Artificial Intelligence

Statistical Learning

MONTANA STATE UNIVERSITY

Probabilistic Learning

- A probabilistic model of data can be used to
  - make inferences about missing inputs,
  - generate predictions about the data,
  - make decisions to minimize expected loss,
  - communicate the data in an efficient way.

MONTANA STATE UNIVERSITY

Features of Bayesian Learning

- Incremental presentation of training examples increase/decrease probability of each hypothesis.
- Considers both prior knowledge (priors) and observed data.
- Accommodates probabilistic hypotheses (e.g., 83% of students will pass the test).
- Can classify new instances by combining predictions from multiple hypotheses.

MONTANA STATE UNIVERSITY

Bayes Rule

- Let
  - $P(h)$ = prior probability of hypothesis h.
  - $P(D)$ = evidence or marginal likelihood of training data D.

$$P(D) = \int P(D \mid h) P(h) dh$$

  - $P(h|D)$ = probability of h given D.
  - $P(D|h)$ = probability of D given h.

$$P(h \mid D) = \frac{P(D \mid h) P(h)}{P(D)}$$

MONTANA STATE UNIVERSITY

Mountains Minds

Some More Definitions

- **Maximum Likelihood Learning:** Does not assume a prior over the model parameters. Finds a parameter setting the maximizes the likelihood of the data $P(D|h)$.
- **Maximum A Posterior (MAP) Learning:** Assumes a prior over the model parameters $P(h)$. Finds a parameter setting that maximizes the posterior: $P(h|D) \propto P(h)P(D|h)$.
- **Bayesian Learning:** Assumes a prior over the model parameters. Computes the posterior distribution of the parameters: $P(h|D)$.

MONTANA STATE UNIVERSITY

Mountains Minds

Choosing Hypotheses

- Generally, we want the most probable hypothesis given the training data.
- ***Def:*** The *maximum a posteriori* hypothesis (h_{MAP}) is given as

$$\begin{aligned} h_{MAP} &= \arg\max_{h \in H} P(h \mid D) \\ &= \arg\max_{h \in H} \frac{P(D \mid h) P(h)}{P(D)} \\ &= \arg\max_{h \in H} P(D \mid h) P(h) \end{aligned}$$

- If we assume $P(h_i) = P(h_j)$ for all i and j, then we can simplify and choose the *maximum likelihood* hypothesis.

$$h_{ML} = \arg\max_{h \in H} P(D \mid h)$$

MONTANA STATE UNIVERSITY

Mountains Minds

Relation to Concept Learning

- Consider the usual concept learning task.
  - Instance space X, hypothesis space H, training examples D.
  - Consider *FindS* learning algorithm (recall it picks a most specific hypothesis from the version space $VS_{H,D}$ consistent with positive examples).
- What would Bayes Rule produce as the MAP hypothesis?

MONTANA STATE UNIVERSITY

Mountains Minds

Relation to Concept Learning

- Assume fixed set of instances $\langle x_1,\ldots,x_m\rangle$
- Assume D is the set of all classified instances, $D = \langle c(x_1),\ldots,c(x_m)\rangle$
- Choose $P(D|h)$
  - $P(D|h) = 1$ if h is consistent with D
  - $P(D|h) = 0$ otherwise.
- Choose $P(h)$ to be uniform distribution
  - $P(h) = 1/|H|$ for all h in H.
- Then, using Bayes Rule, $P(h \mid D) = \begin{cases} \frac{1}{|VS_{H,D}|} & \text{if } h \text{ is consistent with } D \\ 0 & \text{otherwise} \end{cases}$

MONTANA STATE UNIVERSITY

Mountains Minds

Learning a Real-Valued Function

- Consider any real-valued target function f.
- Training examples $\langle x_i,d_i\rangle$, where d_i is noisy training value
  - $d_i = f(x_i) + e_i$
  - e_i is random variable (noise) drawn independently for each x_i according to Gaussian distribution with mean = 0.
- Then maximum likelihood hypothesis h_{ML} is the one that minimizes the sum of squared errors.

