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

\[ x_1 = 2 \quad y_1 = 1 \]
\[ x_2 = 6 \quad y_2 = 2.2 \]
\[ x_3 = 4 \quad y_3 = 2 \]
\[ x_4 = 3 \quad y_4 = 1.9 \]
\[ x_5 = 4 \quad y_5 = 3.1 \]

Given \( x \), want to predict \( y \).

Regression: Picture

![Graph showing regression data points and a linear trend](image)

Linear Regression

Linear regression assumes that the expected value of the output given an input \( E(y|x) \) is linear.

Simplest case:
\[ \text{out}(x) = w \times x \]
for some unknown weight \( w \).

Estimate \( w \) given the data.
1-Parameter Linear Reg.
Assume that the data is formed by
\[ y_i = w x_i + \text{noise} \]
where...
- the noise signals are indep.
- noise normally distributed: mean 0 and unknown variance \( \sigma^2 \)

Data to Model
Fix xs. What w makes ys most likely?
Also known as...
\[ \arg\max_w \Pr(y; y_0, \ldots, y_m | \ldots, x_m, w) \]
\[ = \arg\max_w \prod_i \Pr(y_i | x_i, w) \]
\[ = \arg\max_w \prod_i \exp(-1/2 (y_i - wx_i)^2) \]
\[ = \arg\min_w \sum_i (y_i - wx_i)^2 \]
Minimize sum-of-squared 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 \( \text{Out}(x) = w^* x \).

Distribution for ys
\[ \Pr(y|w, x) \] normally distributed
with mean \( wx \) and variance \( \sigma^2 \)

Residuals

Multivariate Regression
What if inputs are vectors?
\[ X = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix} \]
\[ Y = \begin{bmatrix} y_1 & \ldots & y_n \end{bmatrix} \]
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 \( D \times D \), \( k,j \) elt is sum\( _i x_i x_{ik} \)
\( X^T Y \) is \( D \times 1 \), \( k \) elt is sum\( _i x_{ik} y_i \)

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...

Got Constants?

Gradient Descent

Scalar function: \( f(w): \mathbb{R} \rightarrow \mathbb{R} \)
Want a local minimum.
Start with some value for \( w \).
Gradient descent rule:
\[ w \leftarrow w - \eta \frac{\partial}{\partial w} f(w) \]
\( \eta \) “learning rate” (small pos. num.)
Justify!

Partial Derivatives

\[ E = \text{sum}_k (w^T x_k - y_k)^2 = f(w) \]
\[ w_j \leftarrow w_j - \eta \frac{\partial}{\partial w_j} f(w) \]
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.

\[
\begin{array}{cccccc}
 x_1 & x_2 & x_3 & \ldots & x_D & 1 \\
 w_1 & w_2 & w_3 & \ldots & w_D & w_0 \\
 \text{sum} & & & & & \ \\
 y & & & & & \\
\end{array}
\]

### Learning Rule

Multivariate linear function, trained by gradient descent.

Derive the update rule...

\[
\text{out}(x) = w^Tx \\
E = \text{sum}_n (w^Tx_n - y_n)^2 = f(w) \\
w_j \leftarrow w_j - \eta \frac{\partial}{\partial w_j} f(w)
\]

### “Batch” Algorithm

1. Randomly initialize \(w, \ldots, w_0\)
2. Append 1s to inputs to allow function to miss the origin
3. For \(i=1\) to \(n\), \(\hat{y}_i = y_i - w^T x_i\)
4. For \(j=1\) to \(D\), \(w_j = w_j + \eta \sum_i \hat{y}_i x_{ij}\)
5. If \(\sum_i \hat{y}_i^2\) 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?

### Change Output Function

Solution:

Instead of \(\text{out}(x) = w^Tx\)

we’ll use

\[
\text{out}(x) = g(w^Tx) \\
g(x): \mathbb{R} \rightarrow (0,1), \text{ squashing function}
\]
**Sigmoid**

\[ E = \sum_k (g(w^T x_k) - y_k)^2 = f(w) \]

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

**Classification Percept.**

- Input features: \( x_1, x_2, \ldots, x_d, 1 \)
- Weights: \( w_1, w_2, \ldots, w_d, w_0 \)
- Net: \( \text{net} = \sum w_i x_i \)
- Output: \( y = g(\text{net}) \)
- Squash function

**Classifying Regions**

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Gradient Descent in Perceptrons**

\[ \text{Notice } g'(h) = g(h)(1-g(h)). \]

Let \( \text{net}_i = \sum w_k x_i \), \( \delta_i = y_i - g(\text{net}_i) \)

\[ \text{out}(x_i) = g(\text{net}_i) \]

\[ E = \sum_i (y_i - g(\text{net}_i))^2 \]

\[ \frac{\partial E}{\partial w_j} = \sum_i 2(y_i - g(\text{net}_i))(\frac{\partial g(\text{net}_i)}{\partial w_j}) = -2 \sum_i (y_i - g(\text{net}_i)) g'(\text{net}_i) \frac{\partial g(\text{net}_i)}{\partial w_j} \]

**Delta Rule for Perceptrons**

\[ w_j = w_j + \eta \sum \delta_i \text{out}(x_i)(1-\text{out}(x_i)) x_i \]

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)
- Sigmoid functions 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) $f_1+f_2+...+f_n$ (or), (b) $f_1f_2...f_n$ (and), (c) parity (odd number of features “on”).
3. Show that $g'(h) = g(h)(1-g(h))$. 