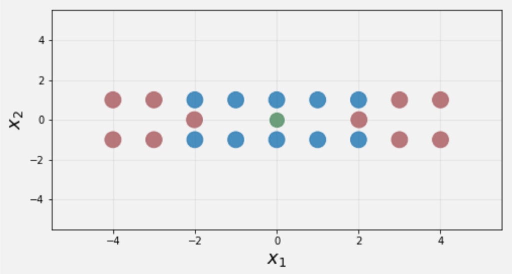
- Important to normalize features to similar sizes
- Consider doing 2-NN on unscaled and scaled data

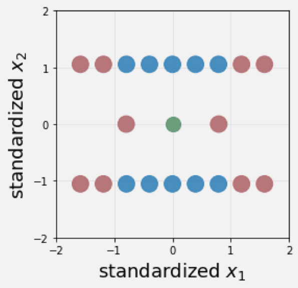


Feature Scaling

Practical tips:

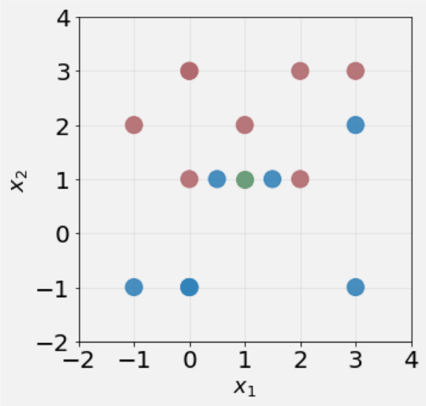
- Important to normalize features to similar sizes
- Consider doing 2-NN on unscaled and scaled data





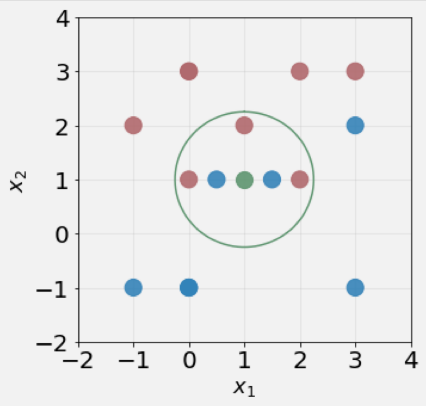
Weighted KNN

What should 5-NN predict in the following figure?



Weighted KNN

What should 5-NN predict in the following figure?



Weighted KNN

Weighted-KNN:

$$h(x) = \arg\max_{y \in \{+1,-1\}} \sum_{(x',y') \in \mathrm{NN}(\mathrm{x}, \mathrm{S}_{\mathrm{train}}, \mathrm{k})} \frac{1}{d(x,x')} I(y = y')$$

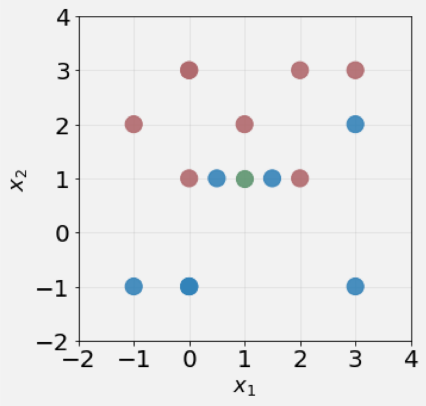
- Find $\mathrm{NN}(\mathrm{x}, \mathrm{S}_{\mathrm{train}}, \mathrm{k})$: the set of K training examples nearest to
- Predict $\hat{y}$ to be weighted-majority label in $\mathrm{NN}(\mathrm{x}, \mathrm{S}_{\mathrm{train}}, \mathrm{k})$, weighted by **inverse-distance**

Improvements over KNN:

- Gives more weight to examples that are very close to query point
- Less tie-breaking required

Weighted KNN

What should 5-NN predict in the following figure?



- Red distance:
- Blue distance:
- Red weighted-Majority vote:
- Blue weighted-majority vote:
- Prediction:

Memory and Efficiency of the naive implementation

How does the algorithm scale?

Memory and Efficiency of the naive implementation

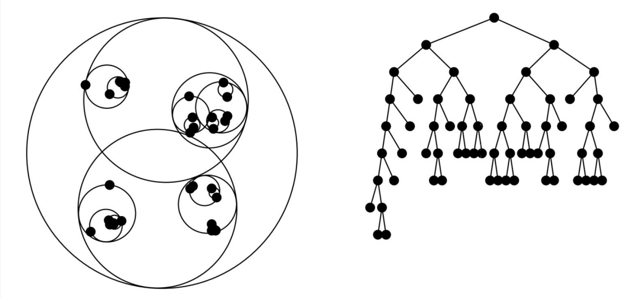
How does the algorithm scale?
When N training examples with d features, the naive implementation takes

- memory: $O(Nd)$
- time: $O(Nd)$

to classify (loop through each example, compute distance)

Better Implementation

Use Ball tree or KD tree to ignore examples farther than the seen ones



- memory: $O(Nd)$
- time: $O(\log(N)d)$

Summary

- Show bias-variance tradeoff using a simple model
- Learned about KNN and weighted KNN

References

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.