Machine Learning Review
Neural Network Training

COSC 7336: Advanced Natural Language Processing
Fall 2017

Some content in these slides has been adapted from Jurafsky & Martin 3rd edition, and lecture slides from Rada Mihalcea, Ray Mooney and the deep learning course by Manning and Socher.

UNIVERSITY of
HOUSTON

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Today’s Lecture

- Machine learning review
  - The learning problem
  - Learning and optimization
  - Generalization
- Neural network training
  - Perceptron training
  - Backpropagation
  - In-class assignment

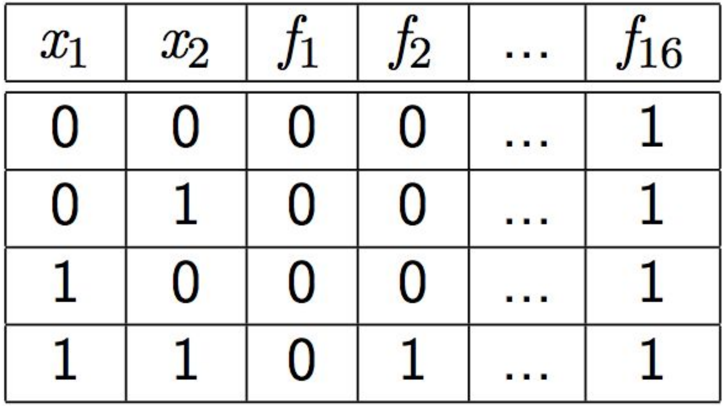
What is machine learning?

- ★ Artificial intelligence
- ★ Pattern recognition
- ★ Computational learning theory
- ★ Data mining, predictive analytics, data science
- ★ Statistics, statistical learning

What is a pattern?

- ★ Data regularities
- ★ Data relationships
- ★ Redundancy
- ★ Generative model

Learning a boolean function



x_1	x_2	f_1	f_2	...	f_{16}
0	0	0	0	...	1
0	1	0	0	...	1
1	0	0	0	...	1
1	1	0	1	...	1

How many Boolean functions of n variables are?

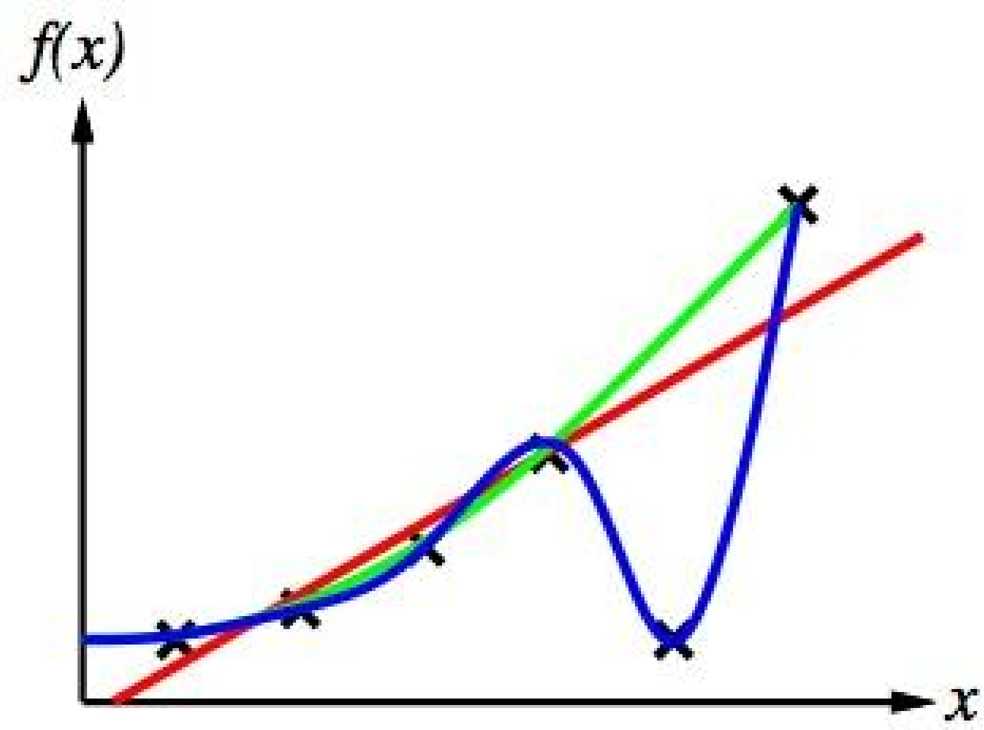
How many candidate functions are removed by a sample?

