Convex Optimization

Lecture 16 - Softmax Regression and Neural Networks

Instructor: Yuanzhang Xiao

University of Hawaii at Manoa

Fall 2017

Today's Lecture

1. Softmax Regression

2. Neural Networks

Outline

1. Softmax Regression
2. Neural Networks

Softmax Regression

extend logistic regression to multi-class classification

training data $(a^{(i)}, b^{(i)}),\ \ i = 1, \ldots, m$

labels with K values: $b^{(i)} \in \{1, \ldots, K\}$

hypothesis:

$$h_x(a) = \begin{bmatrix} P(b = 1 \mid a; x) \\ \vdots \\ P(b = K \mid a; x) \end{bmatrix} = \frac{1}{\sum_{k=1}^{K} \exp(x^{(k)T} a)} \begin{bmatrix} \exp(x^{(1)T} a) \\ \vdots \\ \exp(x^{(K)T} a) \end{bmatrix}$$

parameters to learn:

$$x = [x^{(1)}, \ldots, x^{(K)}] \in \mathbb{R}^{n \times K}$$

Maximum Log-Likelihood Estimator

given training data $(a^{(i)}, b^{(i)})_{i=1}^{m}$, the probability of this sequence is

$$\prod_{i=1}^{m}\prod_{k=1}^{K}\left[\frac{\exp(x^{(k)T}a^{(i)})}{\sum_{j=1}^{K}\exp(x^{(j)T}a^{(i)})}\right]^{\mathbf{1}_{\{b^{(i)}=k\}}}$$

log-likelihood is

$$\ell(x)=\sum_{i=1}^{m}\sum_{k=1}^{K}\mathbf{1}_{\{b^{(i)}=k\}}\cdot\log\frac{\exp(x^{(k)T}a^{(i)})}{\sum_{j=1}^{K}\exp(x^{(j)T}a^{(i)})}$$

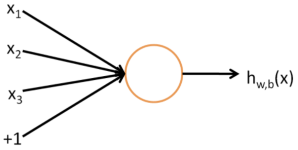
gradient is

$$\frac{\partial\ell(x)}{\partial x^{(k)}}=\sum_{i=1}^{m}a^{(i)}\cdot\left(\mathbf{1}_{\{b^{(i)}=k\}}-\frac{\exp(x^{(k)T}a^{(i)})}{\sum_{j=1}^{K}\exp(x^{(j)T}a^{(i)})}\right)$$

Outline

Neural Networks – A Single Neuron

fit a training example (x, y) with a neuron:



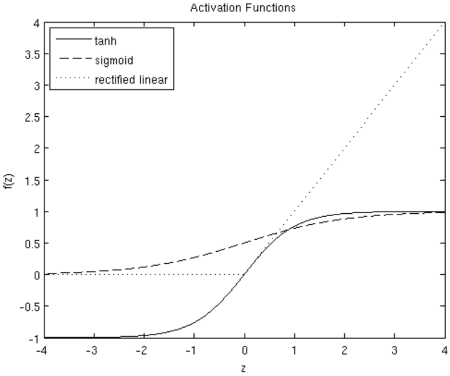
- input: x and a normalization term $+1$
- output: $h_{w,b}(x) = f(w^T x + b)$
- f is the activation function

some choices of activation function:

- sigmoid: $f(z) = \frac{1}{1+e^{-z}}$ (as in logistic regression)
- tanh: $f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- rectified linear: $f(z) = \max\{0, x\}$ (deep neural networks)

