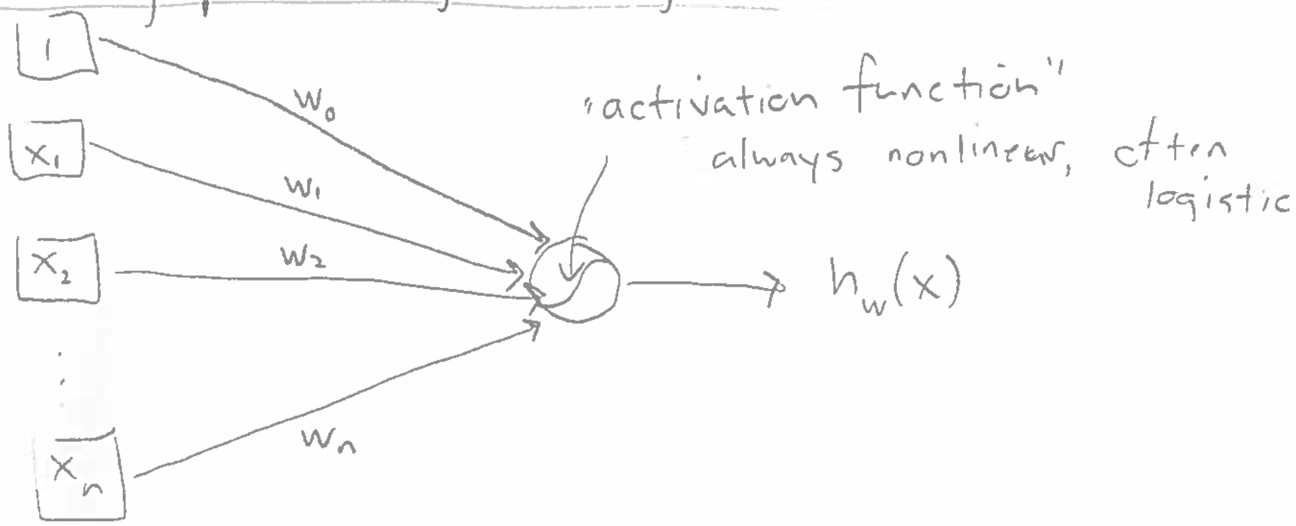
Non-linear supervised learning

- feature expansion + linear model
- kernel methods
- "instance-based" methods (k-NN)
- decision trees
- neural networks: parametric non-linear functions

controlled by small set of parameters

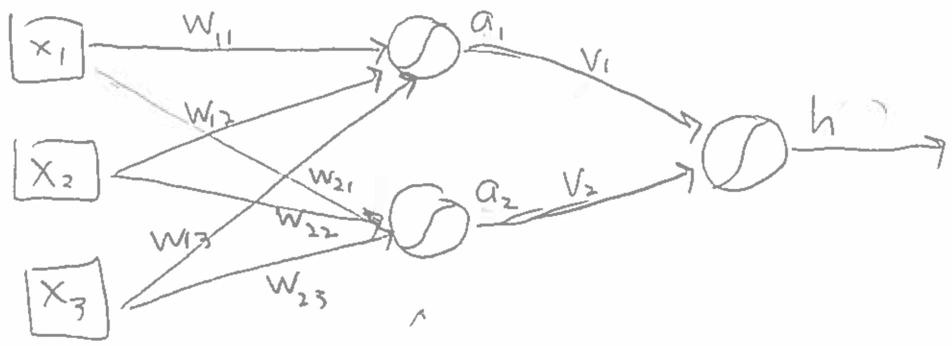
Starting point: logistic regression



Inputs   Weights   neuron/unit   output

$$h_w(x) = g(w_0 + w_1 x + \cdots + w_n x_n)$$
$$= g(w^T x)$$

Multi-layer
<u>Neural network</u>: multiple "hidden" logistic regression models used as inputs to a linear model



Layer 1 Layer 2 Layer 3
"hidden"

$$a_k = g(w_{10} + w_{11}x_1 + \cdots + w_{1n}x_n) \quad k=1,\dots,K$$

$$h = \begin{cases} v_0 + v_1 a_1 + \cdots + v_k a_k & \text{regression} \\ g(v_0 + v_1 a_1 + \cdots + v_k a_k) & \text{classification} \end{cases}$$

same principles apply to any # of layers or hidden units

Exercise: $x \in \mathbb{R}$, $K=2$. What class of functions can be learned?

Learning: "stochastic gradient descent" / "back prop"