Is it possible to generalize from examples?

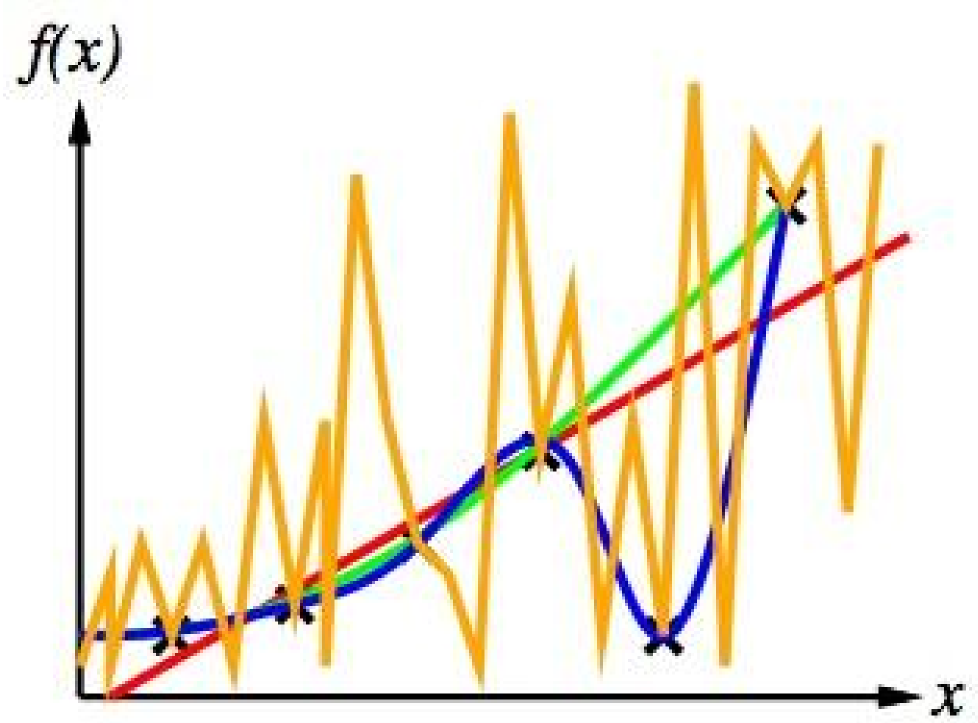
Inductive bias

- In general, the learning problem is ill-posed (more than one possible solution for the same particular problem, solutions are sensitive to small changes on the problem)
- It is necessary to make additional assumptions about the kind of pattern we want to learn
- **Hypothesis space**: set of valid patterns that can be learnt by the algorithm

What is a good pattern?



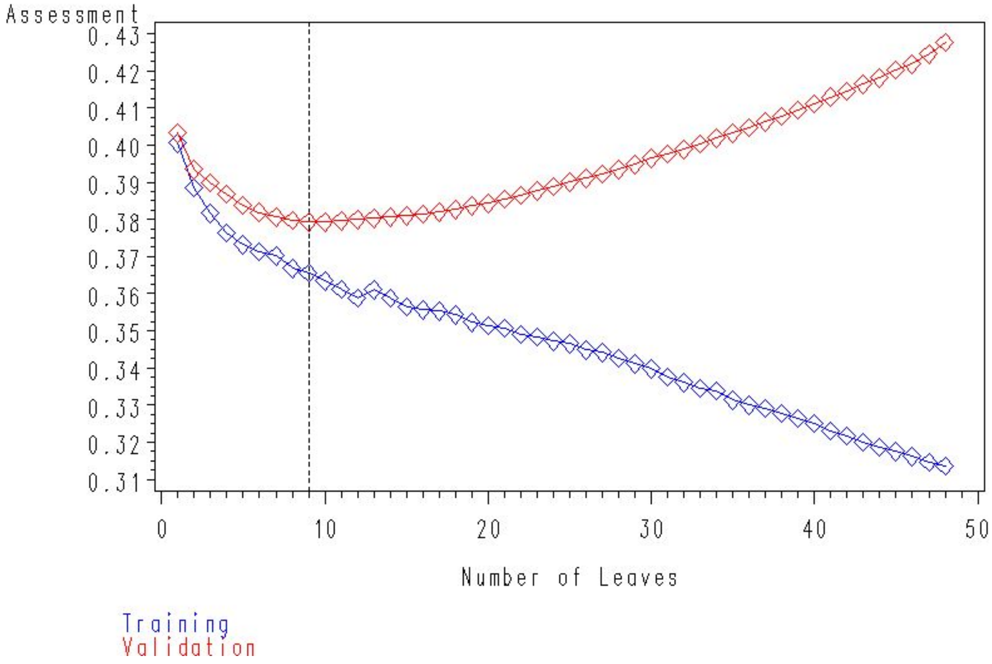
What is a good pattern?



