Neural Networks

CMSC 422 MARINE CARPUAT marine@cs.umd.edu

Neural Networks

- Last time
 - What are Neural Networks?
 - How to make a prediction given an input?
 - Why are neural networks powerful?

Neural Networks

- Last time
 - What are Neural Networks?
 - Multilayer perceptron
 - How to make a prediction given an input?
 - Simple matrix operations + non-linearities
 - Why are neural networks powerful?
 - Universal function approximators!
- Today

– how to train neural networks?

Forward Propagation: given input x, compute network output

Algorithm 24 TwoLayerNetworkPredict(\mathbf{W}, v, \hat{x})

1:for i = 1 to number of hidden units do2: $h_i \leftarrow tanh(w_i \cdot \hat{x})$ 3:end for4:return $v \cdot h$ // compute output unit

Neural Network Training

Backpropagation algorithm = Gradient descent + Chain rule

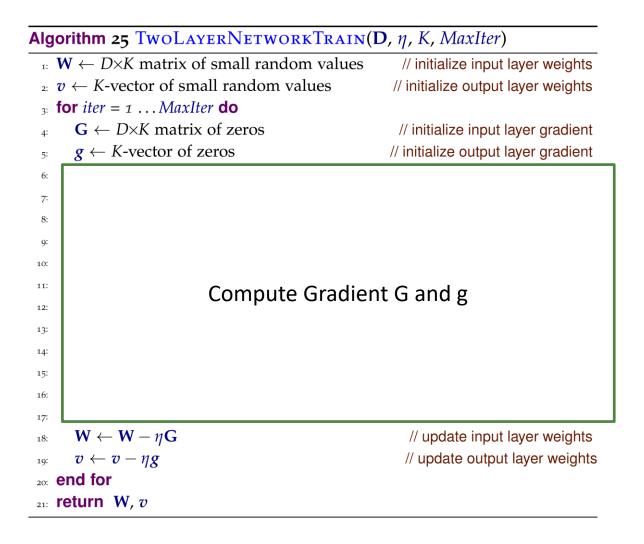
What's our Training Objective?

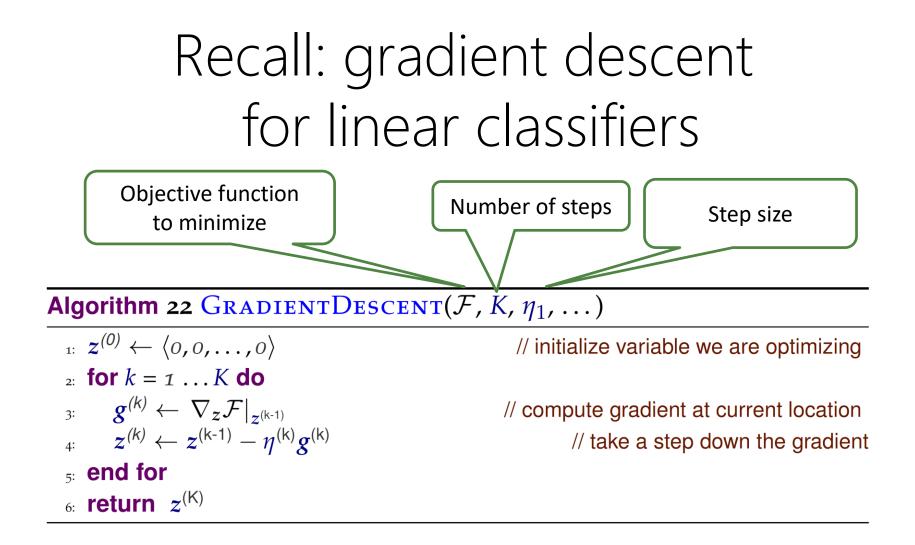
• We'll consider the following objective

$$\min_{\mathbf{W},\mathbf{v}} \quad \sum_{n} \frac{1}{2} \left(y_n - \sum_{i} v_i f(\boldsymbol{w}_i \cdot \boldsymbol{x}_n) \right)^2$$

- i.e. our goal is to find parameters W, v that minimize squared error
- Other objectives are possible (e.g., other loss functions, add regularizer)

