

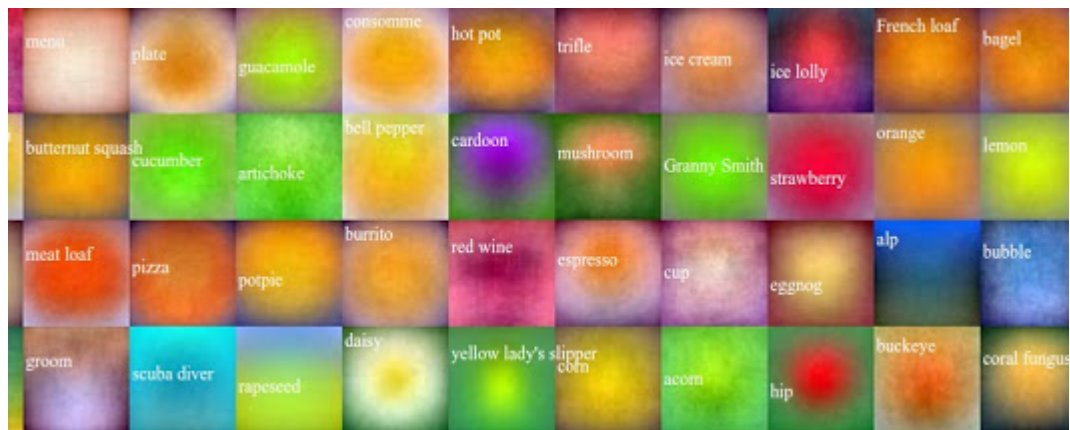
Going Deep into Neural Networks

Beyond linear

- Linear score function $f = Wx$



On CIFAR-10



On ImageNet

Beyond linear

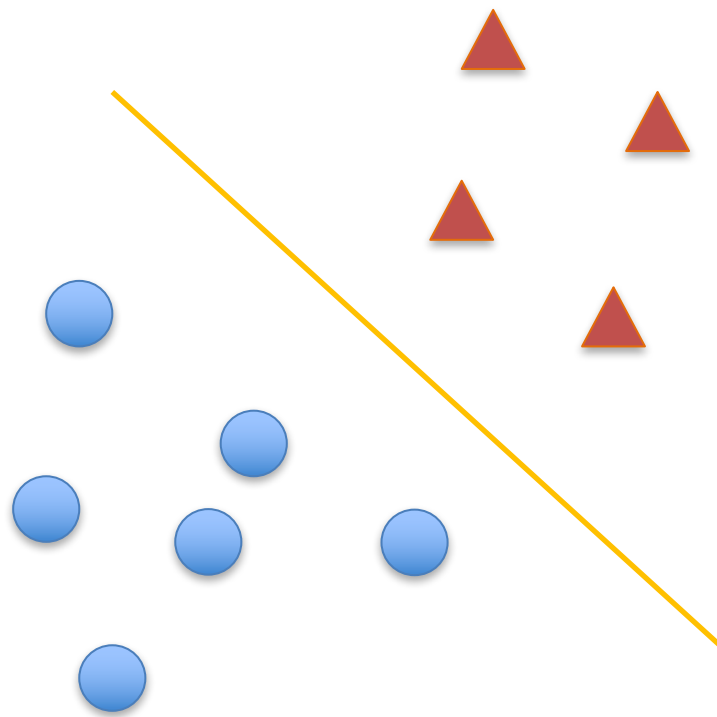
1-layer network: $f = \mathbf{W}\mathbf{x}$



128×128

10

LINEAR
TRANSFORMATION



Beyond linear

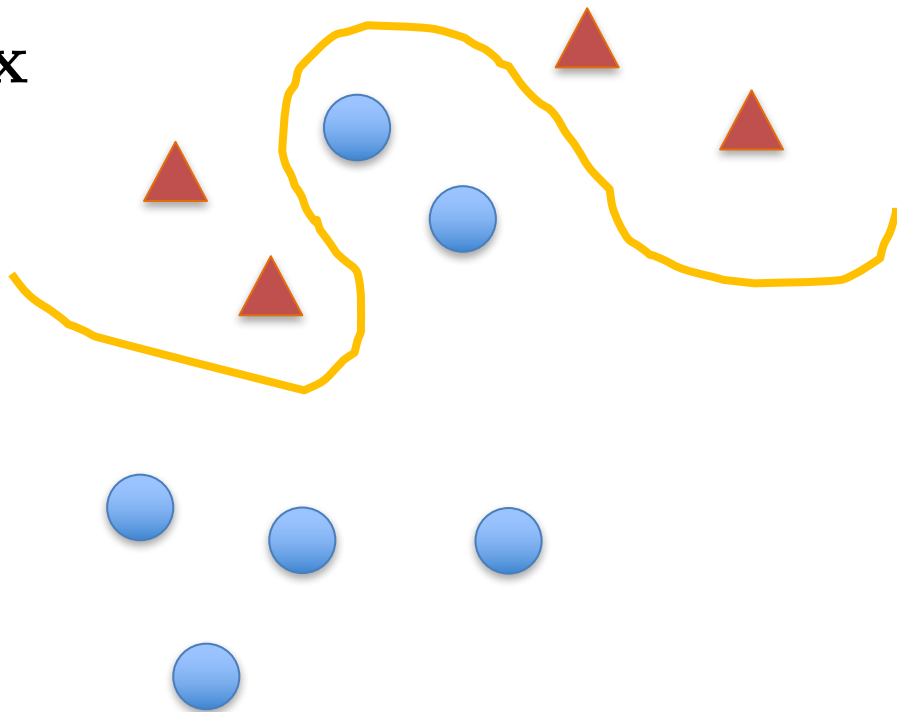
1-layer network: $f = \mathbf{W}\mathbf{x}$