Neural Networks – Activation Functions

illustration of activation functions

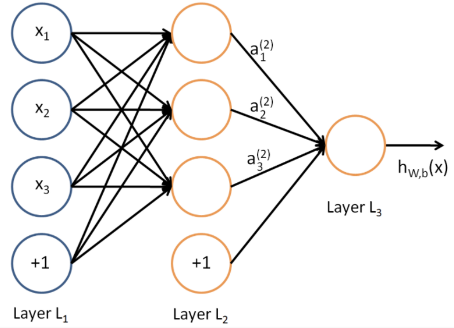


- tanh: rescaled sigmoid
- rectified linear: unbounded

Neural Networks – Basic Architecture

neural network: network of neurons

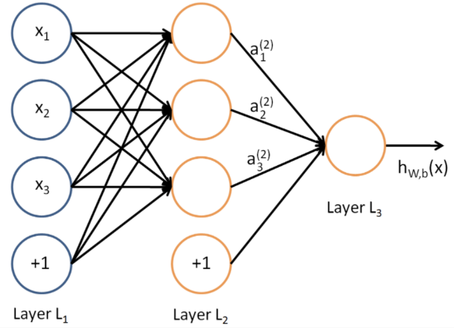
a three-layer neural network:



- input layer: the leftmost layer
- output layer: the rightmost layer
- hidden layer: the layers in the middle

Neural Networks – Parameters to Learn

a three-layer neural network:

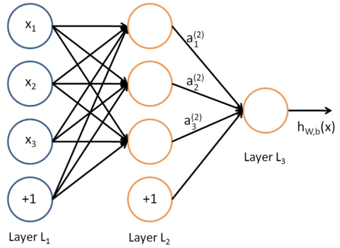


number of layers $n_\ell = 3$

weight of link from unit j in layer ℓ and unit i in layer $\ell + 1$: W_{ij}^{ℓ}

parameters to learn: $(W^{(1)}, b^{(1)}, \ldots, W^{(n_\ell)}, b^{(n_\ell)})$

Neural Networks – Example



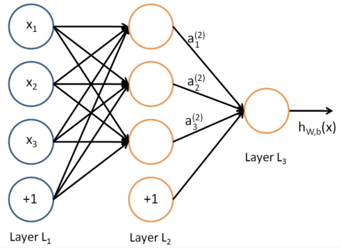
the activation (i.e., output) of unit i at layer ℓ: $a_i^{(\ell)}$

computation:

$$
\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_3^{(2)})
\end{aligned}
$$

Neural Networks – Compact Representation

a three-layer neural network:



define weighted sum of inputs to unit i in layer ℓ as

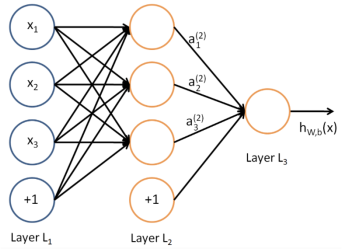
$$z_i^{(\ell)} = \sum_{j=1}^{n} W_{ij}^{(\ell-1)} a_j^{(\ell-1)} + b_i^{(\ell-1)},$$

where

$$a_j^{(\ell-1)} = f(z_j^{(\ell-1)})$$

Neural Networks – Compact Representation

a three-layer neural network:



compact representation: (forward propagation)

$$
\begin{aligned}
z^{(\ell+1)} &= W^{(\ell)} a^{(\ell)} + b^{(\ell)} \\
a^{(\ell+1)} &= f(z^{(\ell+1)})
\end{aligned}
$$

Neural Networks – Extensions

may have different architectures (i.e., network topology)

- different numbers s_ℓ of units in each layer ℓ
- different connectivity

may have loops

may have multiple output units

Neural Networks – Optimization

minimize the prediction error while promoting sparsity:

$$J(W,b) = \left[\frac{1}{m} \sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)}) \right] + \frac{\lambda}{2} \sum_{\ell=1}^{n_\ell - 1} \sum_{j=1}^{s_\ell} \sum_{i=1}^{s_{\ell+1}} \left(W_{ij}^{(\ell)} \right)^2$$

where $J(W,b;x^{(i)},y^{(i)})$ is the prediction error of sample i

$$J(W,b;x^{(i)},y^{(i)}) = \frac{1}{2} \left\| h_{W,b}(x^{(i)}) - y^{(i)} \right\|^2$$

characteristics:

- nonconvex – gradient descent used in practice
- initialization: small random values near 0 (but not all zeros)