Backprop in a 2-layer network





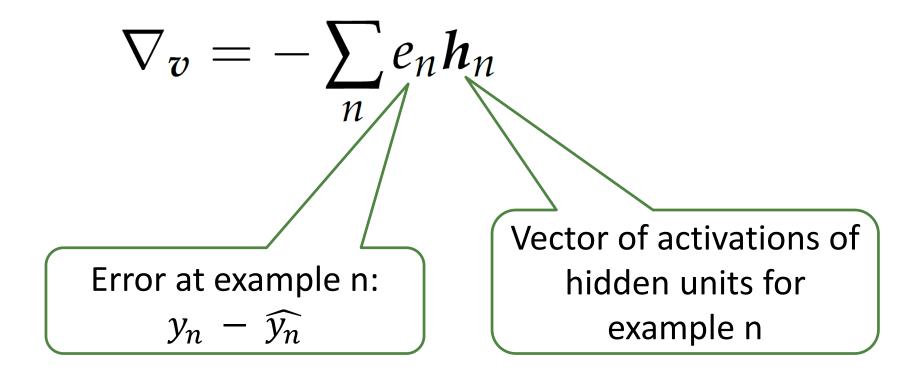
What's our Training Objective?

• We'll consider the following objective

$$\min_{\mathbf{W},\mathbf{v}} \quad \sum_{n} \frac{1}{2} \left(y_n - \sum_{i} v_i f(\boldsymbol{w}_i \cdot \boldsymbol{x}_n) \right)^2$$

- i.e. our goal is to find parameters W, v that minimize squared error
- Other objectives are possible (e.g., other loss functions, add regularizer)

Gradient of objective w.r.t. output layer weights v



Gradient of objective w.r.t. hidden unit weights w; $\mathcal{L}(\mathbf{W}) = \frac{1}{2} \left(y - \sum_{i} v_{i} f(\boldsymbol{w}_{i} \cdot \boldsymbol{x}) \right)^{2}$ Chain rule $\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial f_i} \frac{\partial f_i}{\partial w_i} \frac{\partial \mathcal{L}}{\partial f_i} = -\left(y - \sum_i v_i f(\boldsymbol{w}_i \cdot \boldsymbol{x})\right) v_i = -ev_i$ $\frac{\partial f_i}{\partial w_i} = f'(w_i \cdot x)x$

 $\nabla_{w_i} = -ev_i f'(w_i \cdot x) x$

(This is on one example only)

Backprop in a 2-layer network

Algorithm 25 TwoLayerNetworkTra	IN (D , η, K, MaxIter)	
$\mathbf{W} \leftarrow D \times K$ matrix of small random value	ies // initialize input layer weights	
$_{2} v \leftarrow K$ -vector of small random values	// initialize output layer weights	
$_{3:}$ for <i>iter</i> = 1 <i>MaxIter</i> do		
$_{4:}$ G \leftarrow <i>D</i> × <i>K</i> matrix of zeros	// initialize input layer gradient	
$_{5:}$ $g \leftarrow K$ -vector of zeros	// initialize output layer gradient	
6: for all $(x,y) \in \mathbf{D}$ do		
for $i = 1$ to K do		
8: $a_i \leftarrow w_i \cdot \hat{x}$		Forward
$h_i \leftarrow \tanh(a_i)$	// compute activation of hidden unit i	
10: end for		propagation
$\hat{y} \leftarrow v \cdot h$	// compute output unit	
$e \leftarrow y - \hat{y}$	// compute error	
13: $g \leftarrow g - eh$	// update gradient for output layer	
for $i = 1$ to K do		Update
$\mathbf{G}_i \leftarrow \mathbf{G}_i - ev_i(1 - \tanh^2(a_i))\mathbf{x}$	// update gradient for input layer	gradients
16: end for		gradients
17: end for		
18: $\mathbf{W} \leftarrow \mathbf{W} - \eta \mathbf{G}$	// update input layer weights	Update
19: $v \leftarrow v - \eta g$	// update output layer weights	parameters
20: end for		parameters
20: end tor		

Tricky issues with neural network training

- Sensitive to initialization
 - Objective is non-convex, many local optima
 - In practice: start with random values rather than zeros
- Many other hyperparameters
 - Number of hidden units (and potentially hidden layers)
 - Gradient descent learning rate
 - Stopping criterion

Neural networks vs. linear classifiers

Advantages of Neural Networks:

- More expressive
- Less feature engineering

Inconvenients of Neural Networks:

- Harder to train
- Harder to interpret

Neural Network Architectures

- We focused on a 2-layer feedforward network
- Other architectures are possible
 - More than 2 layers (aka deep learning)
 - Recurrent network (i.e. network has cycles)
 - Can still be trained with backpropagation
 - But more issues arise when networks get more complex (e.g., vanishing gradients)

Try different architectures and training parameters here:

http://playground.tensorflow.org

What you should know

- What are Neural Networks?
 - Multilayer perceptron
- How to make a prediction given an input?
 - Forward propagation: Simple matrix operations + non-linearities
- Why are neural networks powerful?
 - Universal function approximators!
- How to train neural networks?
 - The backpropagation algorithm