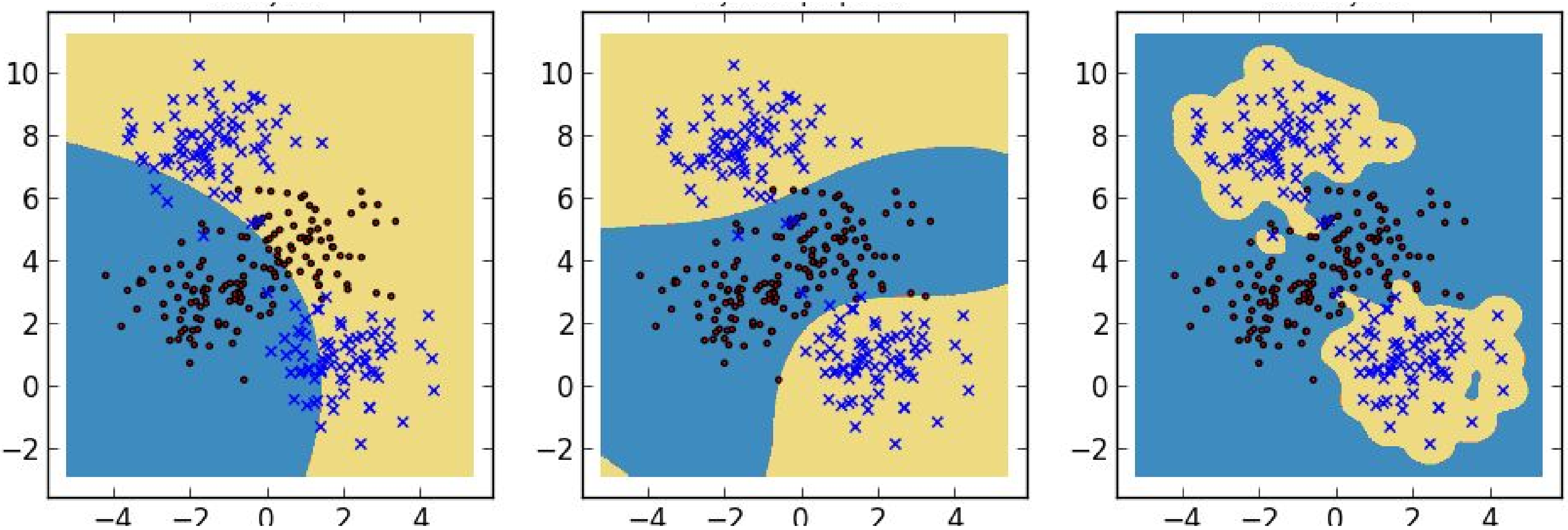
Generalization

- ★ The loss function measures the error in the training set
- ★ Is this a good measure of the quality of the solution?

