Announcements
- Hidden Figures: Tonight 6:30 + through Sunday
- HW3: due date delayed to Monday at noon
  problem 3: shouldn't have more than ~20 clauses

Today
Textbook Sections 18.4.2, 18.4.3, 18.6, 18.7

Background:
Classification learning problem

Given: \( D = (\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), \ldots, (\tilde{x}_n, \tilde{y}_n) \)  

Find: \( h(\tilde{x}) \approx \tilde{y} \)  

\( \Rightarrow h(\tilde{x}) \) with low error

Example of an error function on an example \( \tilde{x} \)  
\( E_h(\tilde{x}) = (\tilde{y} - h(\tilde{x}))^2 \)  — also called “loss”  
\( = \text{Loss}_h(\tilde{x}) \)

Error of a hypothesis \( h \) across all possible \( \tilde{x} \)  
\( E_h = \int_{\tilde{x}} E_h(\tilde{x}) \, p(\tilde{x}) \, d\tilde{x} \)  — Error weighted by probability of seeing \( \tilde{x} \)

Empirical error on data  
\( \text{Error}_h = \frac{1}{m} \sum_{i=1}^{m} E_h(\tilde{x}^{(i)}) \)

Common approach to classifier learning:  
Find an \( h \) with minimal empirical error  
\( \text{argmin}_h \text{Error}_h \)
Returning to perceptions

Linear Separability

A set of data is linearly separable if

\[ \exists \mathbf{w} \text{ such that for all } i \quad h_\mathbf{w}(\mathbf{x}_i) = f(\mathbf{x}_i) \]

(or, stated in terms of empirical error

\[ \exists \mathbf{w} \text{ such that } \text{Error}_{h_\mathbf{w}} = 0 \]

For linear classifiers we'll write Error_\mathbf{w} rather than Error_{h_\mathbf{w}}

Surprising Feature of Perceptrons

1. Representation analogous to a neuron
2. Learns similar to a neuron
3. Can prove it learns (sometimes)!

Perception Convergence Theorem

If there is a \( \mathbf{w} \) such that Error_\mathbf{w} = 0

then there is an \( \mathbf{w}' \) such that the perception learning rule will find a \( \mathbf{w}' \) for which Error_{\mathbf{w}'} = 0

\[ [ \text{For any } \mathbf{x}, \text{ if } \alpha \text{ decays as } O \left( \frac{1}{t} \right) \text{ where } t \ ]
\[ \text{is how many updates have taken place - for example, } \]
\[ \frac{100}{1000+} \text{ - then the perception will converge] } \]

Recall that the update rule is

\[ w_j \leftarrow w_j + \alpha x (f(x) - h(x)) \]

Momentary digression to linear regression...
Formula for a line: 
\[ f(x) = w_1 x + w_0 \]

Regression:

Given \((x_1', y_1'), \ldots, (x_m', y_m')\)

Find \(w_0, w_1\) that minimizes

\[
\frac{1}{m} \sum_{i=1}^{m} (y_i - (w_1 x_i' + w_0))^2
\]

\[ = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i(x))
\]

Solution:

\[ w_1 = \frac{m (\sum x_i y_i') - (\sum x_i')(\sum y_i')}{m (\sum x_i^2') - (\sum x_i')^2}
\]

\[ w_0 = \frac{\sum y_i' - w_1 \sum x_i'}{m}
\]

Can find the \(w_0 + w_1\) without search

Generalizes to \(\hat{x} = (x_1, \ldots, x_n)\) \(\hat{w} = (w_0, w_1, \ldots, w_n)\)

Can also learn them from the data using gradient descent on the error function

Repeat

For \(j = 0\) to \(n\) \(\langle\text{update each weight by a small amount}\rangle\)

\[ w_j \leftarrow w_j - \alpha \frac{\partial \text{Loss}}{\partial w_j} \]

(Recall that Loss is another term for Error)

Until \(\langle\text{stopping criterion}\rangle\)
For linear regression

\[ \text{Loss}_w (\bar{x}) = (f(\bar{x}) - h_w (\bar{x}))^2 \]

and the update rule simplifies to

\[ w_j \leftarrow w_j - \alpha \sum_{i=1}^{m} \bar{x}_i (f(\bar{x}_i) - h_w (\bar{x}_i)) \]

This is a "batch" update rule, in that it computes the error on all data for each update.

Can instead do this incrementally, updating weights on a per example basis

**Known as Stochastic Gradient Descent**

Repeat

For \( i = 1 \) to \( m \) \(< \text{iterate over the data} >\)

For \( j = 0 \) to \( n \) \(< \text{iterate over the features} >\)

\[ w_j \leftarrow w_j - \alpha \frac{\partial \text{Loss}_w (\bar{x}_i)}{\partial w_j} \]

Until \(<\text{stopping criterion}>\)

[Typically reorder the data each time]

If \[ \text{Loss}_w (\bar{x}) = (f(\bar{x}) - h_w (\bar{x}))^2 \]

the update simplifies to

\[ w_j \leftarrow w_j - \alpha \bar{x}_j (f(\bar{x}_i) - h_w (\bar{x}_i)) \]

Same as the Perceptron Learning Rule

**Stochastic Gradient Descent** is one of the key methods in many forms of deep learning today.
Perception problems

1. Problem: Can't represent XOR and other functions.
   Solution: Multilayer networks.

2. Problem: Unclear how to train lower layers.
   Solution: Approximate threshold units.

\[ h_{\vec{w}}(\vec{x}) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{x}}} \]