NEURAL NETWORKS 2

Today

- Reading
  - AIMA 18.6-18.8
- Announcements
  - Final project/HW5
  - Sandbox/Resources
  - Extra credit
- Goals
  - Feed-forward neural networks
  - Backpropagation
  - Naïve Bayes classifiers

Supervised learning

$$\mathbf{x}_i \longrightarrow \boxed{f} \longrightarrow y_i$$

- Training set

$$D = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, N\} \quad \text{where} \quad f(\mathbf{x}_i) = y_i$$

- Hypothesis class

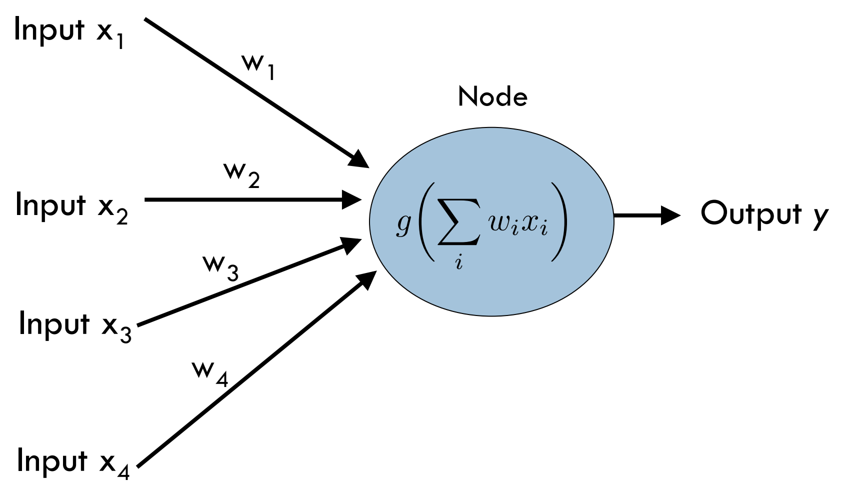
$$h \in \mathcal{H}$$

- Given training set, we want to find the hypothesis in the hypothesis class that "best approximates" f

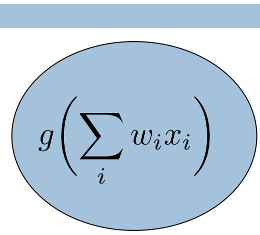
Supervised Learning terminology

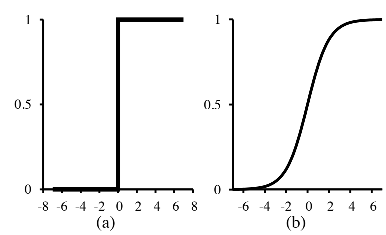
- Regression
  - y is a real-valued number
  - e.g. price of a commodity, pollution levels, brain activity
- Classification
  - y is a discrete (categorical) value
  - e.g. spam or not spam, 5-star ratings
- Structured prediction
  - y is a structured object
  - e.g. given sentence predict parse tree, given words in a sentence predict POS tags

A single perceptron



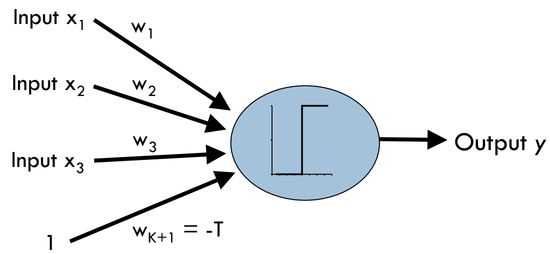
Activation function





- The activation function determines if the "electrical signal" entering the neuron is sufficient to cause it to fire
  - Threshold function – range is {0,1}
  - Sigmoid function – range [0,1]
  - Hyperbolic tangent function – range [-1,1]

Threshold versus "dummy" variable

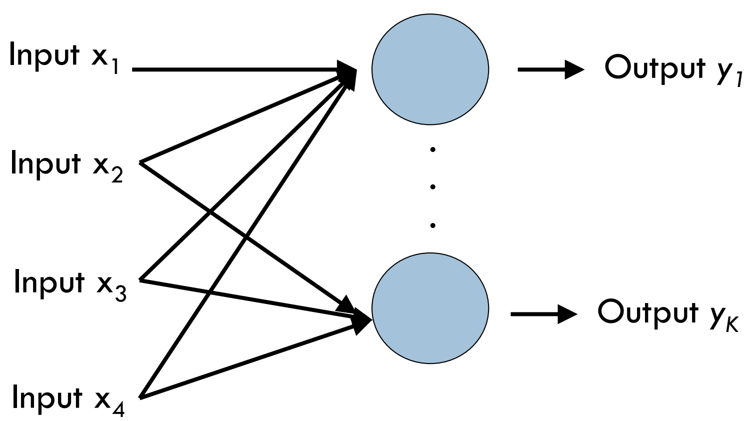


- Having a threshold T is equivalent to creating a "dummy" variable with value always 1

