Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>modelling a continuous function</td>
</tr>
<tr>
<td>Classification</td>
<td>modelling a discrete categorical function</td>
</tr>
<tr>
<td>Supervised Learning</td>
<td>learning a function from (x,y) pairs</td>
</tr>
<tr>
<td>Unsupervised Learning</td>
<td>learning a structure from a data set X</td>
</tr>
<tr>
<td>Overfitting</td>
<td>model become overly complex</td>
</tr>
<tr>
<td>Underfitting</td>
<td>model is not complex enough</td>
</tr>
<tr>
<td>Logistic Function</td>
<td>(\log(1 + e^x))</td>
</tr>
<tr>
<td>Early Stopping</td>
<td>stop training when validation error starts to increase</td>
</tr>
<tr>
<td>Momentum</td>
<td>(\dot{v}<em>t = \mu \dot{v}</em>{t-1} + \eta (\nabla f(a)))</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>L2 regularization on weights</td>
</tr>
</tbody>
</table>

Convolutional Neural Networks (CNNs)

- Used for image/vision tasks

Activation Functions

Logistic Function:
\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]

Hyperbolic Tangent:
\[
\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]

Rectified Linear Units (ReLU):
\[
\sigma(x) = \max(0, x)
\]

Announcements

- No class Monday (Thanksgiving)
- Assignment 4 due Wednesday (October 15th)
- Assignment 2 due Wednesday (October 15th)
initialization is important  
\( \text{say } \text{ReLU (}-100, 100) \)  
will \text{we refer near } "\text{scale}\" \text{of sigmoid (}-1, 1)\)  
(\text{"adversarial" it you’re far away})  
vanishing gradient; \text{gradients less than 0.1 going back layers, learning will be slow because steps will be small}  
relu: not "scaled" with size  

\[
\begin{align*}
\lambda_1 & \quad \lambda_2 \\
\theta & \quad \theta -5 \\
\theta & \quad 0 \\
\theta & \quad 0
\end{align*}
\]

\text{derivative can flow through another path}  

\text{Overfitting and Regularization}  
\text{getting Weird of peculiarities in the training set}  
\text{something like } -250x^3 - 992x^2 + 2594x + 1000 \text{ (canals perfectly at points)}  

\text{training: too high polynomial}  
\text{validation: (minimize error)}  

\text{to try and fix this:}  
\text{Regularization:}  
\begin{equation}
L = \sum_i (y_i - \hat{y}_i)^2 + \lambda \sum |\theta_i|
\end{equation}

\Rightarrow \text{"Weight Decay"}  
\text{\# make weights small}  
\text{\# helps prevent overfitting}  
\text{\# could also use L1}  

\text{Dropout (a regularization method) (2012)}  
\* \text{During training, at every iteration, "drop out" (set to zero) each unit with probability } p.  
\* \text{During prediction, multiply weights by } (1 - p)  

\text{\#\# this value was overfitting with a huge value}  
\text{\#\# with a bunch of different neural networks}
Optimization

- Use Stochastic Gradient Descent
  \[ W_{t+1} = W_t - \alpha_t \nabla L(W_t) \]
  \[ \text{learning rate} \]
  \[ \text{gradient of loss} \]
- Momentum
  \[ \dot{V}_t = \varepsilon V_{t-1} + \nabla L(W_t) \]
  \[ W_{t+1} = W_t - \alpha_t L(W_t) + \dot{V}_t \]
  \[ \text{retain some momentum from previous step} \]

"Second order method"

Example with a ball

- Shallow basin, local minimum
- Derivative 0