Neural Networks – Calculating Gradients

need to compute gradients:

$$\frac{\partial J(W,b)}{\partial W_{ij}^{(\ell)}} = \left[\frac{1}{m}\sum_{i=1}^{m}\frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial W_{ij}^{(\ell)}}\right] + \lambda W_{ij}^{(\ell)}$$

$$\frac{\partial J(W,b)}{\partial b_{i}^{(\ell)}} = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial b_{i}^{(\ell)}}$$

backpropagation to compute $\frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial W_{ij}^{(\ell)}}$ and $\frac{\partial J(W,b;x^{(i)},y^{(i)})}{\partial b_{i}^{(\ell)}}$

Neural Networks – Backpropagation

for the output layer: (the superscript of sample index removed)

$$\begin{aligned}
&\frac{\partial J(W,b;x,y)}{\partial W_{ij}^{(n_\ell-1)}} \\
=\;& \frac{\partial}{\partial W_{ij}^{(n_\ell-1)}} \left\{ \frac{1}{2} \left[y_i - \underbrace{h_{W,b}\left(\sum_{j=1}^{n} W_{ij}^{(n_\ell-1)} a_j^{(n_\ell-1)} + b_i^{(n_\ell-1)} \right)}_{=f(z_i^{(n_\ell)})=a_i^{(n_\ell)}} \right]^2 \right\} \\
=\;& \left(y_i - a_i^{(n_\ell)} \right) \cdot \left[-f'\left(z_i^{(n_\ell)} \right) \right] \cdot \frac{\partial z_i^{(n_\ell)}}{\partial W_{ij}^{(n_\ell-1)}} \\
=\;& -\underbrace{\left(y_i - a_i^{(n_\ell)} \right) \cdot f'\left(z_i^{(n_\ell)} \right)}_{\triangleq \delta_i^{(n_\ell)}} \cdot a_j^{(n_\ell-1)}
\end{aligned}$$

Neural Networks – Backpropagation

for the middle layer $n_\ell - 1$: (the superscript of sample index removed)

$$
\begin{aligned}
&\frac{\partial J(W,b;x,y)}{\partial W_{ij}^{(n_\ell-2)}} \\
= \ & \frac{\partial}{\partial W_{ij}^{(n_\ell-2)}} \left\{ \frac{1}{2} \sum_{k=1}^{s_{n_\ell}} \left[y_k - f(z_k^{(n_\ell)}) \right]^2 \right\} \\
= \ & \sum_{k=1}^{s_{n_\ell}} \left(y_k - a_k^{(n_\ell)} \right) \cdot \left[-f'\left(z_k^{(n_\ell)} \right) \right] \cdot \frac{\partial z_k^{(n_\ell)}}{\partial a_i^{(n_\ell-1)}} \cdot \frac{\partial a_i^{(n_\ell-1)}}{\partial z_i^{(n_\ell-1)}} \cdot \frac{\partial z_i^{(n_\ell-1)}}{\partial W_{ij}^{(n_\ell-2)}} \\
= \ & \underbrace{\sum_{k=1}^{s_{n_\ell}} \delta_k^{(n_\ell)} \cdot W_{ki}^{(n_\ell-1)} \cdot f'\left(z_i^{(n_\ell-1)} \right)}_{\triangleq \delta_i^{(n_\ell-1)}} \cdot a_j^{(n_\ell-2)}
\end{aligned}
$$

Neural Networks – Backpropagation

backpropagation:

- a forward propagation to determine all the $a_i^{(\ell)}, z_i^{(\ell)}$
- for the output layer, set

$$\delta_i^{(n_\ell)} = -(y_i - a_i^{(n_\ell)}) \cdot f'(z_i^{(n_\ell)})$$

- for middle layers $\ell = n_\ell - 1, \ldots, 2$ and each node i in layer ℓ, set

$$\delta_i^{(\ell)} = \left(\sum_{j=1}^{s_{\ell+1}} W_{ji}^{(\ell)} \delta_j^{(\ell+1)} \right) f'(z_i^{(\ell)})$$

- compute gradients

$$\frac{\partial J(W, b; x, y)}{\partial W_{ij}^{(\ell)}} = a_j^{(\ell)} \delta_i^{(\ell+1)}$$

$$\frac{\partial J(W, b; x, y)}{\partial b_i^{(\ell)}} = \delta_i^{(\ell+1)}$$