128×128

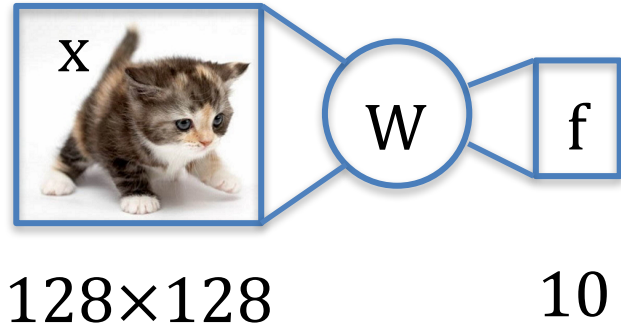
10

LINEAR
TRANSFORMATION



Kernel trick

1-layer network: $f = \mathbf{W}\mathbf{x}$



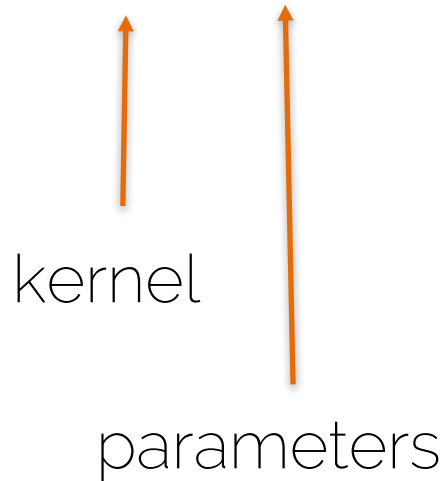
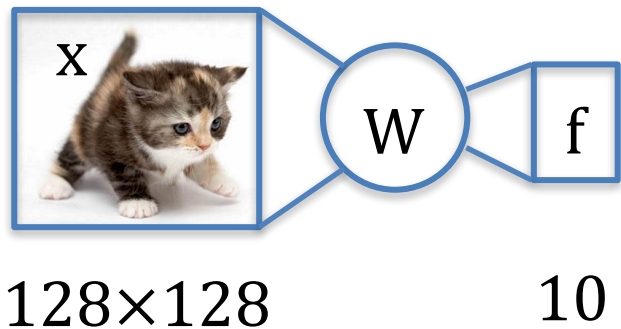
$$f = \mathbf{W}\phi(\mathbf{x})$$



Neural networks

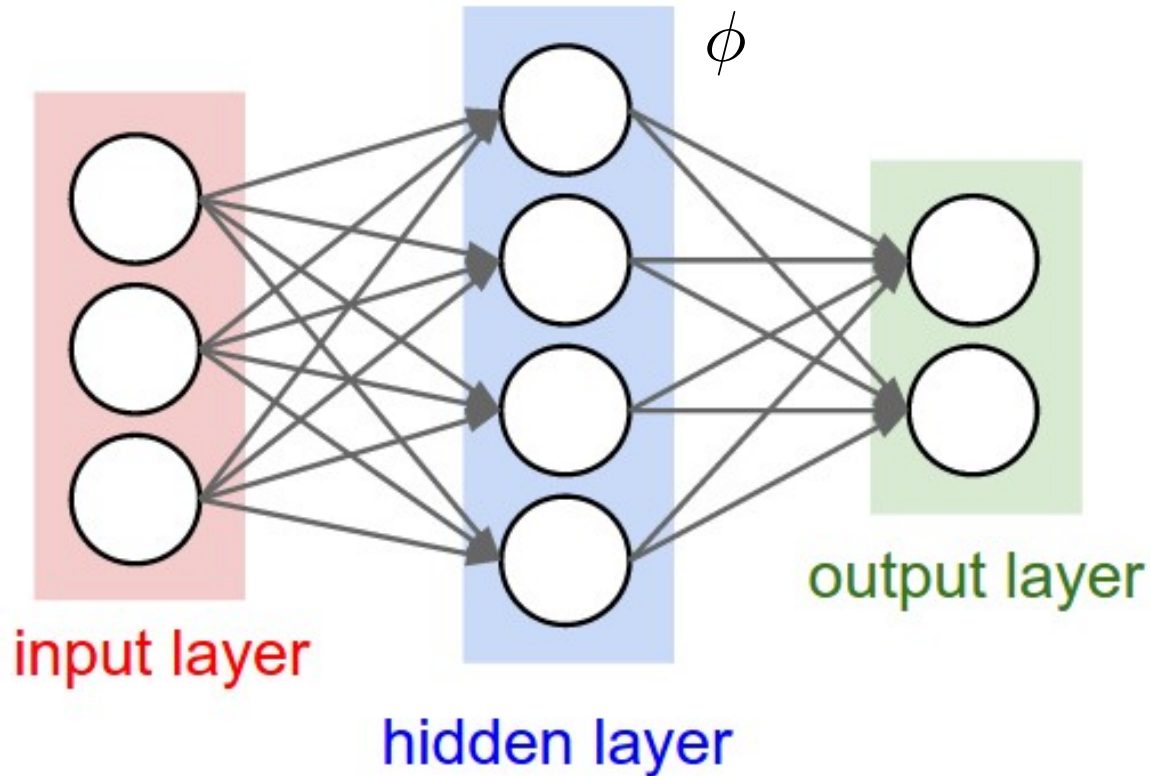
1-layer network: $f = \mathbf{W}\mathbf{x}$

$$f = \mathbf{W}\phi(\mathbf{x}; \boldsymbol{\theta})$$



From the broad family of functions ϕ we learn the best representation by learning the parameters $\boldsymbol{\theta}$

Neural Network



Also SVM
is in this
category

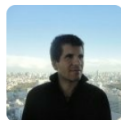
Neural Network

- Problems of going deeper...
- The impact of small decisions (architecture, activation functions...)
- Is my network training correctly?

A typical Deep Learner day

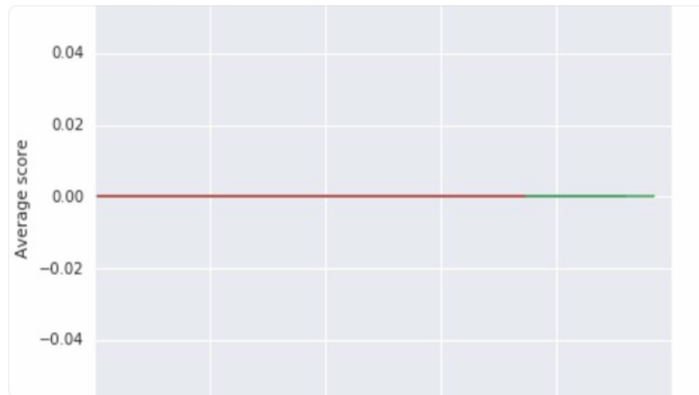


♥ A Andrej Karpathy i Ian Goodfellow els agrada



Oriol Vinyals @OriolVinyalsML · 9h

A typical training curve in Montezuma's Revenge (note: there are several random seeds which overlap) 🤔 #nips #rl #exploration



