Multi-layer neural networks and backpropagation
Beyond linear predictors

- To achieve good accuracy on challenging problems, we need to be able to train nonlinear models.

- Traditional “shallow” approach:
“Shallow” pipeline: Nonlinear SVM

- Perform a nonlinear mapping induced by kernel function, apply linear classifier
- Example: predictor for polynomial kernel of degree 2

\[ y = \text{sign}(w^T \phi(x) + b) \]

Source: Y. Liang
“Shallow” pipeline: Nonlinear SVM

• Perform a nonlinear mapping induced by kernel function, apply linear classifier

• Equivalently, compute kernel function value of input with every support vector, apply linear classifier

\[ y = \text{sgn} \left( \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \right) \]
Two-layer neural network

- Introduce a *hidden layer* of perceptrons computing linear combinations of inputs followed by a *nonlinearity*.

\[
g(w_{11}^T x) \quad g(w_{12}^T x) \quad g(W_1 x) \quad g(w_2^T g(W_1 x))
\]

Why do we need the nonlinearity?

\[
W_1 - \text{matrix with rows } w_{1j}^T
\]
Common nonlinearities

**Sigmoid**

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

**tanh**

\[ \tanh(x) \]

**ReLU**

\[ \max(0, x) \]

Source: Stanford 231n
Two-layer neural network

- Introduce a hidden layer of perceptrons computing linear combinations of inputs followed by a nonlinearity
- This gives a universal function approximator
- But the hidden layer may need to be huge
Beyond two layers
“Deep” pipeline

- Learn a feature hierarchy
- Each layer extracts features from the output of previous layer
- All layers are trained jointly
Multi-Layer network demo

http://playground.tensorflow.org/
How to train a multi-layer network?
How to train a multi-layer network?

- We need to find the gradient of the error w.r.t. the parameters of each layer, $\frac{\partial e}{\partial w_k}$, to perform updates

  $$w_k \leftarrow w_k - \eta \frac{\partial e}{\partial w_k}$$
Computation graph

\[ f_1(x, w_1) \rightarrow f_2(h_1, w_2) \rightarrow \ldots \rightarrow f_K(h_{K-1}, w_K) \rightarrow l(h_K, y) \rightarrow e \]
Chain rule

Let’s start with $k = 1$

\[ e = l(f_1(x, w_1), y) \]
\[ \frac{\partial}{\partial w_1} l(f_1(x, w_1), y) = \]

Example: \[ e = (y - w_1^T x)^2 \]
\[ h_1 = f_1(x, w_1) = w_1^T x \]
\[ e = l(h_1, y) = (y - h_1)^2 \]

\[ \frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_1} \frac{\partial h_1}{\partial w_1} \]
Chain rule

\[ k = 2 \]

\[ e = l(f_2(f_1(x, w_1), w_2)) \]

\[ \frac{\partial e}{\partial w_2} = \]
Chain rule

$$k = 2$$

$$e = l(f_2(f_1(x, w_1), w_2))$$

$$\frac{\partial e}{\partial w_2} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial w_2}$$

Example: $$e = -\log\left(\sigma(w_1^T x)\right)$$ (assume $$y = 1$$)

$$h_1 = f_1(x, w_1) = w_1^T x$$

$$h_2 = f_2(h_1) = \sigma(h_1)$$

$$e = l(h_2, 1) = -\log(h_2)$$

$$\frac{\partial e}{\partial w_1} = \frac{\partial e}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial w_1}$$
Chain rule

General case:

\[
\frac{\partial e}{\partial w_k} = \frac{\partial e}{\partial h_K} \frac{\partial h_K}{\partial h_{K-1}} \cdots \frac{\partial h_{K+1}}{\partial h_K} \frac{\partial h_K}{\partial w_k}
\]

Upstream gradient

Local gradient
Backpropagation summary

Parameter update:
\[
\frac{\partial e}{\partial w_k} = \frac{\partial e}{\partial h_k} \frac{\partial h_k}{\partial w_k}
\]

Upstream gradient:
\[
\frac{\partial e}{\partial h_k} = \frac{\partial e}{\partial h_{k-1}} \frac{\partial h_k}{\partial h_{k-1}}
\]

$W_k$

$h_{k-1}$

$h_k$

Local gradient

Local gradient

$f_k$
What about more general computation graphs?

ResNet

ResNeXt

Figure source
What about more general computation graphs?