Average Square Error (Gini index)



Over-fitting and under-fitting



Generalization error

- ★ Generalization error:

$$E[(L(f_w, D)$$

- ★ How to control the generalization error during training?
    - ○ Cross validation
    - ○ Regularization

Regularization

Vapnik (1995):

Expected loss

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x},\alpha)|dP(\mathbf{x},y)$$

Empirical loss

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(\mathbf{x}_i,\alpha)|.$$

Model complexity

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2l/h)+1) - \log(\eta/4)}{l}\right)}$$

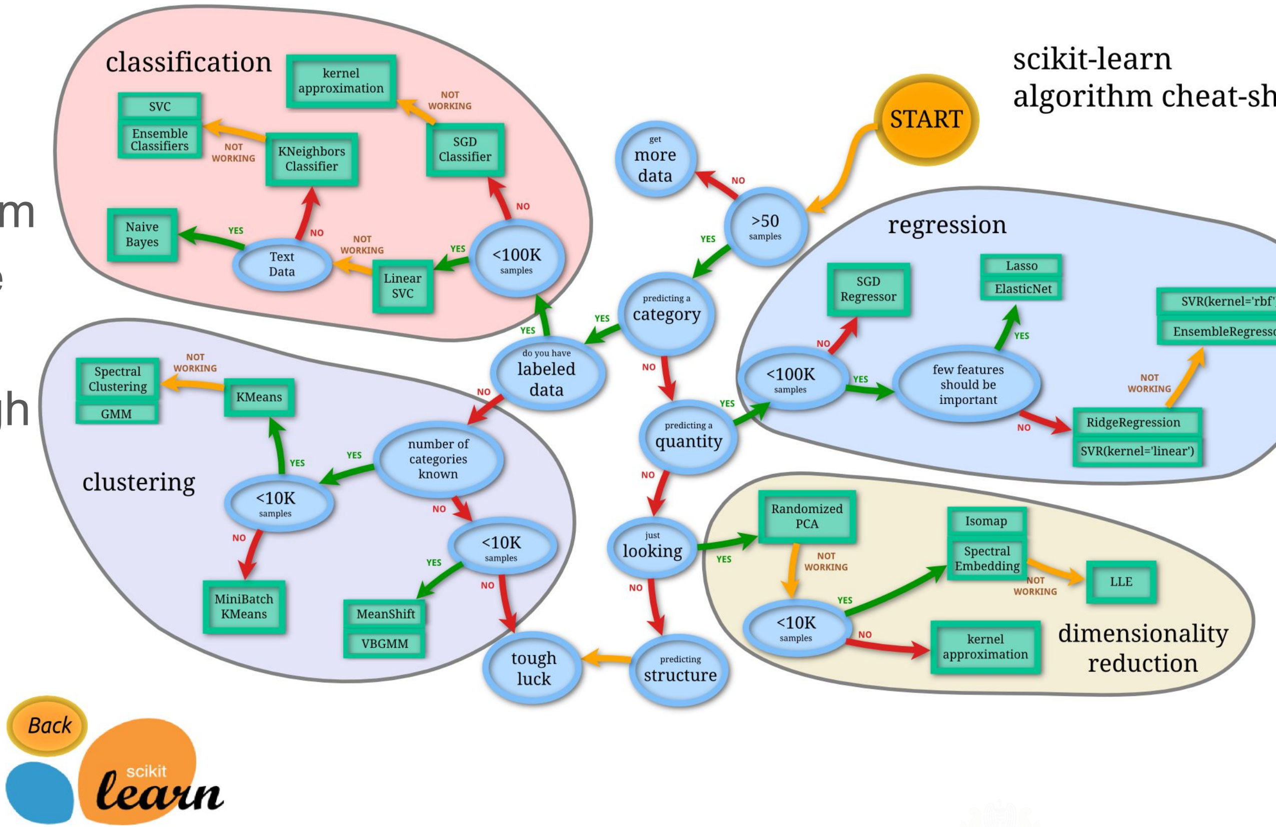
Number of samples

Types of learning problems

- ★ Supervised learning
- ★ Non-supervised learning
- ★ Semi-supervised learning
- ★ Active/reinforcement learning

Methods

- ★ There are many!
- ★ Depend on the problem to solve and underline strategy
- ★ Trends change through time.
- ★ Now:
    - ○ NNs 👍
    - ○ SVMs 👎
- ★ Used to be the other way around



