Feed-forward Neural Networks

\[ f \circ f = g^{(3)}(W^{(3)}h^{(2)} + b^{(3)}) \]

\[ h^{(2)} = g^{(2)}(W^{(2)}h^{(1)} + b^{(2)}) \]

\[ h^{(1)} = g^{(1)}(W^{(1)}x + b^{(1)}) \]

- When \( f \) is a scalar, \( W^{(3)}h^{(2)} = w^{(3)}h^{(2)} \)

- Other architectures possible:
  - "skip connections"
  - Parameterize the \( g \)'s non-linearities

- Special layers for images/audio
  - Conv Nets ... and others.
Initialization

Don't set all weights the same.

Naive code: \( W_{ij}^{(l)} \sim N(0, 1) \) (use \texttt{randn}())

\[
h_k^{(l)} = g \left( \frac{1}{k^{(l-1)}} + b_k^{(l)} \right)
\]

\( g = \sigma(a) \)

Typically how big is this sum over \( k^{(l-1)} \)?

Background on expectations:

Typically sum \( \sim \pm \sqrt{K^{(l-1)}} \)

Example Initialization \( w \sim N(0, (\frac{1}{\sqrt{k}})^2) \)

MLP: More sophisticated suggestions.
NN don't ever have a convex cost

Convex \Rightarrow\text{Unique optimum}

Set \( w^{(1)} \) to their best values.

- These local optima don't matter
  \rightarrow \text{because the function (predictions) are the same.}

- Not all optima are equivalent.
- Use heuristics to get good fits.
- Random restarts. -- not if network large/expensive.
Regularization

Could do L2 regularization

Set $\lambda$... cross-validate.

Cost

![Graph showing training and validation cost over iterations.](image)

This gap is not "overfittingness".

Every $k$ updates:

If val. cost the smallest I've seen:
- Store the weights & val. cost

If val. cost hasn't improved in 20 updates:
- Stop. Return the weights we stored from best val. score.
Getting gradients - Reverse-mode differentiation

Backpropagation

\[ L(f, y) \]

\[ f \]

\[ y \]

\[ a^{(2)} \]

\[ W^{(2)} \]

\[ b^{(2)} \]

\[ a^{(1)} \]

\[ h^{(1)} \]

\[ b^{(1)} \]

\[ W^{(1)} \]

\[ h^{(0)} \]

\[ b^{(0)} \]

\[ x \]

Strategic:

\[ \frac{\partial L}{\partial f} = \frac{\partial c}{\partial f} \]

\[ \frac{\partial L}{\partial a_i^{(2)}} = \frac{\partial c}{\partial a_i^{(2)}} \]

\[ \frac{\partial L}{\partial W^{(2)}} = \frac{\partial c}{\partial W^{(2)}} \]

\[ \frac{\partial L}{\partial b^{(2)}} = \frac{\partial c}{\partial b^{(2)}} \]

\[ \frac{\partial L}{\partial W_{i,j}^{(1)}} = \frac{\partial c}{\partial W_{i,j}^{(1)}} \]

For every intermediate \( \theta \)

get \( \theta = \frac{\partial c}{\partial \theta} \)
Derivative Propagation

\[ \ldots \rightarrow u \xrightarrow{f} w \rightarrow \ldots \rightarrow c \]

Assume we have

\[ \overline{w} = \frac{\partial c}{\partial w} \]

Want:

\[ \overline{u} = \frac{\partial c}{\partial u}, \quad \overline{v} = \frac{\partial c}{\partial v} \]

Chain rule:

\[ \frac{\partial c}{\partial u} = \frac{\partial c}{\partial w} \frac{\partial w}{\partial u} \]

Number which is propagated to us.

Derivative of small local function. Look up expression of \( u, v \) (and/or \( w \)) to use.