CSCI 5622 Machine Learning

ML Instance-based Learning

DATE	READ	DUE
Today, Sept 23	8	Lit Review Plan
Mon, Sept 28	TBA	Peer Fdbk Grades
Wed, Sept 30	TBA	Notes Papers 1&2

www.RodneyNielsen.com/teaching/CSCI5622-F09/

Instructor: Rodney Nielsen

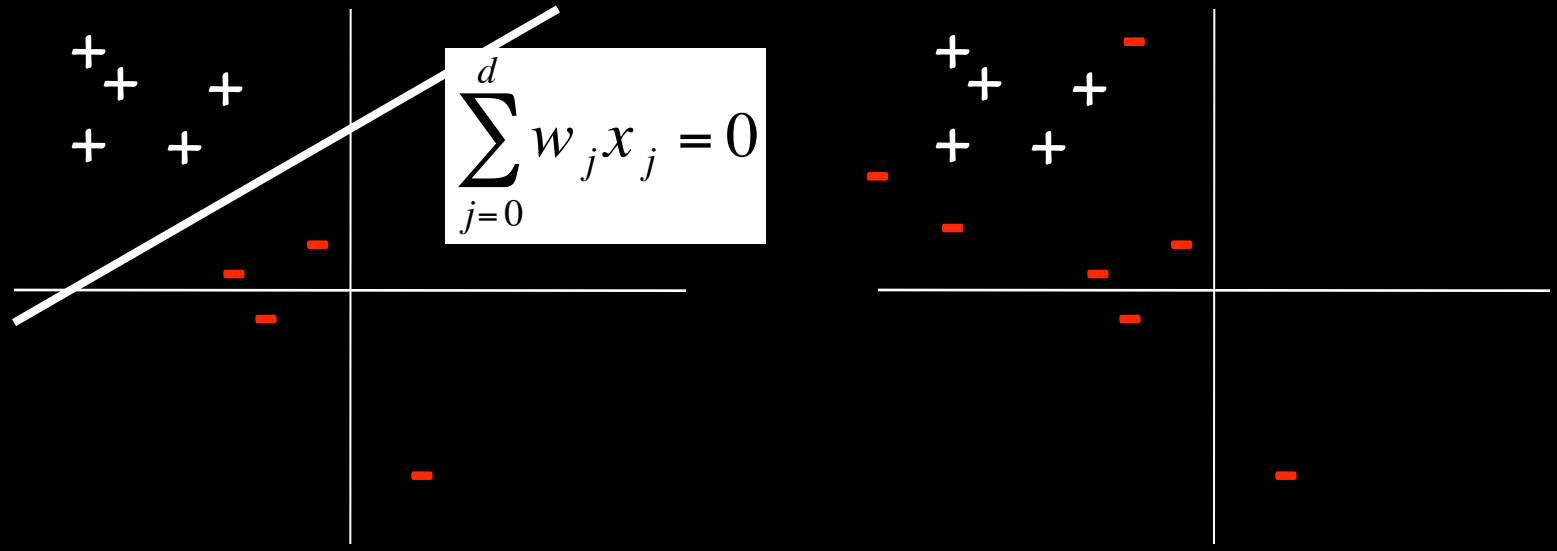
Assistant Professor Adjunct, CU Dept. of Computer Science

Research Assistant Professor, DU, Dept. of Electrical & Computer Engr.

