Learning neural networks with hidden layers
Optimizing a single-layer neuron

We train to minimize sum of squared errors:

$$\ell(w) = \frac{1}{2} \sum_i [y_i - g(w_0 + \sum_j w_j x_i[j])]^2$$

Taking gradients:

$$\frac{\partial}{\partial x} f(g(x)) = f'(g(x))g'(x)$$

$$\frac{\partial \ell}{\partial w_j} = - \sum_i [y_i - g(w_0 + \sum_j w_j x_i[j])] x_i[j] g'(w_0 + \sum_j w_j x_i[j])$$

Solution just depends on $g'$: derivative of activation function!
Forward propagation

1-hidden layer:

\[
out(x) = g(w_0 + \sum_k w_k g(w_0^k + \sum_j w_j^k x[j]))
\]

For fixed weights, forming predictions is easy!

Compute values left to right

1. Inputs: \(x[1], \ldots, x[d]\)
2. Hidden: \(v[1], \ldots, v[d]\)
3. Output: \(y\)
Multilayer neural networks

**Inference and Learning**

- **Forward pass:**
  
  left to right, each hidden layer in turn

- **Gradient computation:**
  
  right to left, propagating gradient for each node
Back-propagation – Learning

• Just gradient descent!!!
• Recursive algorithm for computing gradient
• For each example
  – Perform forward propagation
  – Start from output layer
    • Compute gradient of node $v[k]$ with parents $u[1], u[2], \ldots$:
    • Update weight $w_{j}^{k}$
    • Repeat (move to preceding layer)
Convergence of backprop

Multilayer neural nets not convex
- Gradient descent gets stuck in local minima
Deep features:

Deep learning
+
Transfer learning
Transfer learning: *Use data from one task to help learn on another*

Old idea, explored for deep learning by Donahue et al. ’14 & others

Lots of data:

- Learn neural net
- Great accuracy on cat v. dog

Some data:

- Neural net as feature extractor + Simple classifier
- Great accuracy on 101 categories
What’s learned in a neural net

Neural net trained for Task 1: cat vs. dog

More generic
Can be used as feature extractor

Very specific
to Task 1
Should be ignored
for other tasks
Transfer learning in more detail...

For Task 2, predicting 101 categories, learn only end part of neural net

Neural net trained for Task 1: cat vs. dog

Keep weights fixed!

More generic
Can be used as feature extractor

Use simple classifier
e.g., logistic regression, SVMs, nearest neighbor,...

Very specific
to Task 1
Should be ignored
for other tasks

Class?