Neural nets unit:

- General introduction (last week Wednesday)
- Trying out neural nets (last week Friday, in lab)
- How to train your perceptron (Monday)
- The feed-forward and back-propagation algorithms (**Today**)
- Deep learning: CNNs (Friday and next week Monday)
- Deep learning in practice (next week Wednesday, in lab)

Last time and today:

- Implementing a perceptron
- Training a perceptron
- Implementing an MLP (the feed-forward algorithm)
- Training an MLP (the back-propagation algorithm)

A **perceptron** is a function $\mathbb{R}^D \to \mathbb{R}$ defined as

$$p(\boldsymbol{x}) = h(\boldsymbol{\theta} \cdot \boldsymbol{x} + b) = h\left(b + \sum_{i=0}^{D-1} \theta_i x_i\right)$$

where

- $\boldsymbol{\theta}$ is the vector of weights
- b is the bias term
- h is the activation function

Let $\boldsymbol{t}$ be the target values, that is t_n is the target value for data point $\boldsymbol{x}_n$. We're using t instead of y so that y can be used for the output unit of an MLP. Let η be the learning rate.

For a single perceptron with weights $\boldsymbol{\theta}$, the weights are updated based on data point $\boldsymbol{x}$ with target t by the *perceptron (training) rule*:

$$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} + \eta \underbrace{(t - p(\boldsymbol{x}))}_{\text{error}} \boldsymbol{x}$$

Or, applied to a single feature/dimension i,

$$\theta_i^{\text{new}} = \theta_i^{\text{old}} + \eta \underbrace{(t - p(\boldsymbol{x}))}_{\text{error}} x_i$$

Why multiply the error by x?

"Neuron" perspective on the perceptron rule. $\theta_i^{\text{new}} = \theta_i^{\text{old}} + \eta \underbrace{(t - p(\boldsymbol{x}))}_{\text{error}} x_i$

Target value $t \in \{0, 1\}$ indicates whether the neuron should fire, computed value $p(\boldsymbol{x})$ indicates whether the neuron does fire. Interpret $(t - p(\boldsymbol{x}))$:

- 0 means the perceptron was correct
- -1 means the perceptron fired when it shouldn't have fired. (Threshold too low.)
  - Decrease θ_i if x_i is positive
  - Increase θ_i if x_i is negative
- 1 means the perceptron didn't fire when it should have fired. (Threshold too high.)
  - Increase θ_i if x_i is positive
  - Decrease θ_i if x_i is negative

The intensity of the signal (magnitude of x_i) affects how much the weight is changed.

“Loss” perspective on the perceptron rule. $\theta_i^{\text{new}} = \theta_i^{\text{old}} + \eta \underbrace{(t - p(\boldsymbol{x}))}_{\text{error}} x_i$

$$\mathcal{L}(\boldsymbol{\theta}) = \tfrac{1}{2}(t - p(\boldsymbol{x}))^2$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i} \frac{1}{2}(t - p(\boldsymbol{x}))^2 \\
&= (t - p(\boldsymbol{x}))\ \frac{\partial}{\partial \theta_i}(t - p(\boldsymbol{x})) \\
&= -(t - p(\boldsymbol{x}))\ \frac{\partial}{\partial \theta_i}\,(\theta_0 x_0 + \cdots \theta_D x_D) \\
&= -(t - p(\boldsymbol{x})) x_i
\end{aligned}$$

Wait, what about the negative sign?

The **feed-forward algorithm**:
Given input x, let $\boldsymbol{z}^{(m)}$ be (the output of) the mth hidden layer.

Let $\boldsymbol{z}^{(0)} = \boldsymbol{x}$
For each $m \in [1, M]$ (for each hidden layer)
    For each hidden unit $z_k^{(m)}$ in layer $\boldsymbol{z}^{(m)}$
        Apply $z_k^{(m)}$ to $\boldsymbol{z}^{(}m-1)$
Apply output unit y to $\boldsymbol{z}^{(}M)$

The output of the feed-forward algorithm (distinct from the output of the MLP) is the results of all the units of all the layers.