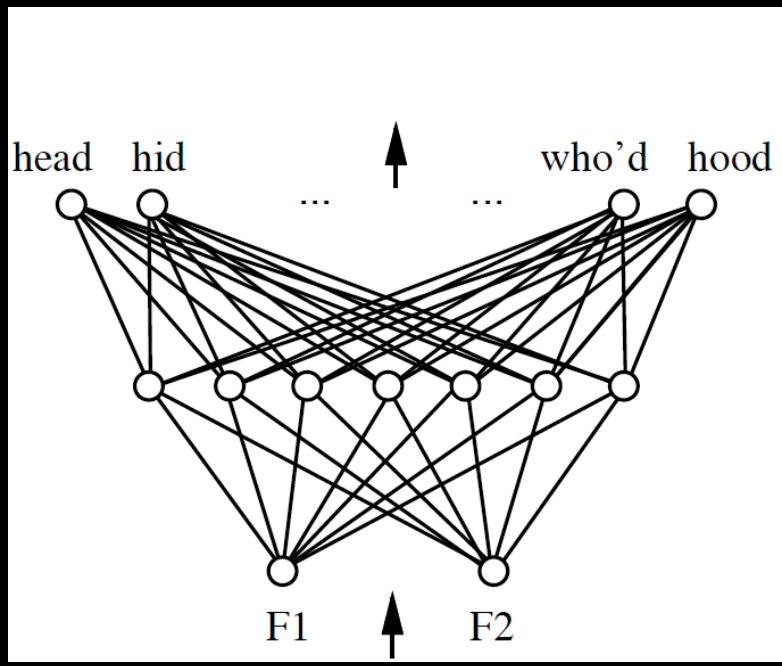
Research Scientist, Boulder Language Technologies

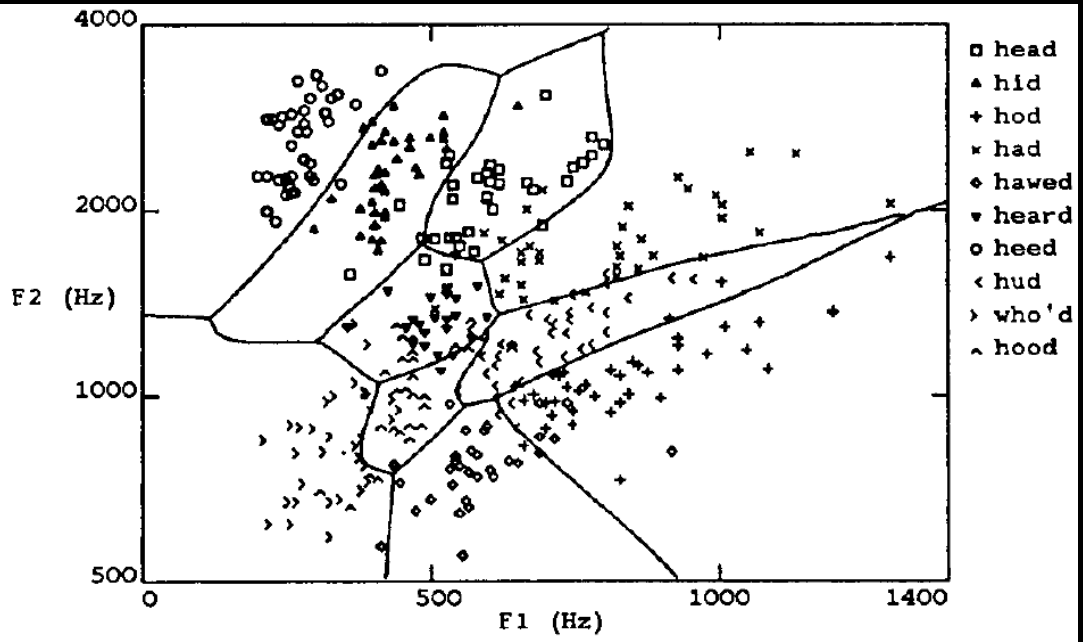
ANN Recap: Perceptrons

- **Perceptron representational power**



Multilayer Networks





Representational Power

- Boolean functions can be learned by an appropriately sized single-hidden-layer network
- Bounded continuous functions can be learned by the right sized single-HL network
- Arbitrary functions can be learned by a two-HL network

Backpropagation Algorithm

Create network and init each w_{jh} & w_{hk} to a small random value

Until termination condition is met, Do: (*1000s of epochs*)

For each <x,y>

Calculate all unit outputs

***Error Backpropagation*:**

For all net outputs, *k* = 0..*K*

$$\delta_k \leftarrow \hat{y}_k\left(1-\hat{y}_k\right)\left(y_k-\hat{y}_k\right)$$

