Too many features can be harmful

- kNN distance function and classification dominated by extra features
- In general with many random features we are likely to get correlations by chance and this leads to overfitting
- Solutions:
  - Instance transformation
  - Feature selection

Too many features

- When working on an application it is tempting to identify and record more and more features that may be relevant for classification
- In some applications (e.g., text) there is a huge set of potential features (e.g., in bag of words representation)
- But too many features can be harmful
- Why?

Principal Component Analysis

- PCA is a standard method for instance transformation
- It is "unsupervised" in that it looks only at example locations and not their labels
- Intuitively: project the data onto k dimensions with highest variation
- [we will not cover technical details]

Instance Transformation

- Sometimes the data is given in a high dim space but has some hidden structure
- Unveiling this structure enables data analysis

Manifold Methods

- Data resides on a "manifold"
  - Embed data in low dim space, while preserving local distances
  - [we will not cover technical details]
**Feature Selection**

- **Supervised methods** make use of labels to identify useful features.
- **How?**
  - Filter methods
  - Wrapper methods

**Relief Algorithm**

- We have already discussed Relief in the context of kNN
- Pick instance + nearest hit + nearest miss
- Update weights based on distances and whether hit is closer/farther than miss
  \[ ||x - y||^2_w = \sum w_i (x_i - y_i)^2 \]
- Weighting scheme not strictly selecting features. But small/zero weight can be interpreted as irrelevant

**Filter Methods**

- Pick a criterion for evaluating features
  - InfoGain
  - Correlation with label
    \[ \rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \]
  - Mutual information with label
    \[ I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \]

**Filter Methods**

- What happens if we duplicate the top ranking feature?
- Are single feature tests sufficient?
**Wrapper Method**

- Eval subset by running and testing alg, e.g., using validation set
- Search adapts to the learning algorithm!
- Incremental search over subsets
- Forward, backward, combined, beam, ...

**Advanced Methods**

- In the context of “machine learning as optimization” the $L_1$ penalty is known to effectively induce sparsity in the results (similar to the weighting in Relief)
- To be discussed later in the course

**Normalizing Feature Range**

- Range of values of a feature can affect performance especially for distance based methods
- e.g. one feature can dominate distance

**Normalizing Feature Range**

- Linear scaling into [0,1]
  \[ x' = \frac{x - x_{min}}{x_{max} - x_{min}} \]

- Scale to have mean 0 and std 1
  \[ x' = \frac{x - \mu_x}{\sigma_x} \]
Discretizing Features

- What if we want to use a discrete-only learning algorithm but have data with real valued features?
- Preprocess to discretize feature
- How?

Unsupervised methods:
- Equal bin size (a fixed grid)
  - Predetermined by feature range
  - Pros and cons?
- Equal frequency
  - Divide to get same number of points in each bin
  - Adapt to data
  - Pros and cons?

Supervised methods:
- Looks for partitions that help identify label
  Example: run DT algorithm on single feature and include all node splits from the tree.
  This is most helpful after pruning as otherwise partitions too specific and we get no generalization.

How should we present the discrete values to the learning algorithm?
- Say \( x_i \) split into intervals 
  \([-100,-1],[-1,1],[1,7]\)
  and we have the value 0.7?
- Two dominant approaches:
  - As unit vectors
  - As "increasing weight" vectors
  1000,1100,1110,1111

Features/Representation Summary

- Supervised vs unsupervised
- Instance transformation methods
- Filter Methods
- Wrapper methods
- Algorithm specific selection methods
- Feature normalization
- Feature discretization