Perceptron Training Demo

Neural Network Training

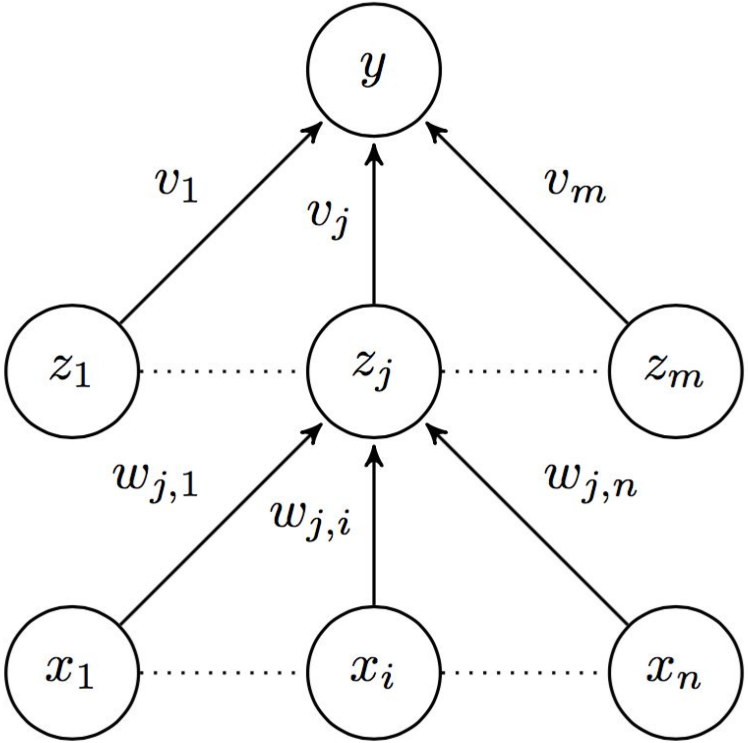
Training multilayer networks

- ★ We will use gradient descent as well.
- ★ We need to calculate

$$\frac{\partial E_\ell}{\partial w_{ji}}$$

- ★ An analytical solution gets very complicated even for a small NN
- ★ Backpropagation is an efficient strategy for gradient calculation

Rumelhart, D.; Hinton, G.; Williams, R. (1986). "Learning representations by back-propagating errors". *Nature*. **323** (6088): 533–536.



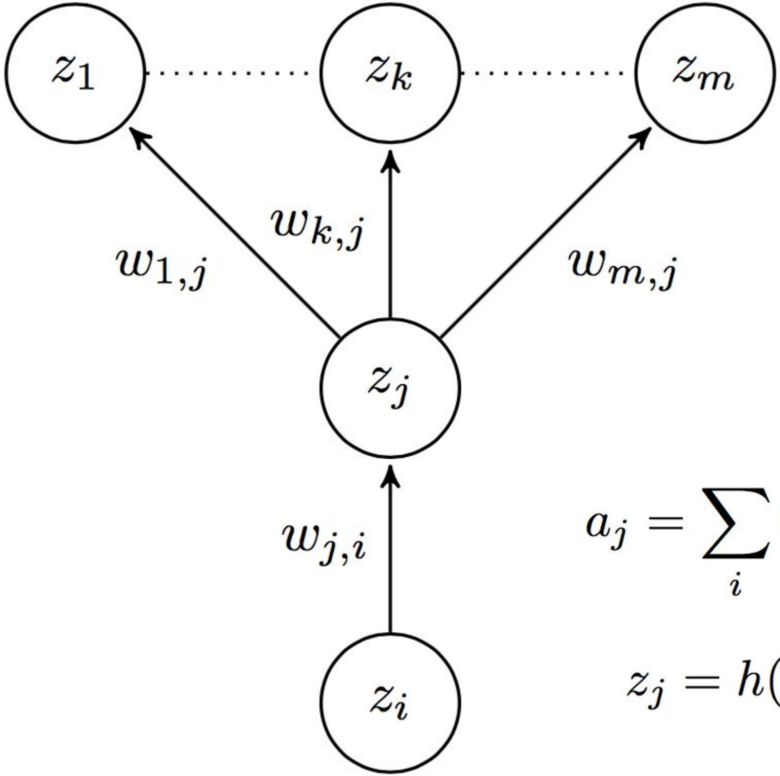
Gradient calculation

- General case for a neuron z_j and z_i in layers n and n-1 respectively
- We want to calculate the gradient:

$$\frac{\partial E_\ell}{\partial w_{ji}}$$

- General strategy: re-express the gradient as a function of two values

$$\frac{\partial E_\ell}{\partial w_{ji}} = \delta_j z_i$$



$$a_j = \sum_i w_{ji} z_i$$

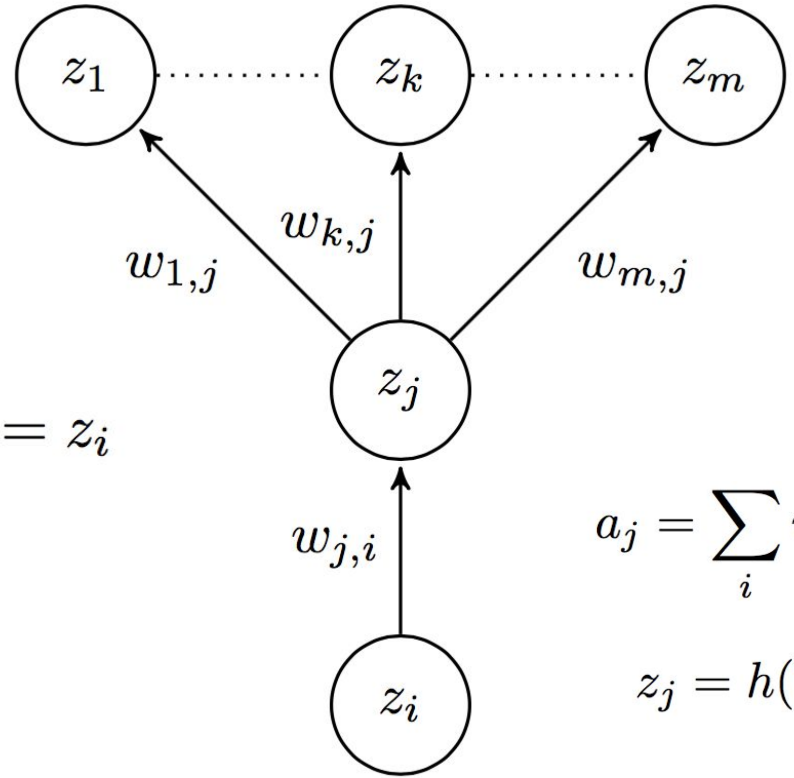
$$z_j = h(a_j)$$

Gradient decomposition

$$\frac{\partial E_\ell}{\partial w_{ji}} = \frac{\partial E_\ell}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}$$ (chain rule)

$$\delta_j = \frac{\partial E_\ell}{\partial a_j} \qquad \frac{\partial a_j}{\partial w_{ji}} = \frac{\partial \sum_i w_{ji} z_i}{\partial w_{ji}} = z_i$$

$$\frac{\partial E_\ell}{\partial w_{ji}} = \delta_j z_i$$



$$a_j = \sum_i w_{ji} z_i$$

$$z_j = h(a_j)$$

Generalized (multidimensional) chain rule (GCR)

$$y = f(u_1, \ldots, u_m)$$

$$\mathbf{u} = g(x_1, \ldots, x_n)$$

$$\frac{\partial y}{\partial x_i} = \sum_{\ell=1}^{m} \frac{\partial y}{\partial u_\ell} \frac{\partial u_l}{\partial x_i}$$

δ_j recursive calculation

$$\delta_j = \frac{\partial E_\ell}{\partial a_j} = \sum_{k=1}^{m} \frac{\partial E_\ell}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

Applying GCR since $E_\ell = f(a_1, \ldots, a_k)$

$$= \sum_{k=1}^{m} \delta_k \frac{\partial a_k}{\partial a_j}$$

Definition of δ_j

$$= \sum_{k=1}^{m} \delta_k \frac{\partial a_k}{\partial z_j} \frac{\partial z_j}{\partial a_j}$$