For all hidden units, *h* = 0..*H*

$$\delta_h \leftarrow o_h\left(1-o_h\right)\sum_{k=1..K} w_{hk}\delta_k$$

Update all *w*..

$$w_{jh} \leftarrow w_{jh} + \Delta w_{jh}, \quad w_{hk} \leftarrow w_{hk} + \Delta w_{hk}$$

$$\Delta w_{jh} = \eta\delta_h x_j, \quad \Delta w_{hk} = \eta\delta_k o_h$$

ANNs Summary

- Real, discrete, and vector-valued functions
- Continuous or discrete valued attributes
- One-HL networks can represent any Boolean or continuous-valued function
- Two-HL networks can represent any function
- Hidden units create new features not in the input space
- Must avoid overfitting, as with any ML algorithm
- Backpropagation only one of many algorithms

Instance-based Learning

- Save detailed descriptions of the training data rather than generalize and abstract away from these details during training
- Zero training time – "Lazy" learning method
- Can be slow at test/classification time
- Methods
  - *k*-Nearest Neighbor
  - Locally weighted regression
  - Case-based reasoning

ML

Instance-based Learning

- Real or discrete-valued target functions
- Real or discrete-valued attributes

Advantages & Disadvantages

- Advantage
  - Zero training time
  - Target function might be too complex for most algorithms, but can still be reasonably represented by local approximations
- Disadvantage
  - Classification can be slow
  - Typically compares all attribute values, even those that are not relevant

k-Nearest Neighbor

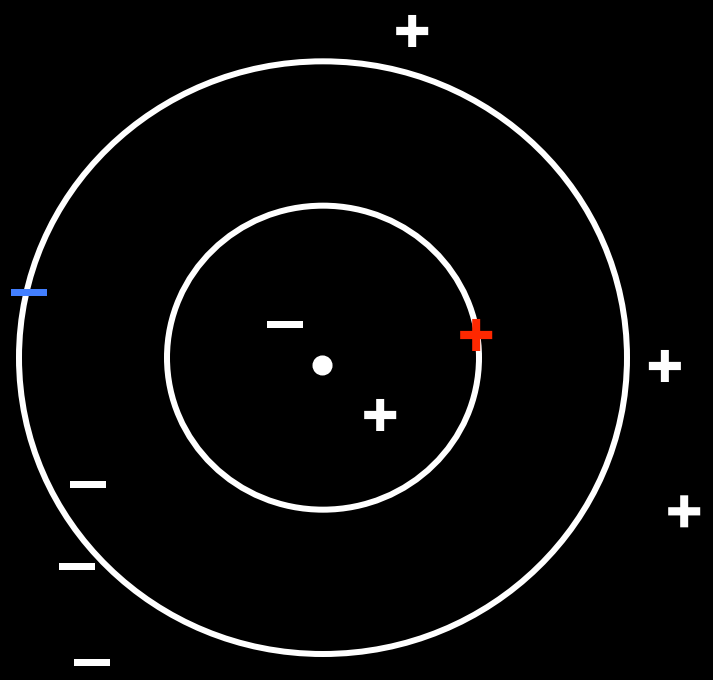
- **Training algorithm**
  - **Store the training examples**
- **Classification algorithm**
  - **Given a query instance $x^{(q)}$ to be classified**
    - **Find the *k* nearest neighbors of $x^{(q)}$ in the training data**
    - **For classification: Return the most common class of these *k* instances**
    - **For regression: Return the mean output (*y*) value of these *k* instances**

Computing "Nearest"

- ***Euclidean distance***
  - $d(x^{(q)}, x^{(i)}) = \sqrt{(x^{(q)} - x^{(i)})^2}$
  - i.e., the square root of the sum of squared differences in the attribute values
- **Inductive Bias**
  - The classification of $x^{(q)}$ will be most similar to nearby training instances

