CS489/698
Lecture 2: January 9th, 2017

Nearest Neighbour
[RN] Sec. 18.8.1, [HTF] Sec. 2.3.2,
[D] Chapt. 3, [B] Sec. 2.5.2,
[M] Sec. 1.4.2

Inductive Learning (recap)

- Induction
  - Given a training set of examples of the form $(x, f(x))$
    - x is the input, $f(x)$ is the output
  - Return a function h that approximates f
    - h is called the hypothesis

Supervised Learning

- Two types of problems
  1. **Classification:**

  2. **Regression**

- NB: The nature (categorical or continuous) of the domain (input space) of f does not matter

Classification Example

- Problem: Will you enjoy an outdoor sport based on the weather?
- Training set:

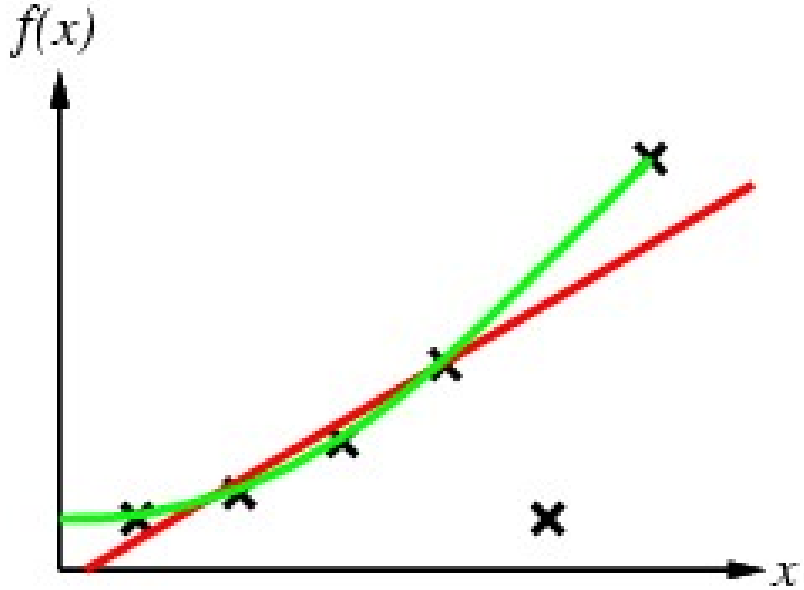
Sky	**Humidity**	**Wind**	**Water**	**Forecast**	**EnjoySport**
Sunny	Normal	Strong	Warm	Same	yes
Sunny	High	Strong	Warm	Same	yes
Sunny	High	Strong	Warm	Change	no
Sunny	High	Strong	Cool	Change	yes

x (Sky … Forecast) $\quad$ $f(x)$ (EnjoySport)

- Possible Hypotheses:
  - $h_1 : S = sunny \rightarrow enjoySport = yes$
  - $h_2 : Wa = cool$ or $F = same \rightarrow enjoySport = yes$

Regression Example

- Find function h that fits f at instances x



More Examples

Problem	**Domain**	**Range**	**Classification / Regression**
Spam Detection			
Stock price prediction			
Speech recognition			
Digit recognition			
Housing valuation			
Weather prediction			

Hypothesis Space

- Hypothesis space H
  - Set of all hypotheses h that the learner may consider
  - Learning is a search through hypothesis space

- Objective: find h that minimizes
  - Misclassification
  - Or more generally some error function

  with respect to the training examples

- But what about unseen examples?