MONTANA STATE UNIVERSITY

Mountains Minds

Learning a Real-Valued Function

- Derivation—minimizing squared error.

$$h_{ML} = \arg\max_{h \in H} P(D \mid h)$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} P(d_i \mid h)$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2}$$

MONTANA STATE UNIVERSITY

Mountains Minds

Learning a Real-Valued Function

- Maximize natural log instead

$$h_{ML} = \arg\max_{h \in H} \sum_{i=1}^{m} \ln \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2$$

$$= \arg\max_{h \in H} \sum_{i=1}^{m} -\frac{1}{2}\left(\frac{d_i - h(x_i)}{\sigma}\right)^2$$

$$= \arg\max_{h \in H} \sum_{i=1}^{m} -(d_i - h(x_i))^2$$

$$= \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - h(x_i))^2$$

MONTANA STATE UNIVERSITY

Mountains Minds

Most Probable Classification

- So far, we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP}).
- Given a new instance x, what is the most probable *classification* for that instance?
  - $h_{MAP}(x)$ is not the most probable classification!
- Consider
  - Three possible hypotheses
    - $P(h_1|D) = .4$, $P(h_2|D) = .3$, $P(h_3|D) = .3$
  - Given new instance, x
    - $h_1(x) = +$, $h_2(x) = -$, $h_3(x) = -$
  - What is most probable classification?

MONTANA STATE UNIVERSITY

Mountains Minds

Bayes Optimal Classifier

- Bayes optimal classification is given by

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

- Example
  - $P(h_1|D) = .4$, $P(-|h_1) = 0$, $P(+|h_1) = 1$
  - $P(h_2|D) = .3$, $P(-|h_2) = 1$, $P(+|h_2) = 0$
  - $P(h_3|D) = .3$, $P(-|h_3) = 1$, $P(+|h_3) = 0$

MONTANA STATE UNIVERSITY

Mountains Minds

Bayes Optimal Classifier

- Therefore

$$\sum_{h_i \in H} P(+ \mid h_i) P(h_i \mid D) = 0.4$$

$$\sum_{h_i \in H} P(- \mid h_i) P(h_i \mid D) = 0.6$$

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D) = -$$

MONTANA STATE UNIVERSITY

Mountains Minds

Gibbs Classifier

- Bayes optimal classifier provides best result, but can be expensive if there is a larger number of hypotheses.
- Gibbs algorithm:
  - Choose one hypothesis at random, according to $P(h|D)$
  - Use this hypothesis to classify a new instance.

MONTANA STATE UNIVERSITY

Mountains Minds

Gibbs Classifier

- Surprising fact: Assume target concepts are drawn at random from H according to priors on H.
  - $E[error_{Gibbs}] \leq 2E[error_{BayesOptimal}]$
- Suppose correct, uniform prior distribution over H, then
  - Pick any hypothesis from VS, with uniform probability.
  - Its expected error is no worse that twice the Bayes optimal.

Naïve Bayes Classifer

- Along with decision trees, neural networks, nearest neighbor, the naïve Bayes classifier is one of the most practical learning methods.
- When to use:
  - Moderate to large training set available
  - Attributes that describe instances are conditionally independent given classification

Naïve Bayes Classifier

- Assume target function $f : X \rightarrow V$, where each instance x described by attributes $\langle a_1, a_2, \ldots, a_n \rangle$.
- Most probable value of $f(x)$ is

$$
\begin{aligned}
v_{MAP} &= \arg\max_{v_j \in V} P(v_j \mid a_1, a_2, \ldots, a_n) \\
&= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)} \\
&= \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j)
\end{aligned}
$$

Naïve Bayes Classifer

- Naïve Bayes Assumption (conditional independence)

$$P(a_1, a_2, \ldots, a_n \mid v_j) = \prod_i P(a_i \mid v_j)$$

- This gives the definition of the Naïve Bayes Classifier

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

MONTANA STATE UNIVERSITY

Mountains Minds

Naïve Bayes Algorithm

Naïve_Bayes_Learn(*examples*)
For each target value v_j
$P'(v_j) \leftarrow$ estimate $P(v_j)$
For each attribute value a_i
of each attribute a
$P'(a_i|v_j) \leftarrow$ estimate $P(a_i|v_j)$

Classify_New_Instance(*x*)

$$v_{NB} = \arg\max_{v_j \in V} P'(v_j) \prod_{a_i \in x} P'(a_i \mid v_j)$$

- $P(v_j)$ estimated as proportion of examples labeled v_j.
- $P(a_i|v_j)$ estimated as proportion of examples labeled v_j with feature value a_i.