← 2

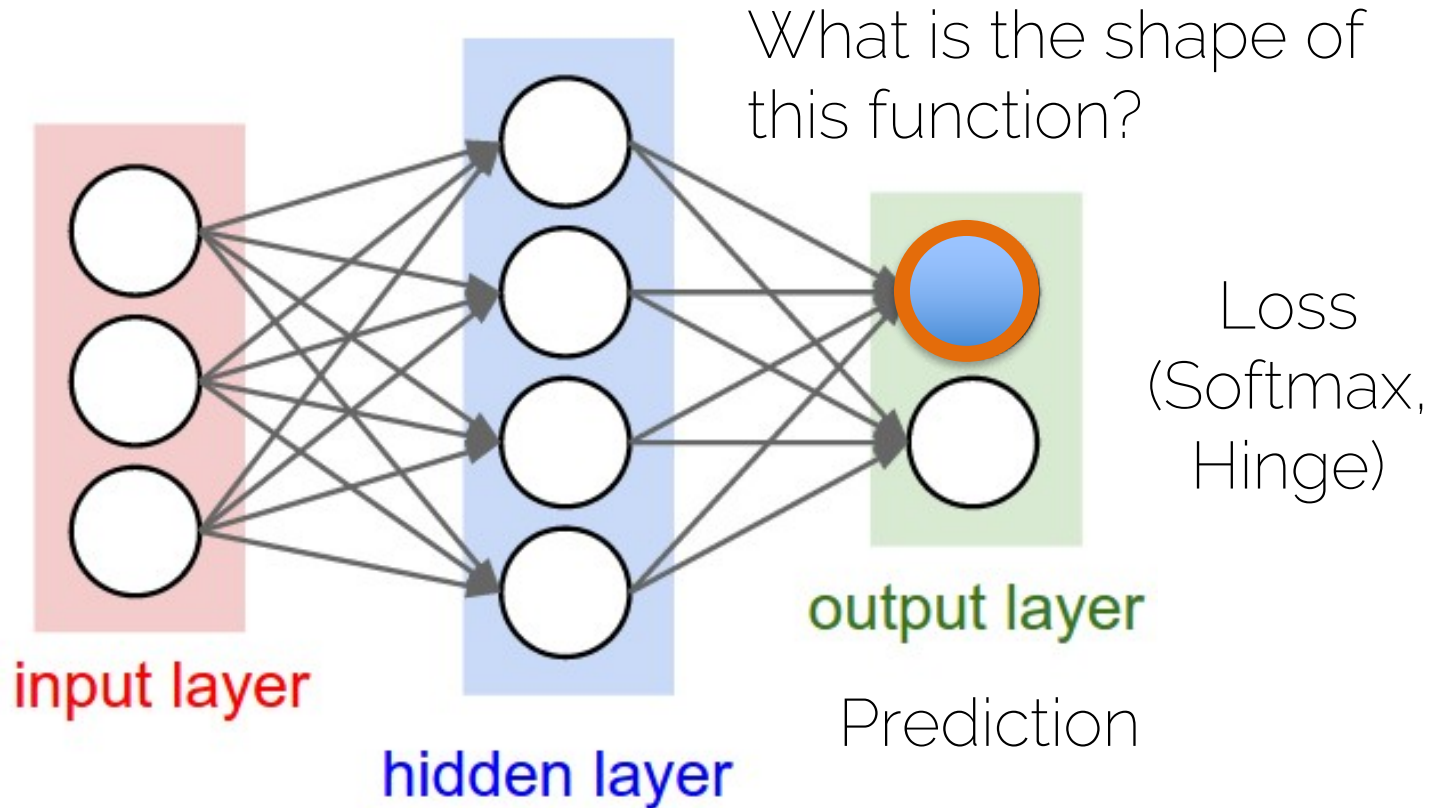
↻ 9

♥ 63

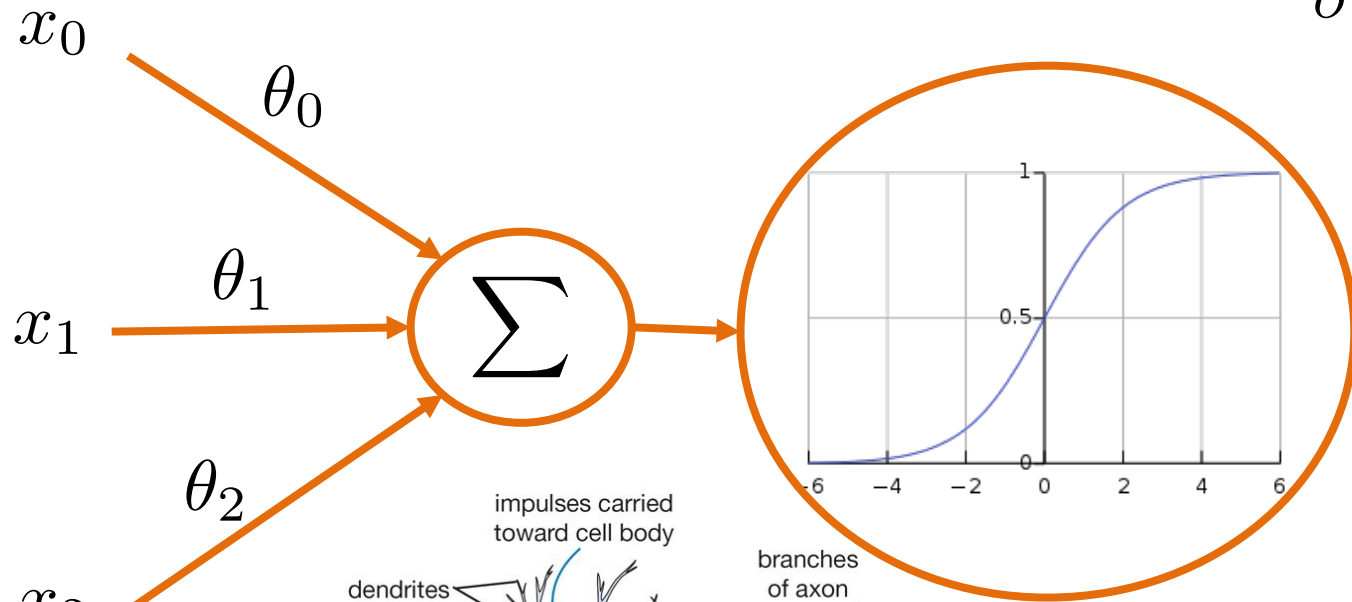


Output functions

Neural networks



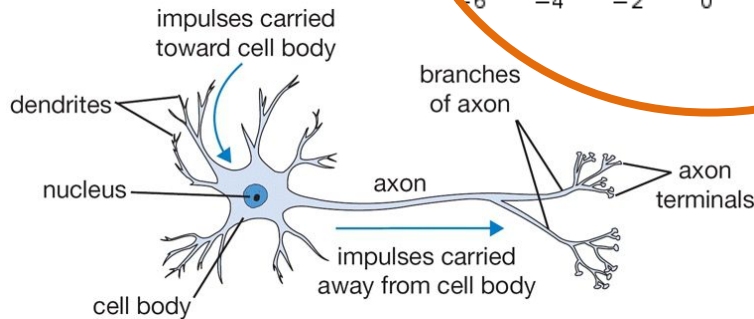
Sigmoid for binary predictions



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

1
↑
↓
0


Can be interpreted as a probability



$$p(y_i = 1 | \mathbf{x}_i, \boldsymbol{\theta})$$

Logistic regression

- Probability of a binary output

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^n \text{Ber}(y_i | \text{sigm}(\mathbf{x}_i, \boldsymbol{\theta}))$$