Chain rule

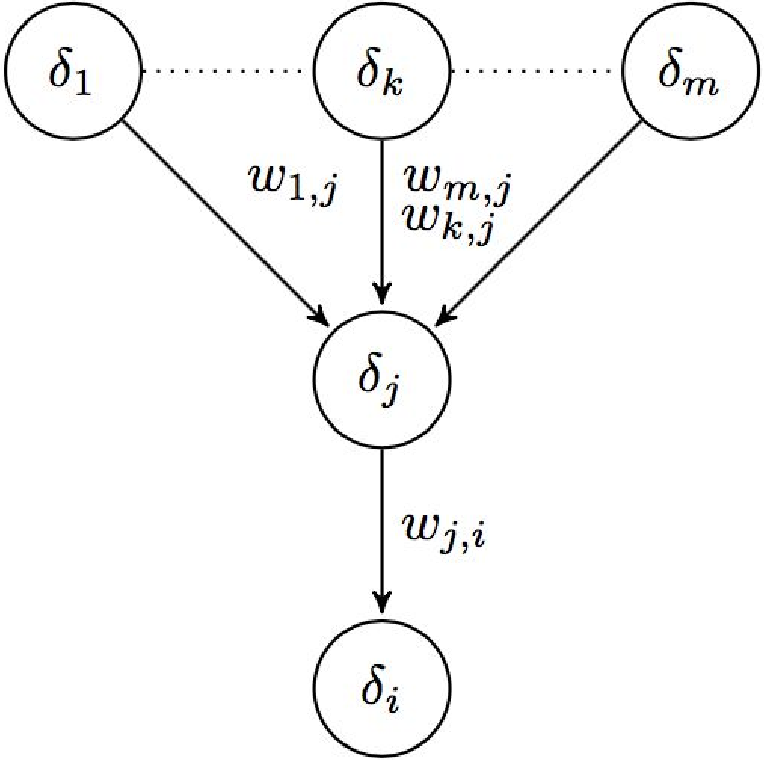
$$= \sum_{k=1}^{m} \delta_k w_{kj} h'(a_j)$$