Generalization

- A good hypothesis will generalize well
  - i.e., predict unseen examples correctly
- Usually …
  - Any hypothesis h found to approximate the target function f well over a **sufficiently large set of training examples** will also approximate the target function well over any unobserved examples

Inductive Learning

- Goal: find an h that agrees with f on training set
  - h is **consistent** if it agrees with f on all examples

- Finding a consistent hypothesis is not always possible
  - Insufficient hypothesis space:
    - E.g., it is not possible to learn exactly $f(x) = ax + b + xsin(x)$ when $H =$ space of polynomials of finite degree
  - Noisy data
    - E.g., in weather prediction, identical conditions may lead to rainy and sunny days

Inductive Learning

- A learning problem is **realizable** if the hypothesis space contains the true function otherwise it is **unrealizable**.
  - Difficult to determine whether a learning problem is realizable since the true function is not known
- It is possible to use a very large hypothesis space
  - For example: H = class of all Turing machines
- But there is a **tradeoff** between **expressiveness** of a hypothesis class and the **complexity** of finding a good hypothesis

Nearest Neighbour Classification

- Classification function

$$h(x) = y_{x^*}$$

where y_{x^*} is the label associated with the nearest neighbour

$$x^* = argmin_{x'}\, d(x, x')$$

- Distance measures: $d(x, x')$

L_1: $d(x, x') = \sum_j^M |x_j - x_j'|$

L_2: $d(x, x') = \left(\sum_j^M |x_j - x_j'|^2\right)^{1/2}$

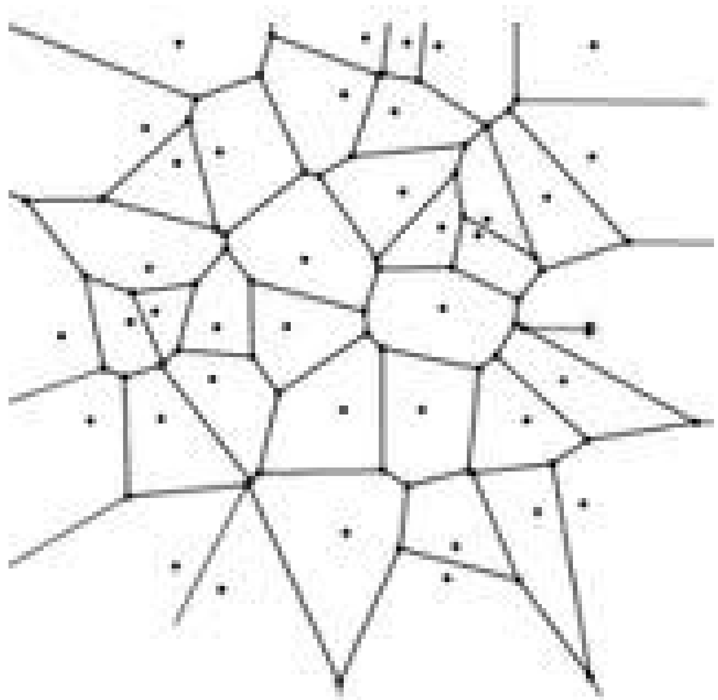
…

L_p: $d(x, x') = \left(\sum_j^M |x_j - x_j'|^p\right)^{1/p}$

Weighted dimensions: $d(x, x') = \left(\sum_j^M c_j |x_j - x_j'|^p\right)^{1/p}$

Voronoi Diagram

- Partition implied by nearest neighbor fn h
  - Assuming Euclidean distance

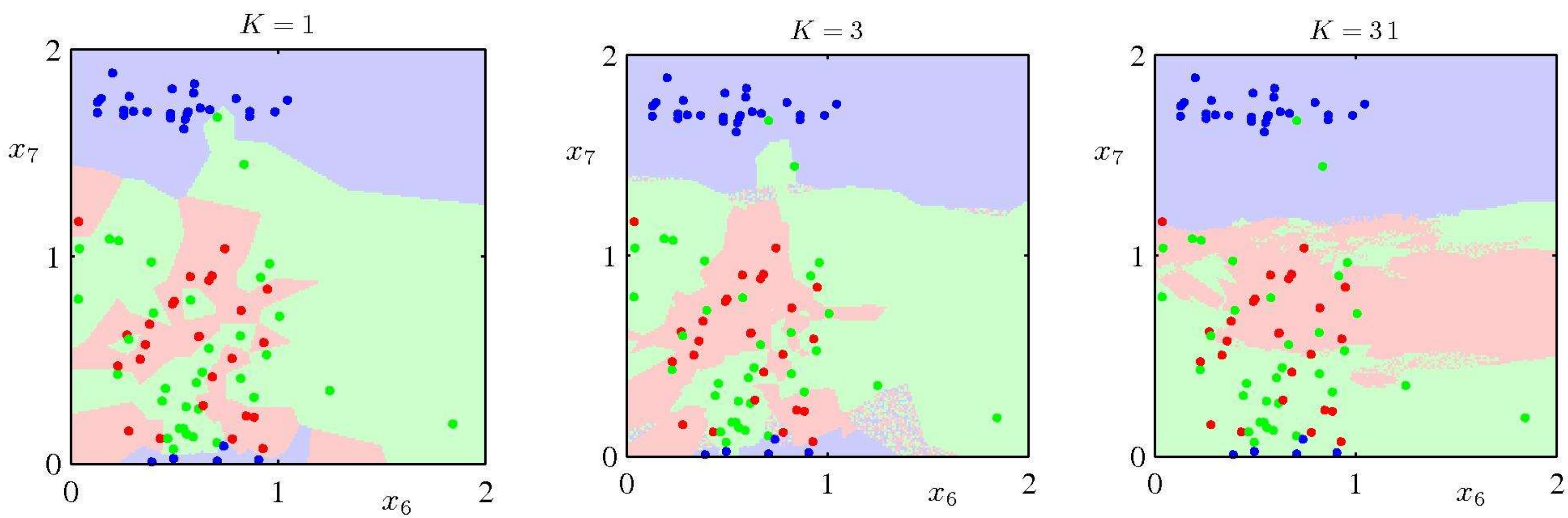


K-Nearest Neighbour

- Nearest neighbour often instable (noise)
- Idea: assign most frequent label among k-nearest neighbours
  - Let $knn(x)$ be the k-nearest neighbours of x according to distance d
  - Label: $y_x \leftarrow mode(\{y_{x'} | x' \in knn(x)\})$

Effect of K

- K controls the degree of smoothing.
- Which partition do you prefer? Why?

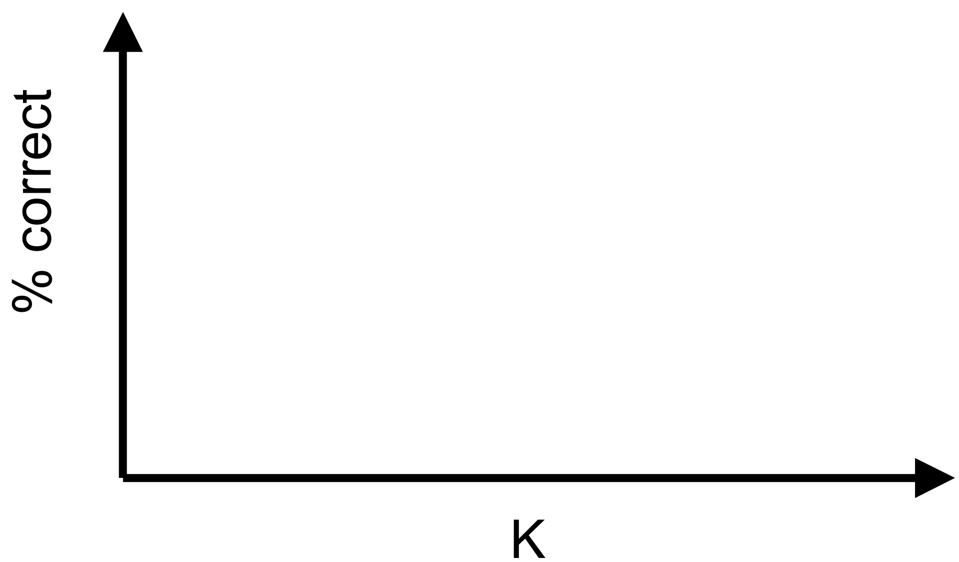


Performance of a learning algorithm

- A learning algorithm is good if it produces a hypothesis that does a good job of predicting classifications of unseen examples
- Verify performance with a **test set**
  1. Collect a large set of examples
  2. Divide into 2 disjoint sets: training set and test set
  3. Learn hypothesis h with training set
  4. Measure percentage of correctly classified examples by h in the test set