Model for
coins

$$p(x|\phi) = \phi^x (1 - \phi)^{1-x} = \begin{cases} \phi & \text{if } x = 1 \\ 1 - \phi & \text{if } x = 0 \end{cases}$$

Logistic regression

- Probability of a binary output

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^n \text{Ber}(y_i | \text{sigm}(\mathbf{x}_i, \boldsymbol{\theta}))$$

$$= \prod_{i=1}^n \underbrace{\left[\frac{1}{1 + e^{-\mathbf{x}_i \boldsymbol{\theta}}} \right]}_{\Pi_i}^{y_i} \left[1 - \underbrace{\frac{1}{1 + e^{-\mathbf{x}_i \boldsymbol{\theta}}} \right]}_{\Pi_i} \right]^{1-y_i}$$

$$p(x|\phi) = \phi^x (1 - \phi)^{1-x}$$

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

Logistic regression

- Probability of a binary output

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^n [\Pi_i]^{y_i} [1 - \Pi_i]^{1-y_i}$$

- Maximum Likelihood

$$\boldsymbol{\theta}_{ML} = \arg \max_{\boldsymbol{\theta}} \log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

Logistic regression

- Probability of a binary output $\Pi_i = \frac{1}{1 + e^{-\mathbf{x}_i \boldsymbol{\theta}}}$

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^n [\Pi_i]^{y_i} [1 - \Pi_i]^{1-y_i}$$

$$\begin{aligned} C(\boldsymbol{\theta}) &= -\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) \\ &= -\sum_{i=1}^n y_i \log(\Pi_i) + (1 - y_i) \log(1 - \Pi_i) \end{aligned}$$

Referred to as cross-entropy

Logistic regression

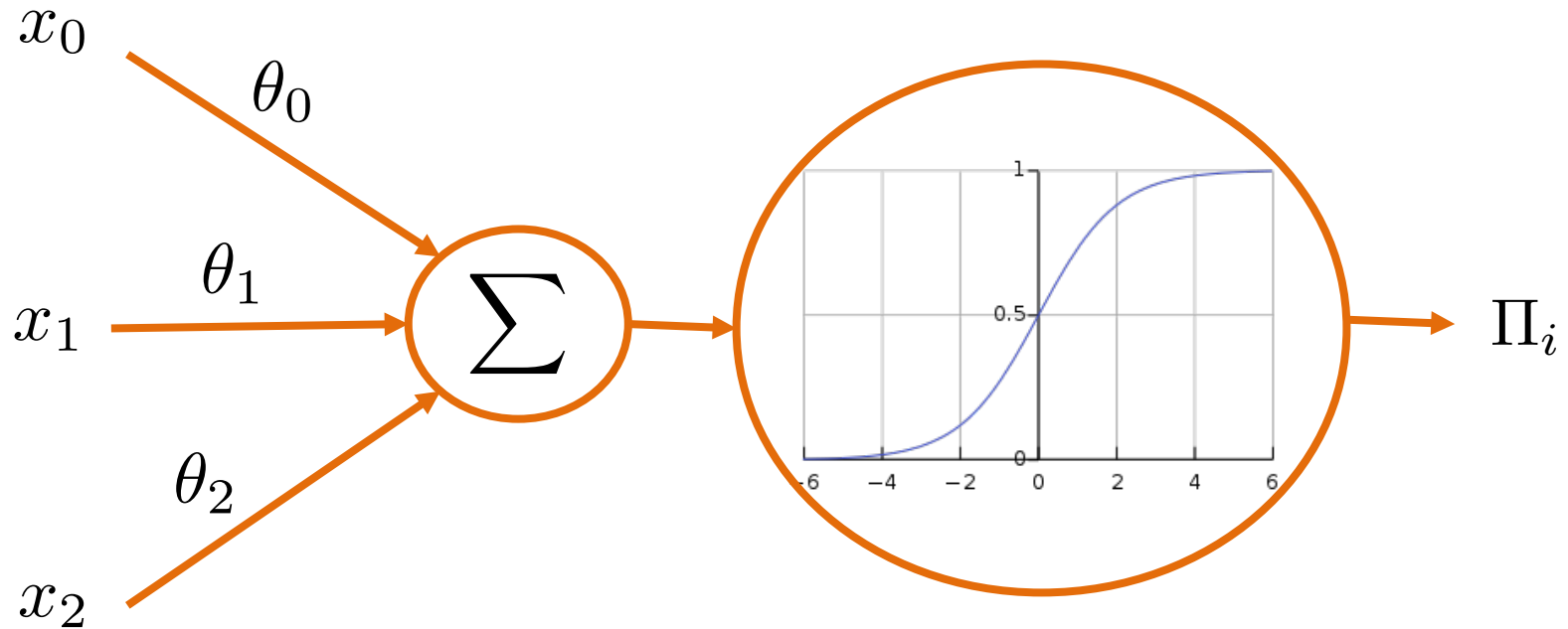
- Optimize using gradient descent
- Saturation occurs only when the model already has the right answer

$$C(\boldsymbol{\theta}) = - \sum_{i=1}^n y_i \log(\Pi_i) + (1 - y_i) \log(1 - \Pi_i)$$

Referred to as cross-entropy

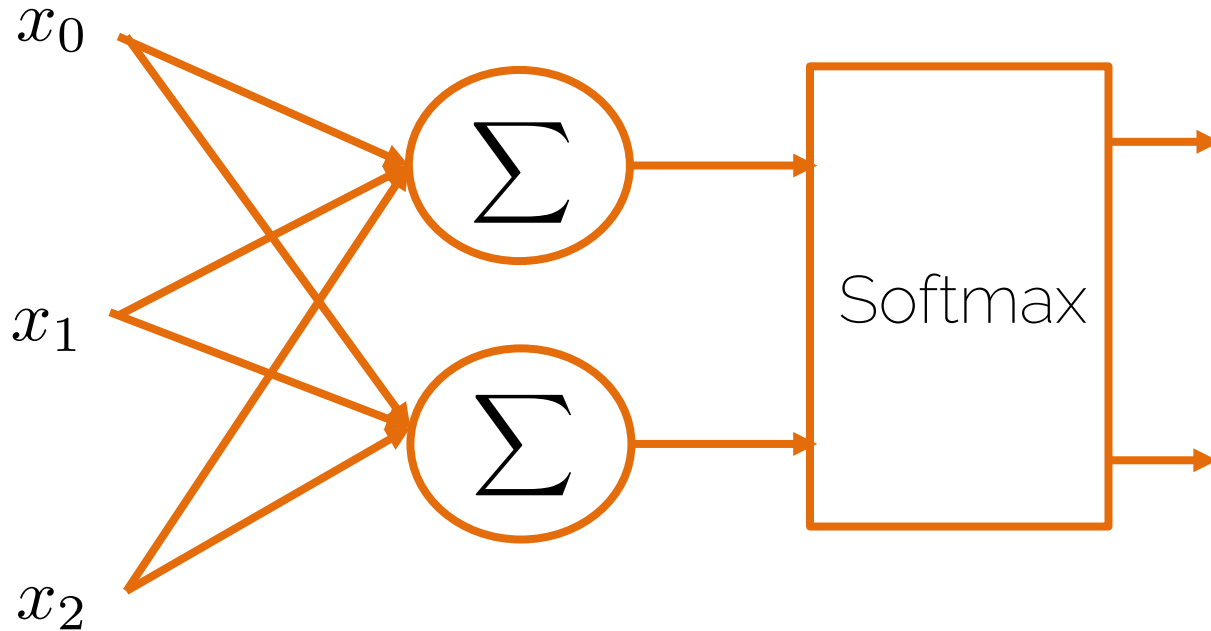
Softmax formulation

- What if we have multiple classes?