Assume cost is additive over examples:

$$J(\theta) = \sum_{i=1}^{m} J^{(i)}(\theta)$$

cost on example $(x^{(i)}, y^{(i)})$

e.g. $J^{(i)}(\theta) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$

Standard ("batch") gradient descent

for $j = 1$ to n //parameters

$$\theta_j \leftarrow \theta_j - \alpha \sum_{i=1}^{m} \underbrace{(h(x^{(i)}) - y^{(i)})\, x^{(i)}}_{\frac{\partial J^{(i)}}{\partial \theta_j}}$$

end

sum over all examples before each update to θ_j

Stochastic gradient descent (SGD)

for $i = 1$ to m //training examples

for $j = 1$ to n //parameters

$$\theta_j \leftarrow \theta_j - \alpha \underbrace{(h(x^{(i)}) - y^{(i)}) \cdot x^{(i)}}_{\frac{\partial J^{(i)}}{\partial \theta_j}}$$

end

end

loop through examples, update θ after each one

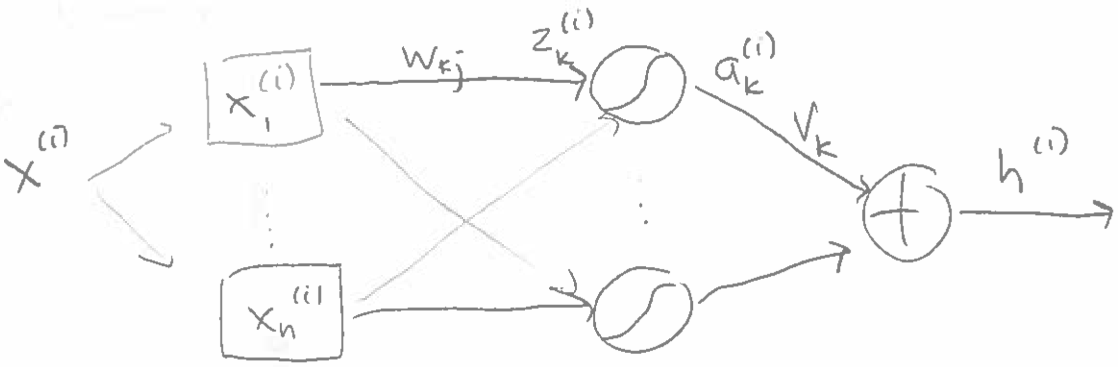
Discuss SGD

- simple
- memory efficient (huge data sets)
- online (handle new training data)
- ...

Back propagation: SGD in a neural net

chain rule $\Rightarrow$ propagation of updates backward through net

Setup (regression, no bias)



$$z_k^{(i)} = \sum_{j=1}^{n} w_{kj}\, x_j^{(i)}$$ input to k^{th} hidden unit

$$a_k^{(i)} = g(z_k^{(i)})$$ output from k^{th} hidden

$$h^{(i)} = \sum_{k=1}^{K} v_k a_k^{(i)}$$ overall output

$$J^{(i)} = \tfrac{1}{2}(h^{(i)} - y^{(i)})^2$$ cost on example i

Parameters : w_{kj} , $j=1,\dots n$, $k=1,\dots,K$

v_k , $k=1,\dots,K$

Partial derivatives:

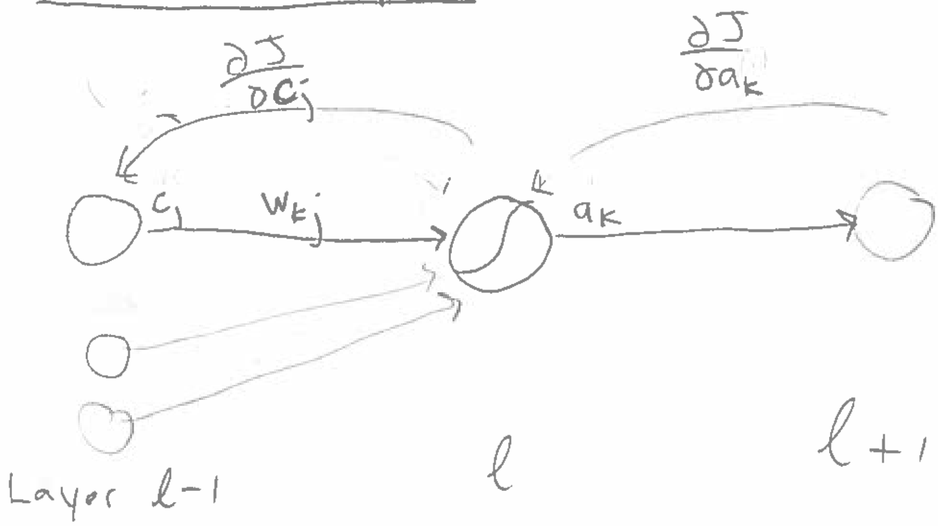
$$\frac{\partial J^{(i)}}{\partial v_k} = \frac{\partial J^{(i)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial v_k}$$

$$= \left(h^{(i)} - y^{(i)}\right) \cdot a_k^{(i)}$$

$$\frac{\partial J^{(i)}}{\partial w_{kj}} = \frac{\partial J^{(i)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial a_k^{(i)}} \cdot \frac{\partial a_k^{(i)}}{\partial z_k^{(i)}} \cdot \frac{\partial z_k^{(i)}}{\partial w_{kj}}$$

$$= \underbrace{\left(h^{(i)} - y^{(i)}\right) \cdot v_k}_{\substack{\text{information} \\ \text{from next layer}}} \cdot a_k^{(i)} \cdot \left(1 - a_k^{(i)}\right) \cdot x_j^{(i)}$$

General backprop



$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial a_k} \cdot a_k \cdot (1-a_k) \cdot c_j$$ // use to update w_{kj}

$$\frac{\partial J}{\partial c_j} = \frac{\partial J}{\partial a_k} \cdot a_k(1-a_k) \cdot w_{kj}$$ // pass to previous layer

---

Architectures