Neural Networks

Introduction to
Artificial Intelligence
COS302
Michael L. Littman
Fall 2001

Administration

11/28 Neural Networks
Ch. 19 [19.3, 19.4]
12/03 Latent Semantic Indexing
12/05 Belief Networks
Ch. 15 [15.1, 15.2]
12/10 Belief Network Inference
Ch. 19 [19.6]

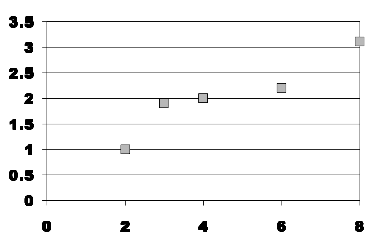
Proposal

11/28 Neural Networks
Ch. 19 [19.3, 19.4]
12/03 Backpropagation in NNs
12/05 Latent Semantic Indexing
12/10 Segmentation

Regression: Data

$x_1 = 2$ $y_1 = 1$
$x_2 = 6$ $y_2 = 2.2$
$x_3 = 4$ $y_3 = 2$
$x_4 = 3$ $y_4 = 1.9$
$x_5 = 4$ $y_5 = 3.1$
Given x, want to predict y.

Regression: Picture



Linear Regression

Linear regression assumes that the expected value of the output given an input E(y|x) is linear.
Simplest case:
out(x) = w x
for some unknown *weight* w.
Estimate w given the data.

1-Parameter Linear Reg.

Assume that the data is formed by

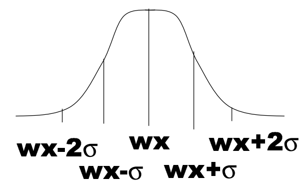
$y_i = w\ x_i$ + noise

where…

- the noise signals are indep.
- noise normally distributed: mean 0 and unknown variance σ^2

Distribution for ys

Pr(y|w, x) normally distributed with mean wx and variance σ^2



Data to Model

Fix xs. What w makes ys most likely? Also known as…

$\text{argmax}_w \Pr(y_1 \ldots y_n | x_1 \ldots x_n, w)$

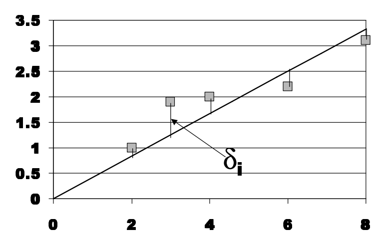
$= \text{argmax}_w \prod_i \Pr(y_i | x_i, w)$

$= \text{argmax}_w \prod_i \exp(-1/2\ ((y_i - wx_i)/\sigma)^2)$

$= \text{argmin}_w \sum_i (y_i - wx_i)^2$

Minimize sum-of-squared *residuals*.

Residuals



How Minimize?

$E = \sum_i (y_i - wx_i)^2$

$= \sum_i y_i^2 - (2 \sum_i x_i y_i)\ w + (\sum_i x_i^2)\ w^2$

Minimize quadratic function of w.

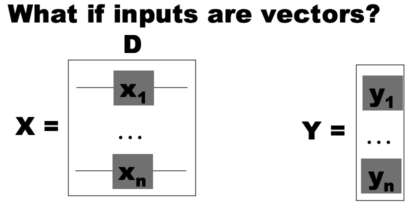
E minimized with

$w^* = (\sum_i x_i y_i) / (\sum_i x_i^2)$

so ML model is Out(x) = w* x.

Multivariate Regression

What if inputs are vectors?

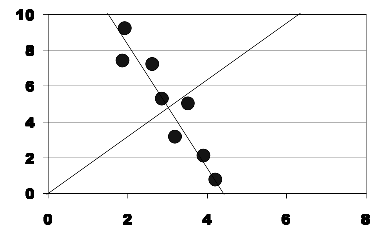


n data points, D components

Closed Form Solution

- Multivariate linear regression assumes a vector w s.t.
- Out(x) = $w^T x$
  = w[1] x[1] + … + w[D] x[D]
- ML solution: $w = (X^T X)^{-1} (X^T Y)$
- $X^T X$ is DxD, k,j elt is $\text{sum}_i\, x_{ij}\, x_{ik}$
- $X^T Y$ is Dx1, k elt is $\text{sum}_i\, x_{ik}\, y_i$

Got Constants?



Fitting with an Offset

- We might expect a linear function that doesn’t go through the origin.
- Simple obvious hack so we don’t have to start from scratch…

Gradient Descent

- Scalar function: f(w): $\Re \rightarrow \Re$
- Want a local minimum.
- Start with some value for w.
- Gradient descent rule:
  $w \leftarrow w - \eta\, \partial/\partial w\, f(w)$
- η “learning rate” (small pos. num.)
- Justify!