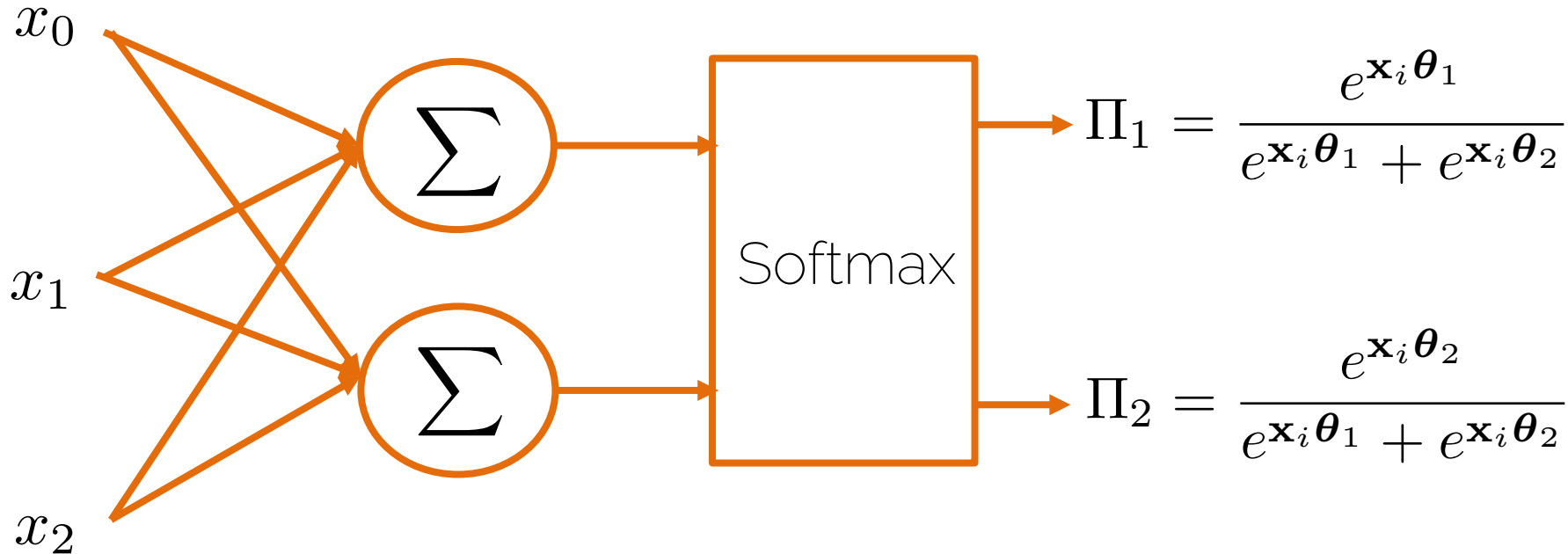
Softmax formulation

- What if we have multiple classes?



Softmax formulation

- What if we have multiple classes?



Softmax formulation

- Softmax

$$p(y_i | \mathbf{x}, \boldsymbol{\theta}) = \frac{e^{\mathbf{x}\boldsymbol{\theta}_i}}{\sum_{k=1}^n e^{\mathbf{x}\boldsymbol{\theta}_k}}$$

exp

normalize

- Softmax loss (ML)

$$L_i = -\log \left(\frac{e^{s_{y_i}}}{\sum_k e^{s_k}} \right)$$

Loss Functions

Naïve Losses

$$L^2 \text{ Loss: } L^2 = \sum_{i=1}^n (y_i - f(x_i))^2$$

- Sum of squared differences (SSD)
- Prone to outliers
- Compute-efficient (optimization)
- Optimum is the mean

$$L^1 \text{ Loss: } L^1 = \sum_{i=1}^n |y_i - f(x_i)|$$

- Sum of absolute differences
- Robust
- Costly to compute
- Optimum is the median

12	24	42	23
34	32	5	2
12	31	12	31
31	64	5	13

x_i

15	20	40	25
34	32	5	2
12	31	12	31
31	64	5	13

y_i

$$L^2(x, y) = 9 + 16 + 4 + 4 + 0 + \dots + 0 = 66$$

$$L^1(x, y) = 3 + 4 + 2 + 2 + 0 + \dots + 0 = 15$$

Cross-Entropy (Softmax)

Softmax $L_i = -\log\left(\frac{e^{s y_i}}{\sum_j e^{s_j}}\right)$

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss

Cross-Entropy (Softmax)

Softmax $L_i = -\log\left(\frac{e^{s y_i}}{\sum_j e^{s_j}}\right)$

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



3.2
5.1
-1.7

scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss

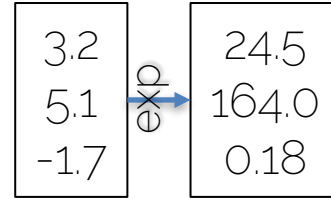
Cross-Entropy (Softmax)

Softmax $L_i = -\log\left(\frac{e^{s y_i}}{\sum_j e^{s_j}}\right)$

Score function $s = f(x_i, W)$
 e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Given a function with weights W ,
 Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss

Cross-Entropy (Softmax)

Softmax $L_i = -\log\left(\frac{e^{s y_i}}{\sum_j e^{s_j}}\right)$

Score function $s = f(x_i, W)$
 e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss

Given a function with weights W ,
 Training pairs $[x_i; y_i]$ (input and labels)



Cross-Entropy (Softmax)

Softmax
$$L_i = -\log\left(\frac{e^{s y_i}}{\sum_j e^{s_j}}\right)$$

Score function
$$s = f(x_i, W)$$

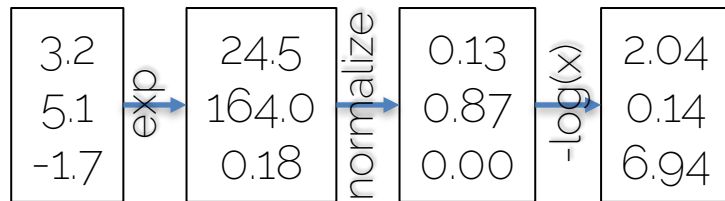
 e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
Loss		2.04	0.14	6.94