MONTANA STATE UNIVERSITY

Mountains Minds

Subtleties of Naïve Bayes

- Conditional independence assumption is often violated

$$P(a_1, a_2, \ldots, a_n \mid v_j) \neq \prod_i P(a_i \mid v_j)$$

- But it works surprisingly well anyway.
- Note we don't need estimated posteriors $P(v_j|x)$ to be correct. Only need

$$\arg\max_{v_j \in V} P'(v_j) \prod_i P'(a_i \mid v_j) =$$

$$\arg\max_{v_j \in V} P(v_j) P(a_1, \ldots, a_n \mid v_j)$$

MONTANA STATE UNIVERSITY

Mountains Minds

Subtleties of Naïve Bayes

- What if none of the training instances with target value v_j have attribute value a_i?
- Result is that

$$P'(a_i \mid v_j) = 0$$

$$P'(v_j)\prod_i P'(a_i \mid v_j) = 0$$

- Typical solution is to use the Bayesian estimate for $P'(a_i|v_j)$

MONTANA STATE UNIVERSITY

Mountains Minds

Estimating $P'(a_i|v_j)$

- Classification will follow some kind of "multinomial" distirbution.
- The so-called "*m*-estimate" is a form of "Dirichlet" prior, which is conjugate for the multinomial distribution.

$$P'(a_i \mid v_j) = \frac{n_c + mp}{n + m}$$

  - n is the number of training examples where $v = v_j$.
  - n_c is the number of examples where $v = v_j$ and a_i.
  - p is the prior estimate for $P'(a_i|v_j)$
  - m is a weight given to the prior (i.e., a number of "virtual" examples)

MONTANA STATE UNIVERSITY

Mountains Minds

Topics Not Covered

- Learning Bayesian Networks
  - Inducing conditional dependence relations (i.e., Bayesian network structure)
  - Inducing conditional probability tables.
- Other probabilistic models.
  - Cognitive maps
  - Qualitative probabilistic networks
  - Other graphical models

MONTANA STATE UNIVERSITY

Mountains Minds

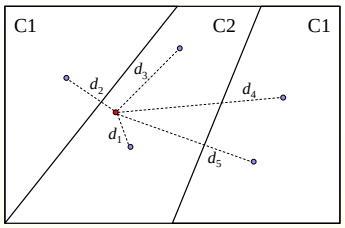
Nearest Neighbor Classification

- Let P = a set of points, $p = \langle \{f_{1,p},\ldots,f_{n,p}\};c^p \rangle$.
- Let $f_{i,p}$ = the i^{th} feature of point p.
- Let c^p = the class label for point p.
- Let $q = \langle \{f_{1,q},\ldots,f_{n,q}\};c^q \rangle \notin P$
- Let $d(p,q)$ = the "distance" between points p and q.
- Find $r \in P \ni \forall p \in P$, $p \neq r$, $d(r,q) < d(p,q)$, and return the associated class c^r.

MONTANA STATE UNIVERSITY

Mountains Minds

Example



MONTANA STATE UNIVERSITY

Mountains Minds

Metric Spaces

- For numeric features, nearest-neighbor depends on the definition of distance which conforms to the following properties:
  - $d(p,q) \geq 0$
  - $d(p,q) = d(q,p)$
  - $d(p,p) = 0$
  - $d(p,q) + d(q,r) \geq d(p,r)$

MONTANA STATE UNIVERSITY

Mountains Minds

Minkowski Metrics

- The most commonly-used distance metrics are called L_p-metrics or Minkowski metrics.

$$d_p(p,q) = \left[\sum_i (f_{i,p} - f_{i,q})^p \right]^{1/p}$$

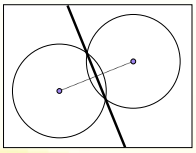
- Most common forms:
  - Manhattan distance $d_1(p,q) = \sum |(f_{i,p} - f_{i,q})|$
  - Euclidean distance $d_2(p,q) = \sqrt{\sum_i (f_{i,p} - f_{i,q})^2}$
  - Max-coordinate distance $d_\infty(p,q) = \max_i |(f_{i,p} - f_{i,q})|$

MONTANA STATE UNIVERSITY

Mountains Minds

Voronoi Diagrams

- A point in Euclidean *n*-space defines the center of a region of that *n*-space. This region is represented by an *n*-dimensional hypersphere.
- Two points define two regions. The separator between the regions is a hyperplane

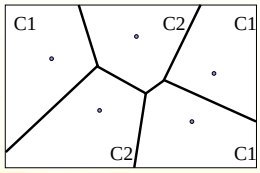


MONTANA STATE UNIVERSITY

Mountains Minds

Voronoi Diagrams

- When several points exist in an *n*-space, regions corresponding to each of these points are characterized by complex polytopes