$$\sum_i x_i w_i \geq T \Longrightarrow 1$$

$$\sum_i x_i w_i - T \geq 0 \Longrightarrow 1$$

A perceptron network

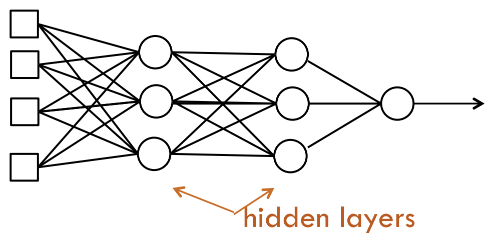


Reduces to K independent perceptrons

Beyond perceptrons

- Feed-forward neural network
  - Forms a directed acyclic graph (DAG) structure
  - Any continuous function of the inputs can be represented using a sufficiently large hidden layer
- Recurrent neural network
  - The output is fed back into the inputs
  - Creates a dynamical system that can have "short-term memory"

Interesting project idea!



Backpropagation

1. Begin with randomly initialized weights
2. Apply the neural network to each training example (each pass through examples is called an epoch)
3. If it misclassifies an example **modify the weights**
4. Continue until the neural network classifies all training examples correctly

(Derive gradient-descent update rule)

Backpropagation

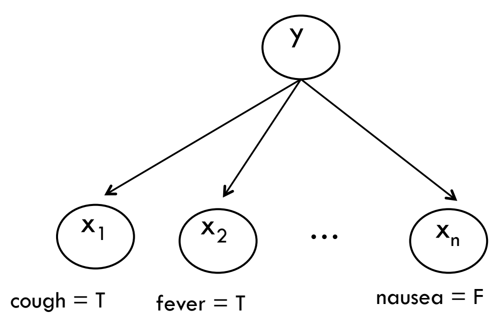
function BACK-PROP-LEARNING(*examples*, *network*) **returns** a neural network
  inputs: *examples*, a set of examples, each with input vector **x** and output vector **y**
    network, a multilayer network with L layers, weights $w_{i,j}$, activation function g
  local variables: Δ, a vector of errors, indexed by network node

  repeat
    for each weight $w_{i,j}$ in *network* **do**
      $w_{i,j} \leftarrow$ a small random number
    for each example $(\mathbf{x}, \mathbf{y})$ **in** *examples* **do**
      /* *Propagate the inputs forward to compute the outputs* */
      for each node i in the input layer **do**
        $a_i \leftarrow x_i$
      for $\ell = 2$ **to** L **do**
        for each node j in layer ℓ **do**
          $in_j \leftarrow \sum_i w_{i,j}\, a_i$
          $a_j \leftarrow g(in_j)$
      /* *Propagate deltas backward from output layer to input layer* */
      for each node j in the output layer **do**
        $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$
      for $\ell = L - 1$ **to** 1 **do**
        for each node i in layer ℓ **do**
          $\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j}\, \Delta[j]$
      /* *Update every weight in network using deltas* */
      for each weight $w_{i,j}$ in *network* **do**
        $w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
  until some stopping criterion is satisfied
  return *network*

Figure 18.24 The back-propagation algorithm for learning in multilayer networks.

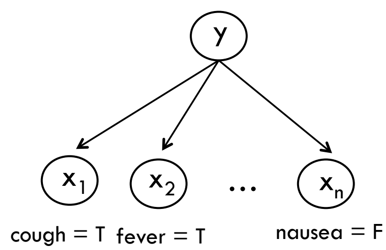
Naïve Bayes Classifier

- Naïve Bayes Classifier
  - Use for classification (i.e. y is categorical)
  - E.g., y = {Flu, Pneumonia, Appendicitis,…}



Naïve Bayes Classifier

- What are the independencies represented by this Bayesian network?
  - Also called the Idiot Bayes classifier
- For this to be a valid Bayesian network, what distributions do we need to define?
  - p(Y) = The prior distribution over the possible classes
  - $p(X_i | Y)$ = The conditional distribution of symptom given illness



Naïve Bayes Classifier

- We're interested in computing the quantity:

$$\begin{aligned}
&p(Y|x_1, x_2, \ldots, x_n) \\
&\propto p(Y, x_1, x_2, \ldots, x_n) \\
&= p(x_1|x_2, \ldots, x_n, Y) \ldots p(x_n|Y)p(Y) \\
&= p(x_1|Y)p(x_2|Y) \ldots p(x_n|Y)p(Y) \\
&= p(Y) \prod_{i=1}^{n} p(x_i|Y)
\end{aligned}$$

Naïve Bayes Classifier

- So, given training data D={(x_i,y_i) | i = 1…n}, how do we estimate these probabilities?

$$
\begin{aligned}
&p(Y|x_1, x_2, \ldots, x_n) \\
&\propto p(Y, x_1, x_2, \ldots, x_n) \\
&= p(x_1|x_2, \ldots, x_n, Y) \ldots p(x_n|Y)p(Y) \\
&= p(x_1|Y)p(x_2|Y) \ldots p(x_n|Y)p(Y) \\
&= p(Y) \prod_{i=1}^{n} p(x_i|Y)
\end{aligned}
$$