Given a function with weights W ,
 Training pairs $[x_i; y_i]$ (input and labels)



Cross-Entropy (Softmax)

Softmax $L_i = -\log\left(\frac{e^{s y_i}}{\sum_j e^{s_j}}\right)$

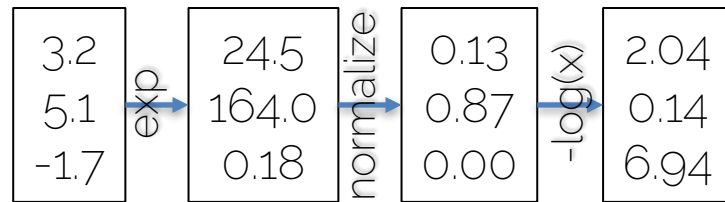
Score function $s = f(x_i, W)$
 e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Given a function with weights W ,
 Training pairs $[x_i; y_i]$ (input and labels)

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
Loss		2.04	0.14	6.94



$$\begin{aligned}
 L &= \frac{1}{N} \sum_{i=1}^N L_i = \\
 &= \frac{L_1 + L_2 + L_3}{3} = \\
 &= \frac{2.04 + 0.14 + 6.94}{3} = \\
 &= \mathbf{9.12}
 \end{aligned}$$

Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Score function $s = f(x_i, W)$

e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss 2.9

$$\begin{aligned} L_1 &= \\ &= \max(0, 5.1 - 3.2 + 1) + \\ &= \max(0, -1.7 - 3.2 + 1) = \\ &= \max(0, 2.9) + \max(0, -3.9) = \\ &= 2.9 + 0 = \\ &= \mathbf{2.9} \end{aligned}$$

Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

Loss	2.9	0
------	-----	---

$$\begin{aligned}
 L_2 &= \\
 &= \max(0, 1.3 - 4.9 + 1) + \\
 &= \max(0, 2.0 - 4.9 + 1) = \\
 &= \max(0, -2.6) + \max(0, -1.9) = \\
 &= 0 + 0 = \\
 &= 0
 \end{aligned}$$

Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1

$$\begin{aligned}
 L_3 &= \\
 &= \max(0, 2.2 - (-3.1) + 1) + \\
 &= \max(0, 2.5 - (-3.1) + 1) = \\
 &= \max(0, 5.3) + \max(0, 5.6) = \\
 &= 5.3 + 5.6 = \\
 &= \mathbf{10.9}
 \end{aligned}$$

Loss	2.9	0	10.9
------	-----	---	------

Hinge Loss (SVM Loss)

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Suppose: 3 training examples and 3 classes



scores	cat	3.2	1.3	2.2
	chair	5.1	4.9	2.5
	"car"	-1.7	2.0	-3.1
Loss		2.9	0	10.9

Full Loss (over all pairs):

$$\begin{aligned}
 L &= \frac{1}{N} \sum_{i=1}^N L_i = \\
 &= \frac{L_1 + L_2 + L_3}{3} = \\
 &= \frac{2.9 + 0 + 10.9}{3} = \\
 &= \mathbf{4.6}
 \end{aligned}$$

Weight Regularization & SVM Loss

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$

Full loss $L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \lambda R(W)$

$$L^1\text{-reg: } R^1(W) = \sum_{i=1}^N \sum_{j \neq y_i} |w_i|$$

$$L^2\text{-reg: } R^2(W) = \sum_{i=1}^N \sum_{j \neq y_i} w_i^2$$

Weight Regularization & SVM Loss

Multiclass SVM loss $L_i = \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$

Full loss $L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \lambda R(W)$

$$L^1\text{-reg: } R^1(W) = \sum_{i=1}^N \sum_{j \neq y_i} |w_i|$$

$$L^2\text{-reg: } R^2(W) = \sum_{i=1}^N \sum_{j \neq y_i} w_i^2$$

Example:

$$\mathbf{x} = [1, 1, 1, 1]$$

$$\mathbf{w}_1 = [1, 0, 0, 0]$$

$$R^2(\mathbf{w}_1) = 1$$

$$\mathbf{w}_2 = [0.25, 0.25, 0.25, 0.25]$$

$$R^2(\mathbf{w}_2) = 0.25^2 + 0.25^2 + 0.25^2 + 0.25^2 = 0.25$$

$$\mathbf{w}_1^T \mathbf{x} = \mathbf{w}_2^T \mathbf{x} = 1$$

Hinge Loss vs Softmax

Hinge loss: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Softmax: $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

Hinge Loss vs Softmax

$$\text{Hinge loss: } L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\text{Softmax: } L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

Given the following scores:

$$s = [5, -3, 2]$$

$$s = [5, 10, 10]$$

$$s = [5, -20, -20]$$

$$y_i = 0$$

Hinge Loss vs Softmax

Hinge loss: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Softmax: $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

Given the following scores:

$$s = [5, -3, 2]$$

Hinge loss:

$$\begin{aligned} &\max(0, -3 - 5 + 1) + \\ &\max(0, 2 - 5 + 1) = 0 \end{aligned}$$

$$s = [5, 10, 10]$$

$$\begin{aligned} &\max(0, 10 - 5 + 1) + \\ &\max(0, 10 - 5 + 1) = 12 \end{aligned}$$

$$s = [5, -20, -20]$$

$$\begin{aligned} &\max(0, -20 - 5 + 1) + \\ &\max(0, -20 - 5 + 1) = 0 \end{aligned}$$

$$y_i = 0$$

Hinge Loss vs Softmax

Hinge loss: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Softmax: $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

Given the following scores:

$$s = [5, -3, 2]$$

Hinge loss:

$$\max(0, -3 - 5 + 1) + \max(0, 2 - 5 + 1) = 0$$

Softmax loss:

Google...
 $-\ln(e^5 / (e^5 + e^{-3} + e^2)) = 0.05$

$$s = [5, 10, 10]$$

$$\max(0, 10 - 5 + 1) + \max(0, 10 - 5 + 1) = 12$$

Google...
 $-\ln(e^5 / (e^5 + e^{10} + e^{10})) = 5.70$

$$s = [5, -20, -20]$$

$$\max(0, -20 - 5 + 1) + \max(0, -20 - 5 + 1) = 0$$

Google...
 $-\ln(e^5 / (e^5 + e^{-20} + e^{-20})) = 2e^{-11}$

$y_i = 0$

Hinge Loss vs Softmax

Hinge loss: $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

Softmax: $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

Given the following scores:

$$s = [5, -3, 2]$$

Hinge loss:

$$\max(0, -3 - 5 + 1) + \max(0, 2 - 5 + 1) = 0$$

Softmax loss:

Google...
 $-\ln(e^5 / (e^5 + e^{-3} + e^2)) = 0.05$

$$s = [5, 10, 10]$$

$$\max(0, 10 - 5 + 1) + \max(0, 10 - 5 + 1) = 12$$

Google...
 $-\ln(e^5 / (e^5 + e^{10} + e^{10})) = 5.70$

$$s = [5, -20, -20]$$

$$\max(0, -20 - 5 + 1) + \max(0, -20 - 5 + 1) = 0$$

Google...
 $-\ln(e^5 / (e^5 + e^{-20} + e^{-20})) = 2e^{-11}$

$y_i = 0$

Softmax *always* wants to improve!
Hinge Loss saturates

Loss in Compute Graph

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

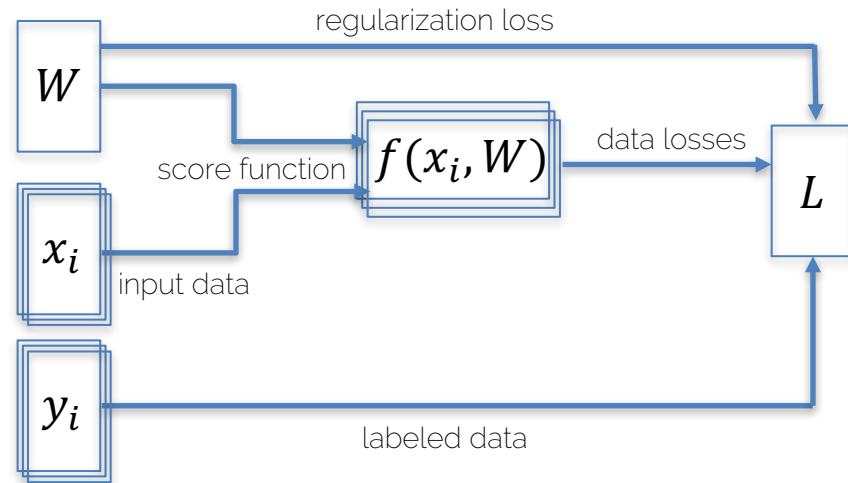
Given a function with weights W ,
Training pairs $[x_i; y_i]$ (input and labels)

Softmax $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

SVM $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

e.g., L^2 -reg: $R^2(W) = \sum_{i=1}^N w_i^2$

Full Loss $L = \frac{1}{N} \sum_{i=1}^N L_i + R^2(W)$



Compute Graphs

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

Softmax $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

SVM $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

e.g., L^2 -reg: $R^2(W) = \sum_{i=1}^N w_i^2$

Full Loss $L = \frac{1}{N} \sum_{i=1}^N L_i + R^2(W)$

Compute Graphs

Score function $s = f(x_i, W)$
e.g., $f(x_i, W) = W \cdot [x_0, x_1, \dots, x_N]^T$

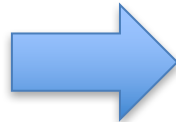
Softmax $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

SVM $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

e.g., L^2 -reg: $R^2(W) = \sum_{i=1}^N w_i^2$

Full Loss $L = \frac{1}{N} \sum_{i=1}^N L_i + R^2(W)$

Want to find optimal W . I.e., weights are unknowns of optimization problem

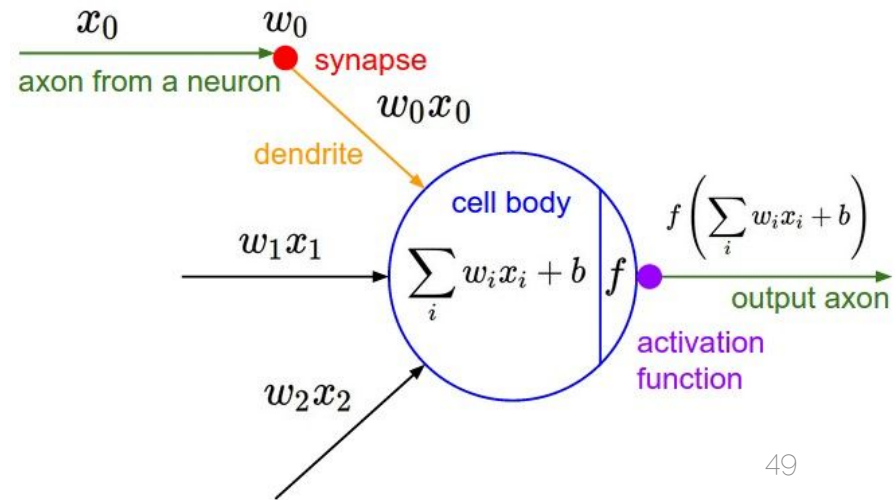
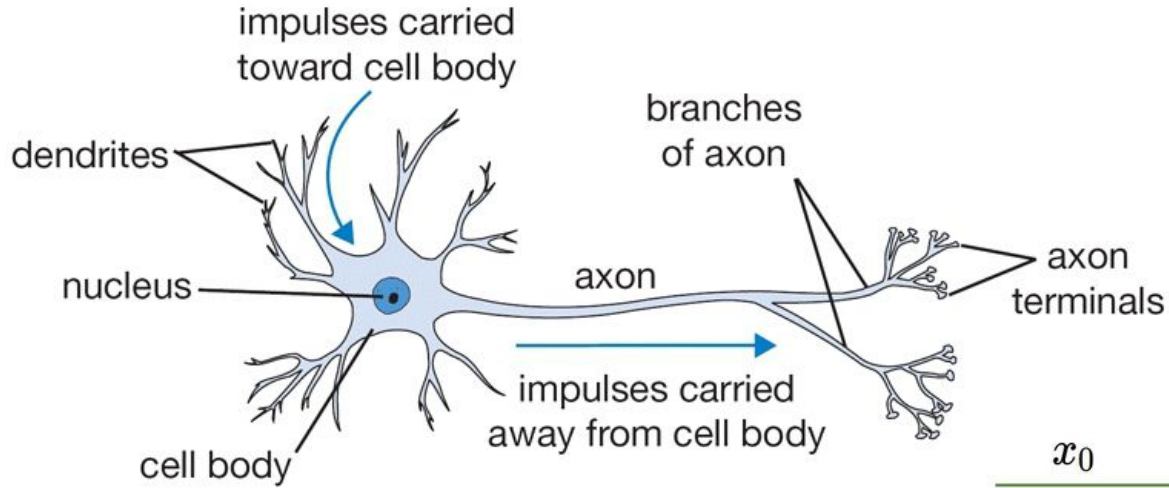


Compute gradient w.r.t. W .