MONTANA STATE UNIVERSITY

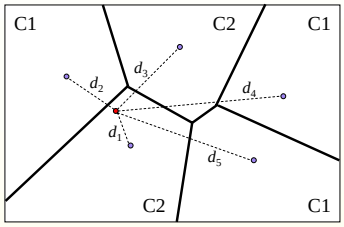
Mountains Minds

Using Voronoi Diagrams

- Label each polytope with the class assigned to the point in the center of the polytope.
- For a new point, determine which region contains that point.
- The point at the center is the new point's nearest neighbor, so assign that point's class.

MONTANA STATE UNIVERSITY
Mountains Minds
31

Example



MONTANA STATE UNIVERSITY
Mountains Minds
32

Problems

- Voronoi diagrams are not practical in high dimensions.
  - Projection
  - *kd*-trees
- Voronoi diagrams are difficult to visualize with symbolic features.
- Minkowski metrics don't apply with symbolic features.

MONTANA STATE UNIVERSITY
Mountains Minds
33

Symbolic Nearest Neighbor

- Assume *all* features are symbolic, i.e., for a feature f_x, assume that feature can take on a value from the discrete set $\{v_{1,x},\ldots,v_{n,x}\}$.
- For each feature, construct a "similarity matrix" comparing pairs of values.
- Then we can define a new distance metric that satisfies the properties of a metric space.

MONTANA STATE UNIVERSITY
Mountains Minds

Value Difference Metric

- Let $\delta(v_i,v_j)$ be the similarity between feature values v_i and v_j.
- Let p be an exponent like in L_p-metrics.
- Let n = the number of classes.
- Let C_i = the number of times v_i is assigned to f_x in the training set.
- Let $C_{i,a}$ = the number of examples in the training set where $f_x = v_i$, and the examples are in class a.

$$\delta(v_i, v_j) = \sum_{l=1}^{n} \left| \frac{C_{i,l}}{C_i} - \frac{C_{j,l}}{C_j} \right|^p$$

MONTANA STATE UNIVERSITY
Mountains Minds

VDM Intuition

- Note that $C_{i,l} / C_i = \Pr(p_i \in l \mid f_x = v_i)$
- Two *feature values* are considered to be similar if they occur with the same relative frequency for all classifications.
- Total difference between two points is

$$\Delta(p, q) = \sum_f \delta(p_f, q_f)$$

MONTANA STATE UNIVERSITY
Mountains Minds

Naïve Nearest Neighbor Search

```
FIND-NN(E,q)
        nearest-dist := ∞;
        nearest := ⊥;
        ∀ ex ∈ E do
                curr-dist := DIST(ex,q);
                if curr-dist < nearest-dist then
                        nearest-dist := curr-dist;
                        nearest := ex;
        return nearest;
```

MONTANA STATE UNIVERSITY

Mountains Minds

Analysis

- May need to examine all examples in **E**, especially if *q* is not in **E**, thus $O(|\mathbf{E}|)$ comparisons.
- Each comparison involves computing DIST(*ex*,*q*) which requires $\Theta(k)$ calculations for *k* dimensions.
- Thus, worst-case performance for naïve nearest neighbor search is $O(|\mathbf{E}|k)$.
- The *k*d-tree is an attempt to reduce the average complexity to $O(k \lg |\mathbf{E}|)$.

MONTANA STATE UNIVERSITY

Mountains Minds

*K*d-Tree

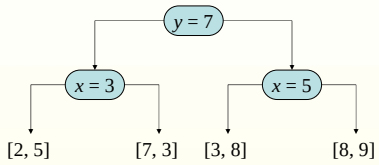
- A *k*d-tree is a binary tree that is used to store a finite set of points from *k*-dimensional space to facilitate efficient search.
- Interior nodes of the tree define split points.
- Leaf nodes define regions containing a single point (i.e., example).
- Balanced tree leads to average traversal time of $O(k \lg |\mathbf{E}|)$.