$$= h'(a_j) \sum_{k=1}^{m} \delta_k w_{kj}$$

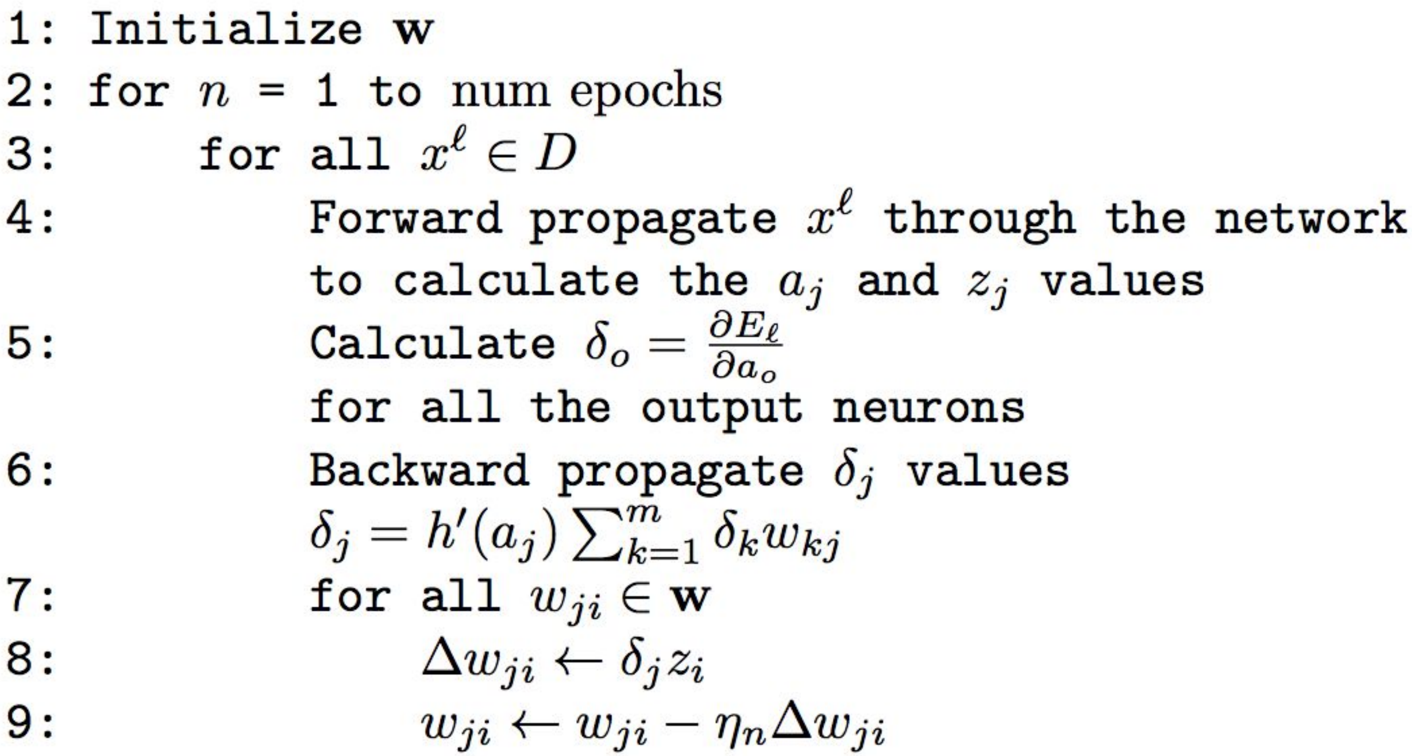
δ_j backpropagation

Deltas in layer *n* are calculated from deltas in layer *n + 1*:

$$\delta_j = h'(a_j) \sum_{k=1}^{m} \delta_k w_{kj}$$



Backpropagation algorithm



```
Initialize $\mathbf{w}$
for $n$ = 1 to num epochs
    for all $x^\ell \in D$
        Forward propagate $x^\ell$ through the network
        to calculate the $a_j$ and $z_j$ values
        Calculate $\delta_o = \frac{\partial E_\ell}{\partial a_o}$
        for all the output neurons
        Backward propagate $\delta_j$ values
        $\delta_j = h'(a_j)\sum_{k=1}^{m} \delta_k w_{kj}$
        for all $w_{ji} \in \mathbf{w}$
            $\Delta w_{ji} \leftarrow \delta_j z_i$
            $w_{ji} \leftarrow w_{ji} - \eta_n \Delta w_{ji}$
```

Two layers NN with sigmoid activation

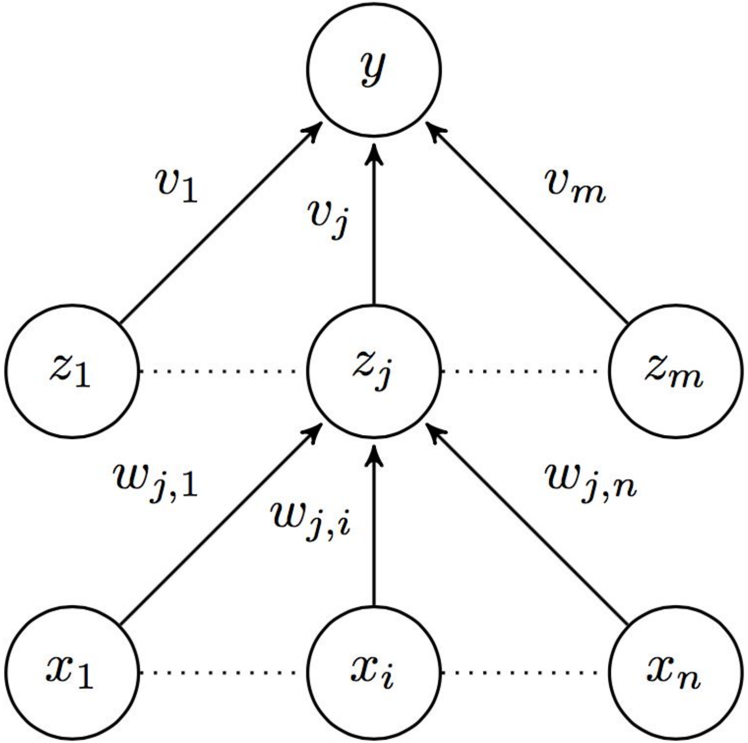
$$a_j = \sum_i w_{ji} x_i$$

$$z_j = \sigma(a_j)$$

$$a_y = \sum_j v_j z_j$$

$$y = \sigma(a_y)$$

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



Two layers NN with sigmoid activation

$$E_\ell(w) = -r^\ell \log y^t - (1 - r^\ell)\log(1 - y^t)$$

$$\delta_y = \frac{\partial E_\ell}{\partial a_y} = \sigma(a_y) - r^\ell = y - r^\ell$$

$$\delta_j = \sigma'(a_j)\delta_y v_j = \sigma(a_j)(1 - \sigma(a_j))\delta_y v_j = z_j(1 - z_j)\delta_y v_j$$