Lecture 7: More on Artificial Neural Networks

- ◇ Variations on backpropagation
- ◇ RBF networks
- ◇ PAC learning theory in infinite spaces: the VC dimension

Recall from last time: Backpropagation Algorithm

Initialize all weights to small random numbers.

Until satisfied, Do

For each training example, Do

1. Input the training example to the network and compute the network outputs
2. For each output unit k

   $$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h

   $$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{hk}\delta_k$$

4. Update each network weight w_{ij}

   $$w_{ij} \leftarrow w_{ij} + \eta\delta_j x_{ij}$$

   x_{ij} is the input from unit i into unit j (so for the output neurons, the x's are the signals received from the hidden layer neurons)

Adding momentum

On the n-th training sample, instead of the update:

$$\Delta w_{ij} \leftarrow \eta \delta_j x_{ij}$$

let's do:

$$\Delta w_{ij}(n) \leftarrow \eta \delta_j x_{ij} + \alpha \Delta w_{ij}(n-1)$$

The second term is called *momentum*

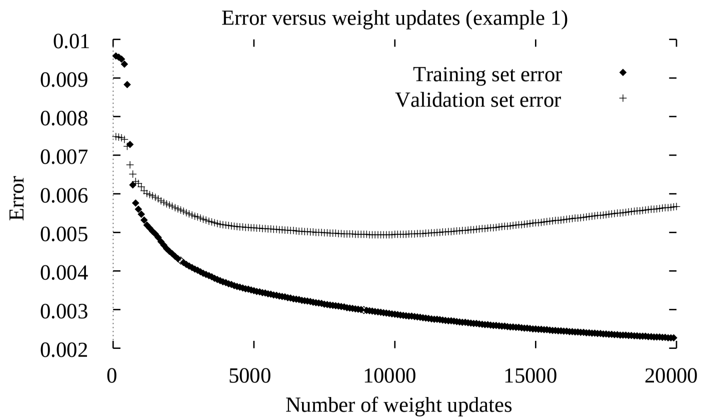
Advantages: gets us past small local minima and lets us go on flat surfaces;

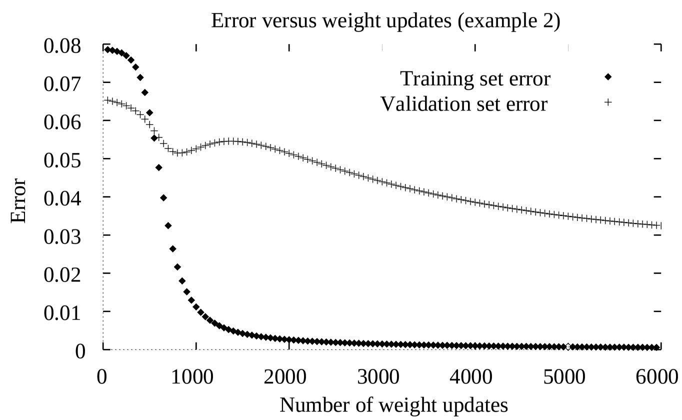
Also Makes us go fast in regions where the gradient stays unchanged

Disadvantage: with too much momentum we could go past a nice global minimum and into the next local one

Also one other parameter to tune, and more chances to get divergence

Overfitting in ANNs





Use a validation set to decide when to stop training!

Sometimes pruning is done too.

Alternative Error Functions

Penalize large weights:

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2 + \gamma \sum_{i,j} w_{ji}^2$$

Used to avoid overfitting.

Train on target slopes as well as values:

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} \left[(t_{kd} - o_{kd})^2 + \mu \sum_{j \in inputs} \left(\frac{\partial t_{kd}}{\partial x_d^j} - \frac{\partial o_{kd}}{\partial x_d^j} \right)^2 \right]$$

Tie together weights: Train each weight individually, but then replace the value of a weight with the mean of the weights obtained by backpropagation.

Constructive methods for Neural Networks

Meiosis networks (Hanson):

- Start with just one hidden unit, train using backprop
- The variance of each weight is maintained in addition to the magnitude
- If a unit has one or more weights of high variance, it is split into two units, and the weights are perturbed

Cascade correlation (Fahlman & Lebiere):

- Start with outputs only and train using backprop
- Add a neuron connected to all inputs, and train it to correlate to the residual error
- Connect the enuron to the output node, then freeze its weights and train the output again
- Continue adding units while the residual error is above a threshold

Radial Basis Function (RBF) Networks

- Many parts of the brain have neurons which are “locally tuned” to respond only if the input is within a certain range
- E.g. neurons in the auditory part of the brain are tuned to respond to different frequencies
- But sigmoid neurons do not have this characteristic!