MONTANA STATE UNIVERSITY

Mountains Minds

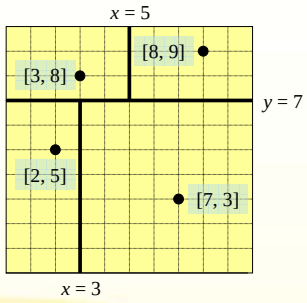
Example

E = {[7, 3], [2, 5], [3, 8], [8, 9]}



MONTANA STATE UNIVERSITY

Decision Partitions for Example



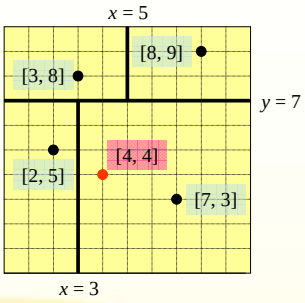
MONTANA STATE UNIVERSITY

Query Point

- Using the *k*d-tree involves a form of branch-and-bound search.
- Suppose our query point is q = [4, 4]. What is the nearest neighbor?
- A first approximation is found at the leaf node whose region contains q.
- Examining the decision partition diagram, we would assume that the nearest neighbor is, therefore, [7, 3].

MONTANA STATE UNIVERSITY

Query Point



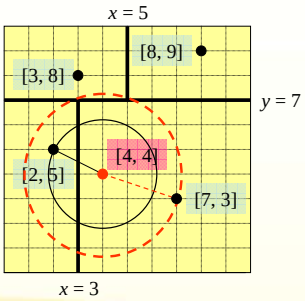
MONTANA STATE UNIVERSITY

Correct Neighbor

- Unfortunately, [7, 3] is not the correct nearest neighbor (assuming Euclidean distance).
- From the leaf, we must travel back to the root and consider the “other” side of the split.
- We can prune other sides if there is no way their regions can contain a nearer point.
- This is determined by examining a “ball” centered on *q* with radius = current NN.

MONTANA STATE UNIVERSITY

Correct Neighbor



MONTANA STATE UNIVERSITY

Determining Search Regions

- If the ball does not overlap a region, that region does not need to be searched.
- If the ball does overlap a region then,
  - Search the region until a point is found
  - If that point is nearer, update the radius of the ball and resume up the tree.
  - Otherwise, just resume up the tree without updating the radius.

MONTANA STATE UNIVERSITY

Mountains Minds

Determining Ball Overlap

- To determine ball overlap, let q = the point at the center of the ball (i.e., the query point).
- Let $x^{min}(i)$ be the minimum value of region for i^{th} dimension. Let $x^{max}(i)$ be the maximum value of region for the i^{th} dimension.
- Find p' in the region nearest q, where for each $p'_i \in P$,

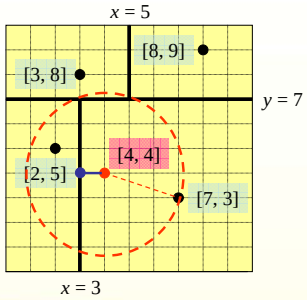
$$p'_i = \begin{cases} x^{min}(i) & \text{if } q_i \le x^{min}(i) \\ q_i & \text{if } x^{min}(i) \le q_i \le x^{max}(i) \\ x^{max}(i) & \text{if } x^{max}(i) \le q_i \end{cases}$$

- Ball overlaps if DIST(p',q) ≤ radius.

MONTANA STATE UNIVERSITY

Mountains Minds

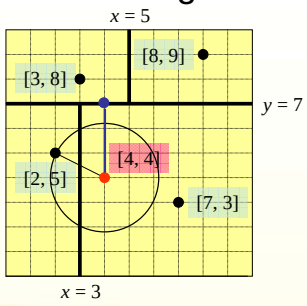
Correct Neighbor



MONTANA STATE UNIVERSITY

Mountains Minds

Correct Neighbor



MONTANA STATE UNIVERSITY

*K*d-Trees for *K*-Nearest Neighbor

- Nearest neighbor search can be expanded to find the *k* nearest neighbors to a query point.
- Maintain a priority queue of *k* nearest points seen so far. These could be initialized to the first *k* points in **E**.
- Set the ball radius to the distance between *q* and the k^{th} so far.
- Update priority queue and radius using standard *k*d-tree traversal.

MONTANA STATE UNIVERSITY

Error in Nearest Neighbor

- Generally, it is not reasonable to assume examples accurately represent concepts.
- Errors are introduced for several reasons:
  - Noise
  - Lack of precision
  - Human error
  - Irrelevant attributes
- For continuous concepts, examples are at best an "approximation."

MONTANA STATE UNIVERSITY

Reducing Error

- Can reduce error by reducing the size of the exemplar set.
- Nearest neighbor with a reduced training set is called *edited nearest neighbor*.
- There are several methods for editing the exemplar set.
- We will consider two of them.