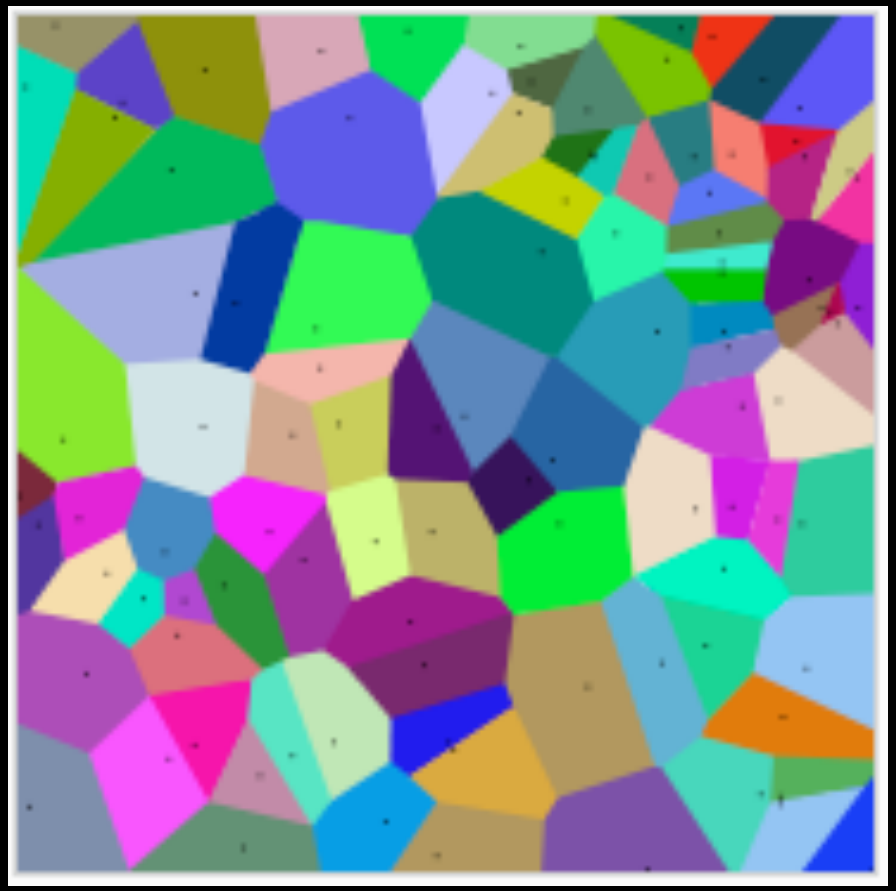
k-Nearest Neighbor

- ***k*-NN algorithm**
  - **Lazy vs. Eager Learning**



Nearest Neighbor

- **Voronoi diagram**

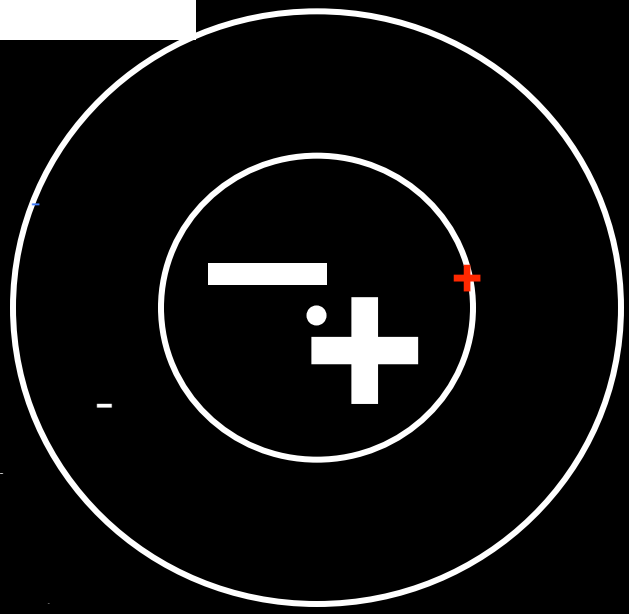


Distance-Weighted *k*-NN

- **Place more confidence in closer neighbors**

$$\hat{f}\left(x^{(q)}\right) \leftarrow \underset{c \in Classes}{\arg\max} \sum_{i=1}^{k} \frac{1}{d\left(x^{(q)}, x^{(i)}\right)^2} \delta\left(c, f\left(x^{(i)}\right)\right)$$

$$\delta(u,v) = \begin{cases} 1 & \text{if } u = v \\ 0 & \text{otherwise} \end{cases}$$