Structure of an RBF Network

- There are a number of hidden units of the form:

$$z_i(\mathbf{x}) = \exp(-\frac{||\mathbf{x} - \mu_i||}{2\sigma_i^2}$$

- I.e. every unit is a Gaussian of mean μ_i and standard deviation σ_i, which will get "activated" if the input vector $\mathbf{x}$ is close to the mean μ_i
- The outputs are just linear combinations of the hidden units:

$$y_j = w_0 + \sum_i w_i z_i(\mathbf{x})$$

- Other choices of z_i are possible besides the Gaussian

Training RBF networks

- We want to find good values for the weights w_i, the centers μ_i and the widths σ_i
- Main idea: gradient descent!
- We can compute the derivative of the error function with respect to each parameter and get a learning rule that way
- The performance of this procedure is similar to that of sigmoid multi-layered networks. But one would hope for a faster learning process...
- Idea: Train the hidden units first, then it will be easy to determine wieghts for them
- Heuristics for determining means: choose randomly a number of training examples; use clustering
- Heuristic to determine widths: choose the distance to the closest other unit as a width
- These ensure fast training, but generalization performance is worse

ablishing PAC-like results for feed-forward neural networks

Recall from previous lectures:

$$m \geq \frac{1}{\epsilon}(\ln |H| + \ln(1/\delta))$$

is lower bound on the number of examples

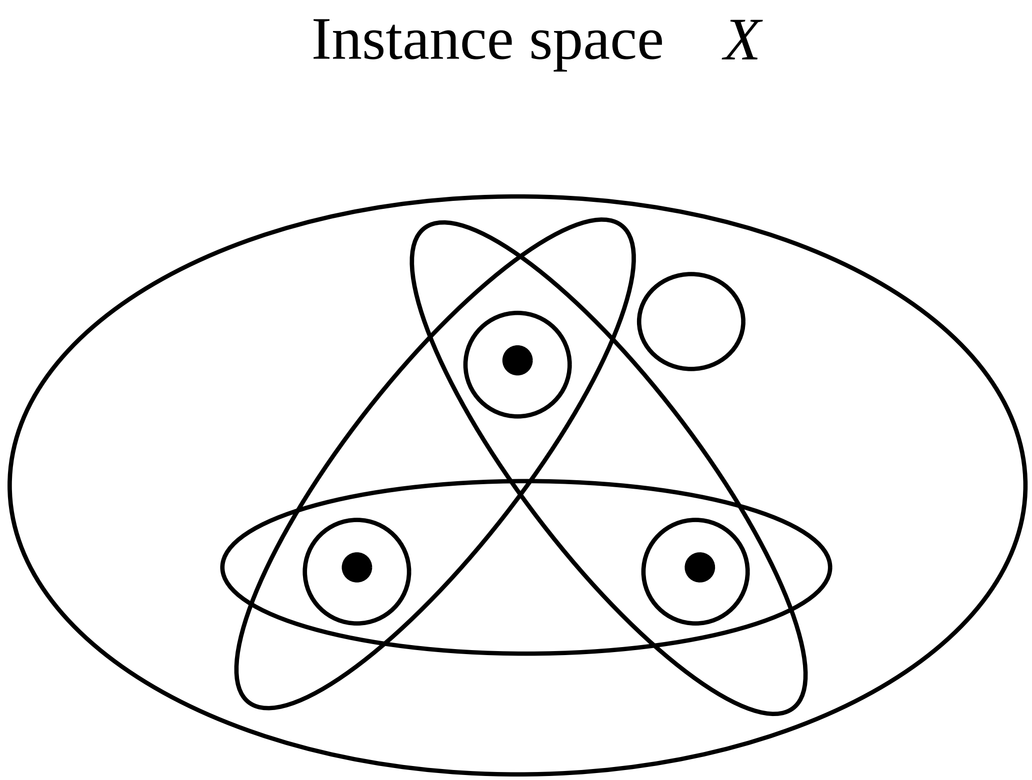
What if $|H|$ is infinite?

Shattering a Set of Instances

Definition: A **dichotomy** of a set S is a partition of S into two disjoint subsets.

Definition: A set of instances S is **shattered** by hypothesis space H if and only if for every dichotomy of S there exists some hypothesis in H consistent with this dichotomy.

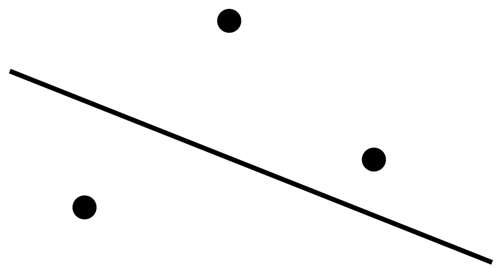
Three Instances Shattered



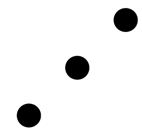
The Vapnik-Chervonenkis Dimension

Definition: The **Vapnik-Chervonenkis dimension**, $VC(H)$, of hypothesis space H defined over instance space X is the size of the largest finite subset of X shattered by H. If arbitrarily large finite sets of X can be shattered by H, then $VC(H) \equiv \infty$.

VC Dimension of Linear Decision Surfaces



(a)



(b)

For an n-dimensional space, VC dimension of linear estimators is $n+1$.