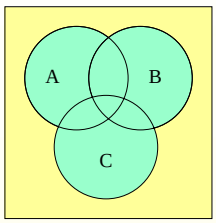
Editing Misclassified Examples

- Use points in the exemplar set to classify each other point using *k*-NN.
- If the point is misclassified, remove it from the exemplar set.
- Such editing can be done incrementally or in "batch" mode.
- Repeat process until performance begins to degrade or no further points are edited out.

Editing Correct Examples

- Use points in the exemplar set to classify each other point using *k*-NN.
- If the point is correctly classified, remove it from the exemplar set.
- As before, such editing can be done incrementally or in "batch" mode.
- Repeat process until performance begins to degrade or no further points are edited out.

Original Point Set

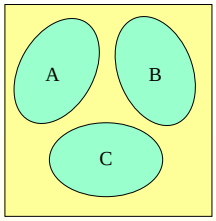


MONTANA STATE UNIVERSITY

55

Mountains Minds

Editing Misclassified Examples

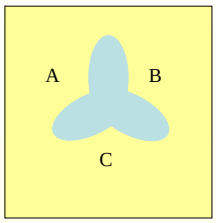


MONTANA STATE UNIVERSITY

56

Mountains Minds

Editing Correct Examples



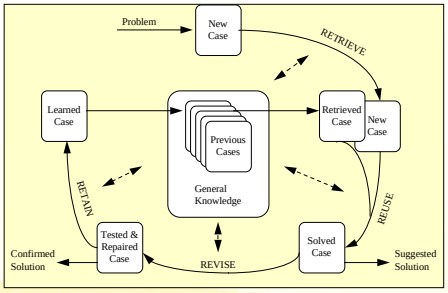
MONTANA STATE UNIVERSITY

57

Mountains Minds

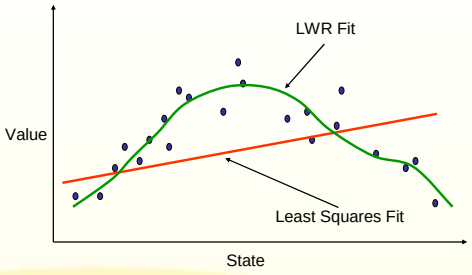
Memory Based Reasoning

Case Based Reasoning:



Memory Based Learning

Local Models:



Neural Network Classification

- Classification problem can be thought of as an “associative memory” problem.
- We want to learn an association between an instance of a concept and the concept itself.
- Associative memories are function approximators.
  - f^* : Example $\rightarrow$ Class Label
- We will be focusing on the feedforward neural network.

Neural Networks

- Want learning an arbitrary target function.
- Map a vector of real numbers to another vector of real numbers.

$$t : \mathbf{X} \subset \Re^m \rightarrow \mathbf{Y} \subset \Re^n$$

- Train on set **S** of *m* examples <**x**,*y*> such that **y** = *t*(**x**).
- Hope *t* will be able to *generalize* to do well on unseen examples

MONTANA STATE UNIVERSITY

Mountains Minds

Mean Squared Error

- Given a network that computes $f_{\mathbf{w}}(\mathbf{x})$ but is trying to learning target function $t(\mathbf{x})$ and a fixed sample **S** with *m* elements,
- Define the *mean squared error* of the network on **S** to be:

$$E(\mathbf{w}) = \frac{1}{|\mathbf{S}|} \sum_{\mathbf{x} \in \mathbf{S}} (f_{\mathbf{w}}(\mathbf{x}) - t(\mathbf{x}))^2$$

- Goal is to minimize MSE

MONTANA STATE UNIVERSITY

Mountains Minds

Gradient Descent

- Gradient descent is a hill climbing algorithm that we will use to minimize MSE.
- GD involves taking small steps in the direction that reduces error the most.
- Start with an initial weight vector $\mathbf{w} \in \Re^k$.
- Let η be a learning rate parameter and $\Delta\mathbf{w}$ be the direction of change.
- At each iteration, update.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \Delta \mathbf{w}_i$$

MONTANA STATE UNIVERSITY

Mountains Minds

Determining the Direction

- The direction that reduces error the fastest is *opposite* the gradient ∇E.