- **Local vs. global**

Are You a Skier

ML

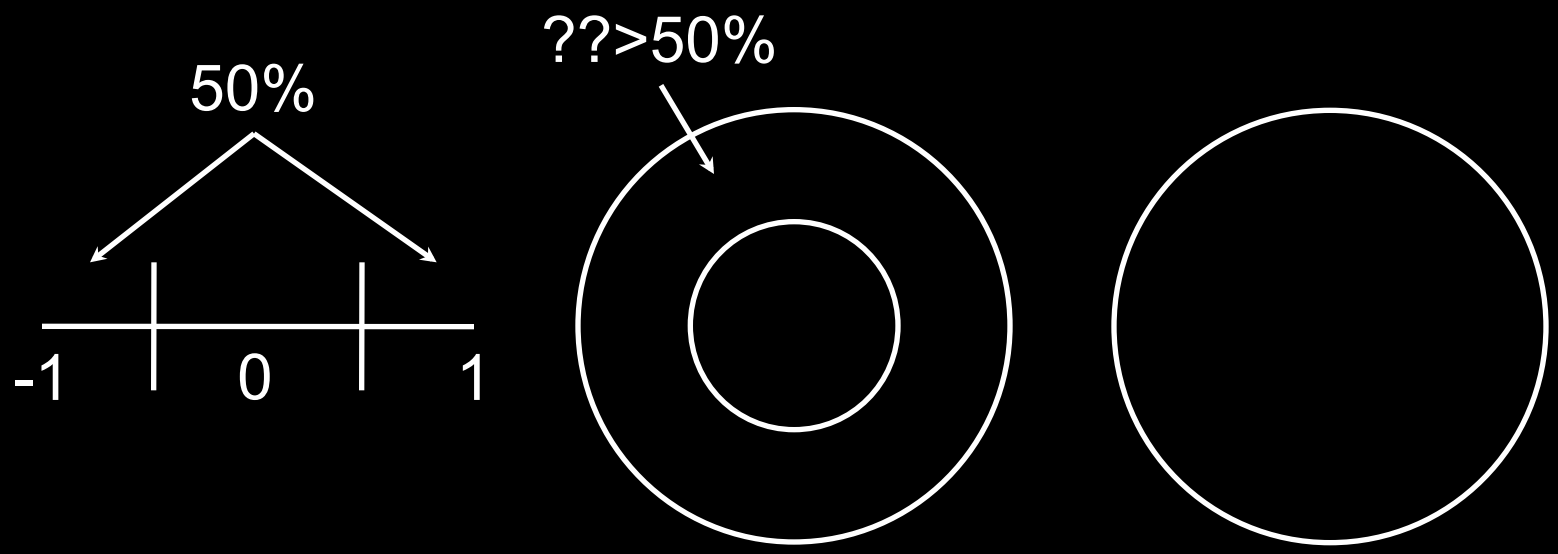
- **Attributes**
  - **First Name**
  - **Middle Name**
  - **Last Name**
  - **Telephone extension**
  - **Blood Type**
  - **Gender**
  - **Eye Color**
  - **Coordinated?**

The Curse of Dimensionality

- ***k*-NN distance, $d(u,v)$, is based on all attributes**
- **If only a few out of several attributes are relevant to the classification, accuracy will be hurt**
- **Stretch axes by multiplying each by some factor z_j**
  - **Use cross validation to determine appropriate factors**

The Curse of Dimensionality

- **All points are near the edge of space in high dimensional datasets**



Locally Weighted Regression

- For each new instance to be classified, x_q, create a new approximation of f based on the examples in the same region of space

$$E\left(x^{(q)}\right) \equiv \sum_{x \in kNNs \text{ of } x^{(q)}}^{k} g_w\left(d\left(x^{(q)}, x\right)\right)\left(f(x) - \hat{f}(x)\right)^2$$

$$\Delta w_j = \eta \sum_{x \in kNNs \text{ of } x^{(q)}}^{k} g_w\left(d\left(x^{(q)}, x\right)\right)\left(f(x) - \hat{f}(x)\right)x_j$$