M is the number of hidden units. j and ℓ range over hidden units, as in z_j. We are assuming a single output unit y.

The weights in the output unit are θ_{yj}, that is, the jth weight (corresponding to the hidden unit z_j) in output unit y.

The weights of the hidden units are θ_{ji}, that is, the $i$the weight (corresponding to the input component x_i) in hidden unit z_j.

The sum of squares error, as a function of the parameters (weights) $\boldsymbol{\theta}$ is

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{n=0}^{N-1} (y(\boldsymbol{x}_n) - t_n)^2$$

Or, applied to a single data point $\boldsymbol{x}, t$

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} (y(\boldsymbol{x}) - t)^2$$

To simplify the notation, let y stand in for the result of output unit y when the MLP is applied to $\boldsymbol{x}$.
Let $\boldsymbol{\theta}_y$ be the weight vector of output unit y. Let j index into $\boldsymbol{\theta}_y$. Let σ (logistic function) be the activation function.
Finding the partial derivative with respect to θ_{yj}.

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \tfrac{1}{2}(y-t)^2 \\
\frac{\partial \mathcal{L}}{\partial \theta_{yj}} &= \frac{\partial}{\partial \theta_{yj}}\left(\tfrac{1}{2}(y-t)^2\right) = (y-t)\frac{\partial}{\partial \theta_{yj}}(y-t) = (y-t)\frac{\partial}{\partial \theta_{yj}} y \\
&= (y-t)\ \frac{\partial}{\partial \theta_{yj}}\ \underbrace{\sigma\left(\sum_{\ell=0}^{M} \theta_{y\ell} z_\ell\right)}_{y} \\
&= (y-t)\quad \underbrace{y\ (1-y)}_{\text{from derivative of } \sigma}\quad \frac{\partial}{\partial \theta_{yj}}\left(\textstyle\sum_{\ell=0}^{M} \theta_{y\ell} z_\ell\right) = (y-t)\ y\ (1-y)\ z_j
\end{aligned}
$$

Let $\boldsymbol{z}_j$ be the a hidden unit, and let $\boldsymbol{\theta}_j$ be the weight vector of that unit. Let θ_{ji} be the ith weight of the jth hidden unit.
Finding the partial derivative with respect to θ_{ji}:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta_{ji}} &= (y-t)\frac{\partial}{\partial \theta_{ji}} y = (y-t)\; y\; (1-y)\frac{\partial}{\partial \theta_{ji}}\left(\sum_{\ell=0}^{M} \theta_{y\ell} z_\ell\right) \\
&= (y-t)\; y\; (1-y)\frac{\partial}{\partial \theta_{ji}}(\theta_{yj}\boldsymbol{z}_j) \\
&= (y-t)\; y\; (1-y)\; \theta_{yj}\frac{\partial}{\partial \theta_{ji}}\boldsymbol{z}_j \\
&= (y-t)\; y\; (1-y)\; \theta_{yj}\frac{\partial}{\partial \theta_{ji}}\sigma\left(\sum_{i=0}^{D} \theta_{ji} x_i\right) \\
&= (y-t)\; y\; (1-y)\; \theta_{yj} \quad \underbrace{z_j\; (1-z_j)}_{\text{from derivative of } \sigma} \quad \frac{\partial}{\partial \theta_{ji}}\left(\sum_{i=0}^{D} \theta_{ji} x_i\right) \\
&= (y-t)\; y\; (1-y)\; \theta_{yj}\; z_j\; (1-z_j)\; x_i
\end{aligned}$$

Coming up:

Due Wed, Apr 9:
Read and respond to two articles about bias in algorithms
(See Canvas)

Due Fri, Apr 11: *Read excerpt from Geron introducing convolutional neural nets*
(See Canvas)

Due Wed, Apr 16:
Implement perceptron training, feed-forward, and back-propagation
(You know enough to do the first part)

Sometime between Mar 31 and Apr 17:
Make an office-hours appointment for project check-in
(Originally the deadline was Apr 11)