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots, \frac{\partial E}{\partial w_k} \right)$$

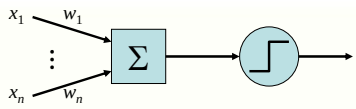
- This makes the update rule:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E$$

MONTANA STATE UNIVERSITY

Mountains Minds

McCulloch-Pitts Nodes

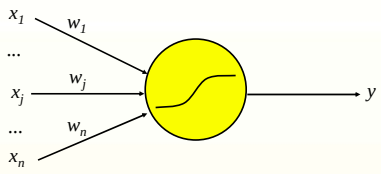


$$n_i(t+1) = \theta\left(\sum_j w_{ij} n_j(t) - \mu_i \right)$$

MONTANA STATE UNIVERSITY

Mountains Minds

Sigmoid Neuron



$$y = f(x) = \frac{1}{1 + e^{-\left(\sum_i w_i x_i + \theta\right)}}$$

MONTANA STATE UNIVERSITY

Mountains Minds

General Neuron

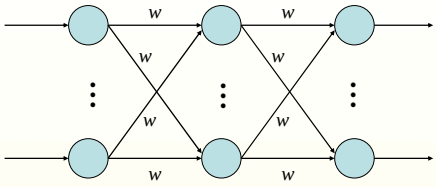
- To generalize McCulloch-Pitts model, do the following:
  - Replace $\theta(x)$ with general activation function.
  - Make updates asynchronous (i.e., eliminate time step).
  - Learn threshold μ by assuming additional input clamped at the value 1.

$$n_i = g\left(\sum_j w_{ij} n_j\right)$$

MONTANA STATE UNIVERSITY

Mountains Minds

Gradient Search and Neural Nets



$$\nabla E = \frac{\partial E}{\partial w} = \frac{\partial}{\partial w}\frac{1}{2}\sum_i (O_i - T_i)^2$$

MONTANA STATE UNIVERSITY

Mountains Minds

Perceptrons

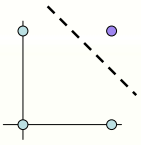
- Consider a simple two-layer network (inputs to outputs).
- Suppose we wish to construct such a network to implement the Boolean AND function.
- Inputs: ξ_1, ξ_2
- Output: ζ

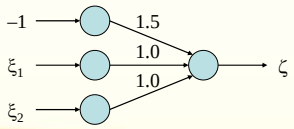
MONTANA STATE UNIVERSITY

Mountains Minds

AND Function

ξ_1	ξ_2	ζ
0	0	0
0	1	0
1	0	0
1	1	1





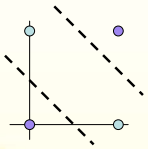
MONTANA STATE UNIVERSITY

Mountains Minds

70

Linear Separability

- If there exists a hyperplane dividing the input space to produce an accurate output, then the problems is ***linearly separable*.**
- If a problem is not linearly separable, then a perceptron (i.e., 2-layer network) is not capable of solving it.



MONTANA STATE UNIVERSITY

Mountains Minds

71

Multi-Layer Networks

- Nonlinear separability can be addressed by introducing additional layers into the feedforward network.
- The difficulty arises in constructing such a network.
- It is possible to “learn” the weights for the network (given a structure) from a set of examples.

MONTANA STATE UNIVERSITY

Mountains Minds

72

Initial Weight Vectors

- Choice of initial weight vector $\mathbf{w}_0$ is important in determining whether GD converges to a local or global minimum.
- Methods for choosing $\mathbf{w}_0$ include:
  - Random generation
  - Predefined, fixed vector
  - Carefully selected initial vector
- In general, there is no way to guarantee the global minimum will be found.
- Multiple restarts used frequently.

Learning Rate

- The choice of learning rate, η, determines if GD will converge, and how long it will take.
- If η is too small, GD will progress too slowly.
- If η is too large, GD will overshoot the minimum and possibly not converge.
- There are no systematic methods for choosing η; however, there are adaptive approaches (e.g., annealing).

Backpropagation

- Construct the network.
- Initialize all weights to small random values.
- Choose some pattern ξ^p and apply it to the input layer.
- Propagate the signal forward through the network using a sigmoid activation function.

Backpropagation

- Compute delta term for output layer.
- Compute delta terms for hidden layers, working back toward input layer.

$$\delta_i^{p,n} = O_i^{p,n}(1 - O_i^{p,n})(\zeta_i^p - O_i^{p,n})$$

$$\delta_j^{p,l-1} = O_j^{p,l-1}(1 - O_j^{p,l-1}) \sum_i w_{ij} \delta_i^{p,l}$$

MONTANA STATE UNIVERSITY

Mountains Minds

Backpropagation

- Update the weights using update rule at all layers.
- Then select another pattern and repeat.