The effect of K

- Best K depends on
  - Problem
  - Amount of training data



Underfitting

- **Definition:** underfitting occurs when an algorithm finds a hypothesis h with training accuracy that is lower than the future accuracy of some other hypothesis h'

- Amount of underfitting of h:
$$max\ \{0, \max_{h'} futureAccuracy(h') - trainAccuracy(h)\}$$
$$\approx max\ \{0, \max_{h'} testAccuracy(h') - trainAccuracy(h)\}$$

- Common cause:
  - Classifier is not expressive enough

Overfitting

- **Definition:** overfitting occurs when an algorithm finds a hypothesis h with higher training accuracy than its future accuracy.

- Amount of overfitting of h:

$$\max\ \{0, trainAccuracy(h) - futureAccuracy(h)\}$$
$$\approx \max\ \{0, trainAccuracy(h) - testAccuracy(h)\}$$

- Common causes:
  – Classifier is too expressive
  – Noisy data
  – Lack of data

Choosing K

- How should we choose K?
  - Ideally: select K with highest future accuracy
  - In practice: select K with highest test accuracy

- Need to ensure that test accuracy is close to future accuracy
  - Test accuracy becomes more reliable as we increase the size of the test set
  - However, this reduces the amount of data left for training

- Popular solution: cross-validation

Cross-Validation

- Repeatedly split data in two parts, one for training, one for testing the accuracy of a hypothesis and report the average accuracy.
- ***k*-fold cross validation**: split data in k equal size subsets. Run k experiments, each time testing on one subset the hypothesis trained on the remaining subsets. Compute the average accuracy of the k experiments.
- Picture:

Selecting the Number of Neighbours by Cross-Validation

Let k be the number of neighbours

Let k' be the number of data splits

For $k = 1$ to max # of neighbours

  For $i = 1$ to k' do

    $h_{ki} \leftarrow train(k, data_{1..i-1,i+1..k'})$

    $acc_{ki} \leftarrow test(h_{ki}, data_i)$

  $acc_k \leftarrow average(\{acc_{ki}\}_{\forall i})$

$k^* \leftarrow argmax_k\ acc_k$

$h_{k^*} \leftarrow train(k^*, data_{1..k'})$

Return k^*, h_{k^*}, acc_{k^*}

Choosing a Hypothesis

- After determining the optimal number of neighbours k^* by cross validation, what hypothesis should we return?

- We could return any of the h_{ki} hypotheses, but we can do better.

- Instead return a new hypothesis h_{k^*} trained with all the data
  - The future accuracy of h_{k^*} is likely to be better than the future accuracy of each h_{k^*i} since h_{k^*} is trained with all the data
  - acc_{k^*} (average test accuracy of each h_{k^*i}) is a good (conservative) estimate of the future accuracy of h_{k^*}

Weighted K-Nearest Neighbour

- We can often improve K-nearest neighbours by weighting each neighbour based on some distance measure

$$w(x, x') \propto \frac{1}{distance(x, x')}$$

- Label

$$y_x \leftarrow argmax_y \sum_{\{x' | x' \in knn(x) \wedge y = y_{x'}\}} w(x, x')$$

K-Nearest Neighbour Regression

- We can also use knn for regression
- Let y_x be a real value instead of a categorical label

- K-nearest neighbour regression:

$$y_x \leftarrow average(\{y_{x'} | x' \in knn(x)\})$$

- Weighted K-nearest neighbour regression:

$$y_x \leftarrow \sum_{x' \in knn(x)} w(x, x') y_{x'}$$