Partial Derivatives

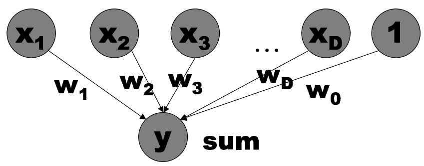
- $E = \text{sum}_k\, (w^T x_k - y_k)^2 = f(w)$
- $w_j \leftarrow w_j - \eta\, \partial/\partial w_j\, f(w)$
- How would a small increase in weight w_j change the error?
- Small positive? Large positive?
- Small negative? Large negative?

Neural Net Connection

- Set of weights w.
- Find weights to minimize sum-of-squared residuals. Why?
- When would we want to use gradient descent?

Linear Perceptron

Earliest, simplest NN.



Learning Rule

Multivariate linear function, trained by gradient descent.

Derive the update rule…

$out(x) = w^T x$

$E = sum_k (w^T x_k - y_k)^2 = f(w)$

$w_j \leftarrow w_j - \eta\, \partial/\partial w_j\ f(w)$

"Batch" Algorithm

1. Randomly initialize $w_1 \ldots w_D$
2. Append 1s to inputs to allow function to miss the origin
3. For i=1 to n, $\delta_i = y_i - w^T x_i$
4. For j=1 to D, $w_j = w_j + \eta\ sum_i\ \delta_i\ x_{ij}$
5. If $sum_i\ \delta_i^2$ is small, stop, else 3.

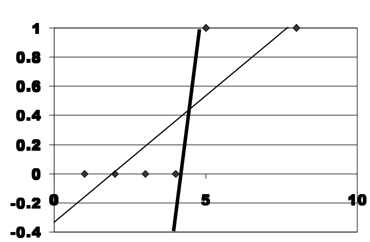
Why squared?

Classification

Let's say all outputs are 0 or 1.

How can we interpret the output of the perceptron as zero or one?

Classification



Change Output Function

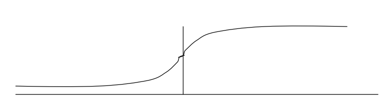
<u>Solution</u>:

Instead of $out(x) = w^T x$

we'll use

$out(x) = g(w^T x)$

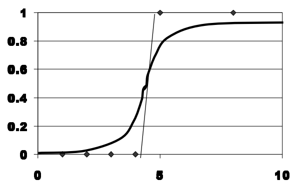
$g(x)$: $\Re \rightarrow (0,1)$, squashing function



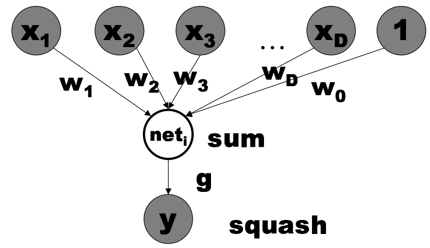
Sigmoid

$E = \text{sum}_k\, (g(w^T x_k) - y_k)^2 = f(w)$

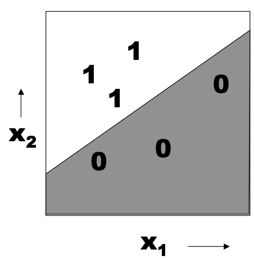
where $g(h) = 1/(1+e^{-h})$



Classification Percept.



Classifying Regions



Gradient Descent in Perceptrons

Notice $g'(h) = g(h)(1-g(h))$.

Let $net_i = \text{sum}_k\, w_k\, x_{ik}$, $\delta_i = y_i - g(net_i)$

$out(x_i) = g(net_i)$

$E = \text{sum}_i\, (y_i - g(net_i))^2$

$\partial E/\partial w_j = \text{sum}_i\, 2(y_i - g(net_i))\, (-\partial/\partial w_j\, g(net_i))$

$= -2\, \text{sum}_i\, (y_i - g(net_i))\, g'(net_i)\, \partial/\partial w_j\, net_i$

$= -2\, \text{sum}_i\, \delta_i\, g(net_i)\, (1-g(net_i))\, x_i$

Delta Rule for Perceptrons

- $w_j = w_j$
  $+ \eta\, \text{sum}_i\, \delta_i\, out(x_i)\, (1-out(x_i))\, x_{ij}$
- Invented and popularized by Rosenblatt (1962)
- Guaranteed convergence
- Stable behavior for overconstrained and underconstrained problems

What to Learn

- Linear regression as ML
- Gradient descent to find ML
- Perceptron training rule (regressions version and classification version)
- Sigmoids for classification problems

Homework 9 (due 12/5)

1. **Write a program that decides if a pair of words are synonyms using wordnet. I'll send you the list, you send me the answers.**
2. **Draw a decision tree that represents (a) $f_1+f_2+\ldots+f_n$ (or), (b) $f_1f_2\ldots f_n$ (and), (c) parity (odd number of features "on").**
3. **Show that $g'(h) = g(h)(1-g(h))$.**