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

$$= w_{ij} + \eta \delta_i^{p,l} O_j^{p,l-1}$$

MONTANA STATE UNIVERSITY

Mountains Minds

Updates

- Given a sample set, **S**, GD can be applied incrementally (i.e., on each $\mathbf{x} \in \mathbf{S}$) or in batch mode.
- Incremental minimizes error for one sample on each iteration.
- Batch determines ∇E as follows:

$$\nabla E(\mathbf{S}) = \frac{1}{|\mathbf{S}|} \sum_{\mathbf{x} \in \mathbf{S}} \nabla E(\mathbf{x})$$

MONTANA STATE UNIVERSITY

Mountains Minds

Momentum

- GD has difficulty in landscapes with narrow, gently sloping valleys.
- These landscapes tend to cause oscillation.
- Momentum provides a way to cancel out short term oscillations.

$$\Delta \mathbf{w}_{t+1} = \alpha \Delta \mathbf{w}_t - (1-\alpha)\nabla E$$

MONTANA STATE UNIVERSITY Mountains Minds

79

Ex: Learning Linear Functions

- Suppose we want to learn $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{wx}$, where $\mathbf{w}, \mathbf{x} \in \Re^k$.
- This is similar to a perceptron, but with no threshold (i.e., bias).
- For batch updating:

$$E(\mathbf{w}) = \frac{1}{|\mathbf{S}|}\sum_{i=1}^{|\mathbf{S}|}(\mathbf{w}\cdot\mathbf{x}_i - y_i)^2$$

$$\nabla E = \frac{\partial E(\mathbf{w})}{\partial(\mathbf{w})} = \frac{2}{|\mathbf{S}|}\sum_{i=1}^{|\mathbf{S}|}(\mathbf{w}\cdot\mathbf{x}_i - y_i)\mathbf{x}_i$$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla E$$

MONTANA STATE UNIVERSITY Mountains Minds

80

Ex: Learning Linear Functions

- If we perform incremental (i.e., single point) updating, the update rule becomes

$$\mathbf{w} \leftarrow \mathbf{w} - \eta(\mathbf{w}\cdot\mathbf{x} - y)\mathbf{x}$$

- Note this update equation is very similar to the perceptron rule, except now the update is proportional to the error.

MONTANA STATE UNIVERSITY Mountains Minds

81

Sigmoid Functions

- As can be seen, sigmoid functions are similar to threshold functions except
  - They are smooth
  - They have computable, continuous derivatives
- Other properties include
  - Domain is the set of real numbers
  - Range is a bounded interval
  - Function is monotonic

MONTANA STATE UNIVERSITY

82

Mountains Minds

Hyperbolic Tangent

- The hyperbolic tangent

$$z = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- The derivative

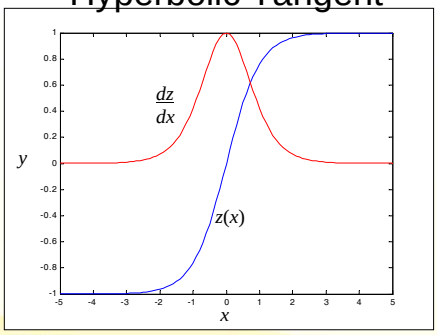
$$\frac{dz}{dx} = \operatorname{sech}^{2}(x) = \frac{4}{(e^{x} + e^{-x})^{2}}$$

MONTANA STATE UNIVERSITY

83

Mountains Minds

Hyperbolic Tangent



MONTANA STATE UNIVERSITY

84

Mountains Minds

Logistic Function

- The logistic function

$$z = \frac{1}{1+e^{-x}}$$

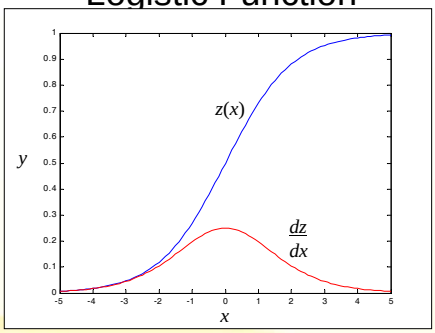
- The derivative

$$\frac{dz}{dx} = z(1-z)$$

MONTANA STATE UNIVERSITY

Mountains Minds

Logistic Function



MONTANA STATE UNIVERSITY

Mountains Minds