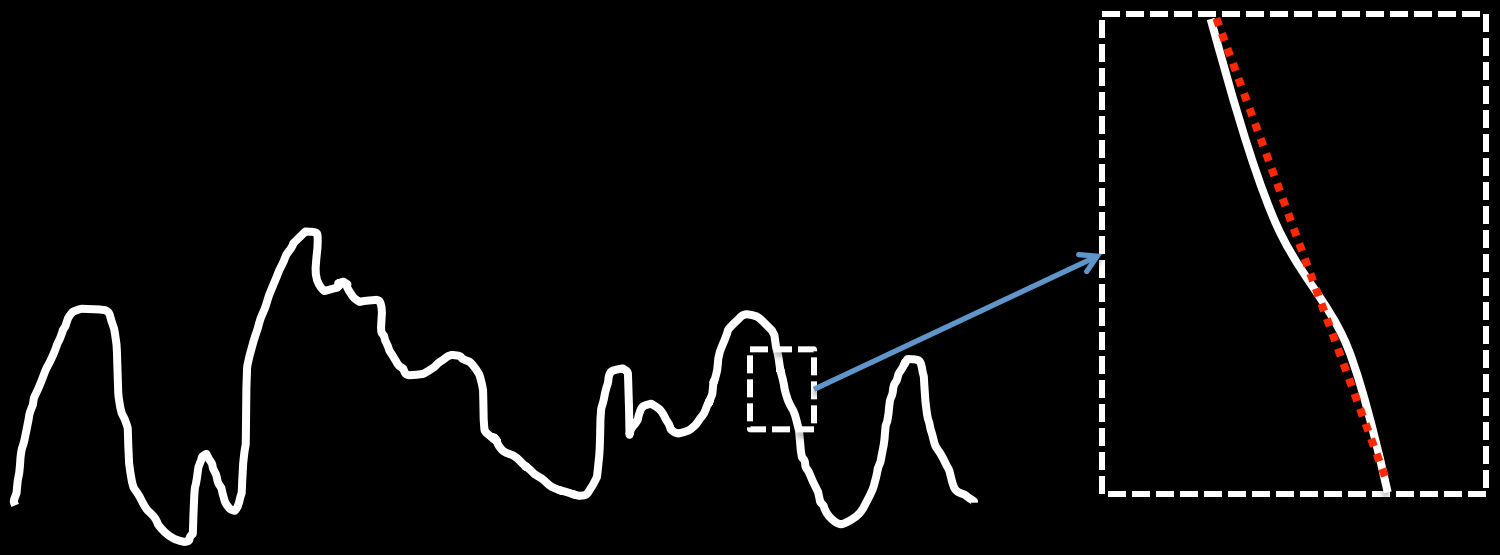
Locally Weighted Regression

$$E\left(x^{(q)}\right) \equiv \sum_{x \in kNNs \text{ of } x^{(q)}}^{k} g_w\left(d\left(x^{(q)}, x\right)\right)\left(f(x) - \hat{f}(x)\right)^2$$

$$\Delta w_j = \eta \sum_{x \in kNNs \text{ of } x^{(q)}}^{k} g_w\left(d\left(x^{(q)}, x\right)\right)\left(f(x) - \hat{f}(x)\right)x_j$$

Linear Approximation

- Generally, you should not need a complicated function approximation
  - In a small enough local region a linear or quadratic function is a reasonable approximation