+ Gradients add at branches

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]} \]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]] \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)\}] \]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]} \]

\[ \exp(-1) \times (-0.53) = -0.20 \]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]] \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]} \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]} \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]} \]

Source: Stanford 231n
A detailed example

$$f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]]$$

Source: Stanford 231n
A detailed example

$$f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]]$$

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)\right]} \]

Source: Stanford 231n
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]] \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]] \]
A detailed example

\[ f(x, w) = \frac{1}{1 + \exp[-(w_0 x_0 + w_1 x_1 + w_2)]} \]

Can simplify computation graph

Sigmoid gate \( \sigma(x) \)

\[
\sigma'(x) = \sigma(x)(1 - \sigma(x))
\]

\[
\sigma(1)(1 - \sigma(1)) = 0.73 \times (1 - 0.73) = 0.20
\]

Source: Stanford 231n
Patterns in gradient flow

Source: Stanford 231n
Patterns in gradient flow

Add gate: “gradient distributor”

Source: Stanford 231n
Patterns in gradient flow

Add gate: “gradient distributor”
Multiply gate: “gradient switcher”

Source: Stanford 231n
Patterns in gradient flow

Add gate: “gradient distributor”
Multiply gate: “gradient switcher”
Max gate: “gradient router”

Source: Stanford 231n
Dealing with vectors

\[
\frac{\partial Z}{\partial x} = \begin{pmatrix}
\frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_M} \\
\vdots & \ddots & \vdots \\
\frac{\partial z_N}{\partial x_1} & \cdots & \frac{\partial z_N}{\partial x_M}
\end{pmatrix}
\]

Jacobian

\[
\begin{align*}
\frac{\partial e}{\partial x} &= \frac{\partial e}{\partial z} \frac{\partial z}{\partial x} \\
1\times M & \times 1\times N & N\times M
\end{align*}
\]

\[
\frac{\partial e}{\partial z} \\
1\times N
\]
Simple case: Elementwise operation
Simple case: Elementwise operation

\[ f(x) = \max(0, x) \]

\[ \frac{\partial z}{\partial x} = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_M}{\partial x_1} & \cdots & \frac{\partial z_M}{\partial x_M} \end{pmatrix}_{M \times M} \]

1×M Jacobian
Simple case: Elementwise operation

$$\frac{\partial z}{\partial x} = \begin{pmatrix} \mathbb{1}[x_1 > 0] & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \mathbb{1}[x_M > 0] \end{pmatrix}$$

Jacobian

$$f(x) = \max(0, x)$$

$$\frac{\partial e}{\partial x} = \mathbb{1}[x > 0] \frac{\partial e}{\partial z}$$

$$\frac{\partial e}{\partial x_i} = \mathbb{1}[x_i > 0] \frac{\partial e}{\partial z_i}$$

$$\frac{\partial e}{\partial x} = \mathbb{1}[x > 0] \ast \frac{\partial e}{\partial z}$$
Matrix-vector multiplication

\[ f(x, W) = xW \]

\[ \frac{\partial e}{\partial W} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial W} \]

\[ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]
Matrix-vector multiplication

\[(z_1 \ldots z_N) = (x_1 \ldots x_M) \begin{pmatrix} W_{11} & \ldots & W_{1N} \\ \vdots & \ddots & \vdots \\ W_{M1} & \ldots & W_{MN} \end{pmatrix} \]

\[z_j = \sum_{i=1}^{M} x_i W_{ij}\]

Want:

\[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x}
\]

1×M  1×N  N×M

\[
\frac{\partial z_j}{\partial x_i} = \frac{\partial z}{\partial x} = W^T
\]

\[
\frac{\partial e}{\partial x} = \frac{\partial e}{\partial z} \frac{\partial z}{\partial x} = \frac{\partial e}{\partial z} W^T
\]

jth row, ith column of Jacobian
Matrix-vector multiplication

\[
(z_1 \ldots z_N) = (x_1 \ldots x_M) \begin{pmatrix} W_{11} & \ldots & W_{1N} \\ \vdots & \ddots & \vdots \\ W_{M1} & \ldots & W_{MN} \end{pmatrix}
\]

\[
z_j = \sum_{i=1}^{M} x_i W_{ij}
\]

Want:

\[
\frac{\partial e}{\partial W} = \frac{\partial e}{\partial z} \begin{bmatrix} \frac{\partial z}{\partial W} \end{bmatrix}
\]

\[
| \begin{array}{c} M \times N \\ 1 \times N \\ N \times (M \times N) \end{array} |
\]

\[
\frac{\partial z_k}{\partial W_{ij}} = \frac{\partial e}{\partial W_{ij}}
\]

\[
z_k \text{ depends only on } k^{th} \text{ column of } W
\]

\[
\frac{\partial e}{\partial W} = x^T \frac{\partial e}{\partial z}
\]
General tips

- Derive error signal (upstream gradient) directly, avoid explicit computation of huge local derivatives
- Write out expression for a single element of the Jacobian, then deduce the overall formula
- Keep consistent indexing conventions, order of operations
- Use dimension analysis

- **Useful resource:** see Lecture 4 of Stanford 231n and associated links in the syllabus