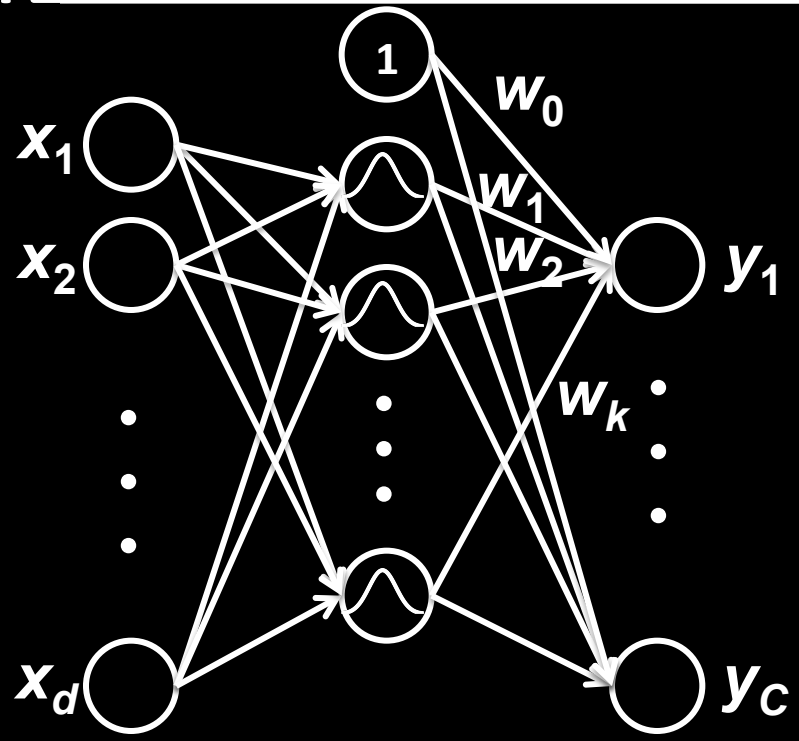
Radial Basis Functions

$$\hat{f}\left(x^{(q)}\right) \equiv w_0 + \sum_{k=1}^{n_k} w^{(k)} g^{(k)}\left(d\left(x^{(q)}, x^{(k)}\right)\right)$$

$$h(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$$

$$g^{(k)}\left(d\left(x^{(q)}, x^{(k)}\right)\right) = \exp\left(-\frac{d\left(x^{(q)}, x^{(k)}\right)}{2\sigma^{(k)2}}\right)$$

ML Radial Basis Function Network



$$\mathbf{x}^{(k)} = \mathbf{x}^{(i)},$$
$$\mathbf{x}^{(k)} = \overline{\mathbf{x}}^{(i)},$$
$$\ldots$$

$$\hat{f}\left(\mathbf{x}^{(q)}\right) \equiv w_0 + \sum_{k=1}^{n_k} w^{(k)} \exp\left(-\frac{\left(\mathbf{x}^{(q)} - \mathbf{x}^{(k)}\right)^2}{2\sigma^{(k)2}}\right)$$

Case-based Reasoning

- Lazy learning method
- Classify (or act on) a query instance based on similar instances
- Typically not based on Euclidean distance
  - Complex symbolic representations and retrieval mechanisms

Cased-based Reasoning

- AT&T: configuring phone services
- Helene Curtis: Product up-sales
- Illinois Chamber of Commerce & Illinois Dept. of Labor: Matching job applicants with open positions
- Xerox: configuring high-end document processing machines
- Jefferson County: address correction

Cased-based Reasoning

- Address correction
  - Street number
  - Street # suffix
  - Street name
  - Directional
  - Street type
  - Unit type
  - Unit number
  - Unit #suffix
  - City
  - State
  - Zip code

Lazy versus Eager

ML

- Computation time
  - Lazy methods are faster during training, generally
  - Eager methods are faster at classification time, generally
- Inductive Bias
  - Lazy methods can consider the query instance $x^{(q)}$
  - Eager methods make a single global approximation in advance versus local approximations at classification / test time
  - Are Eager methods always more restricted in choosing a hypothesis?

ML Summary of Instance-based Learning

- Put off processing training data until they see the query instance and then form local approximation
- Can model complex target function by simpler local approximations
- Less efficient labeling new instances, typically
- Might be difficult to choose distance metric
- Hard to deal with irrelevant features

ML Summary of Instance-based Learning

- ***k*-NN – based on Euclidean distance**
- **Locally weighted regression generalizes this to create a local approximation**
- **Radial basis function networks are a type of ANN constructed from localized kernels**
- **Case-based reasoning generally involves complex symbolic representations and search algorithms**