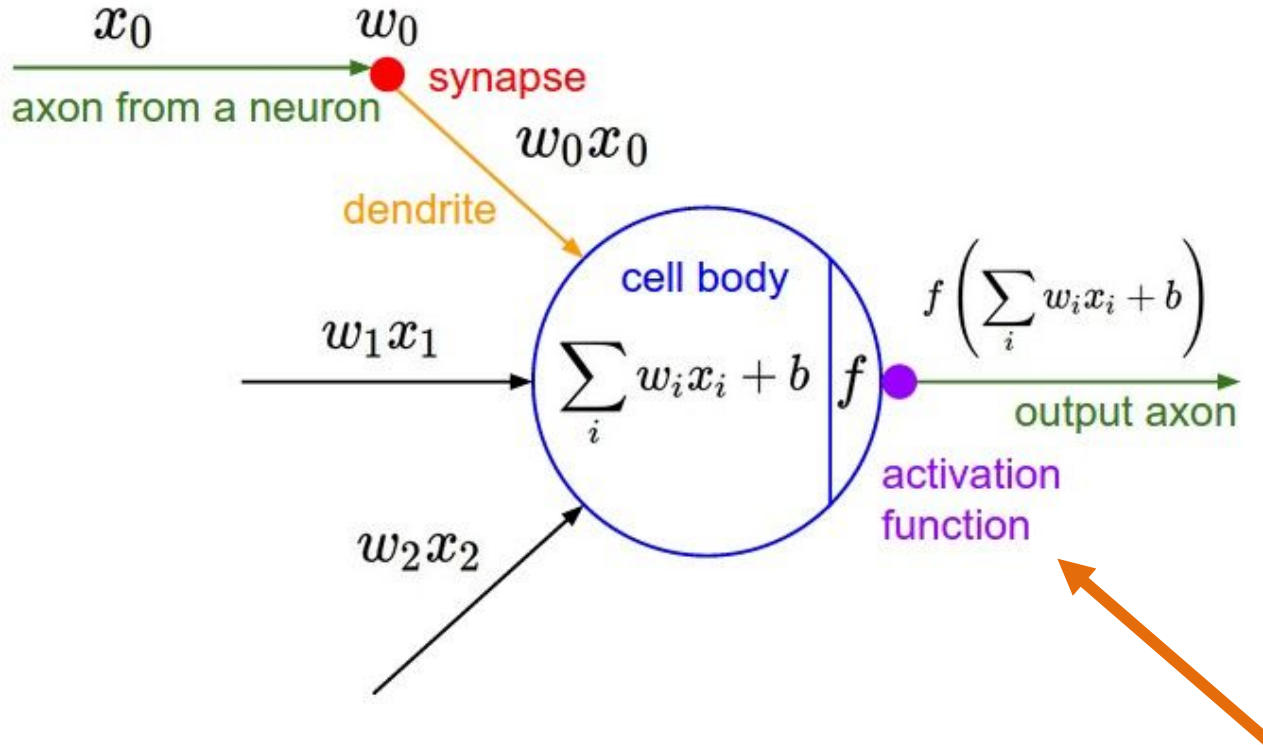
Gradient $\nabla_W L$ is computed via backpropagation

Activation functions

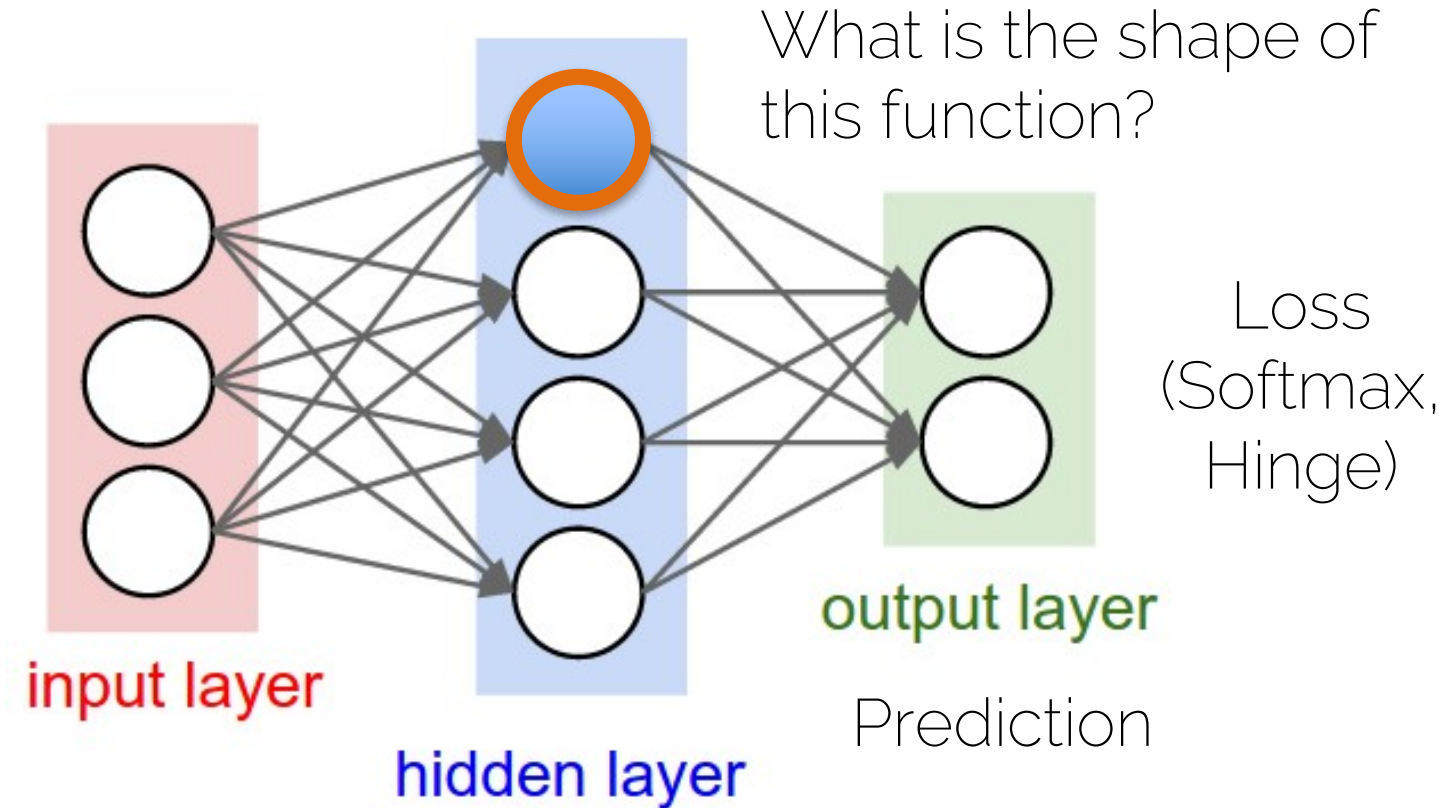
Neurons



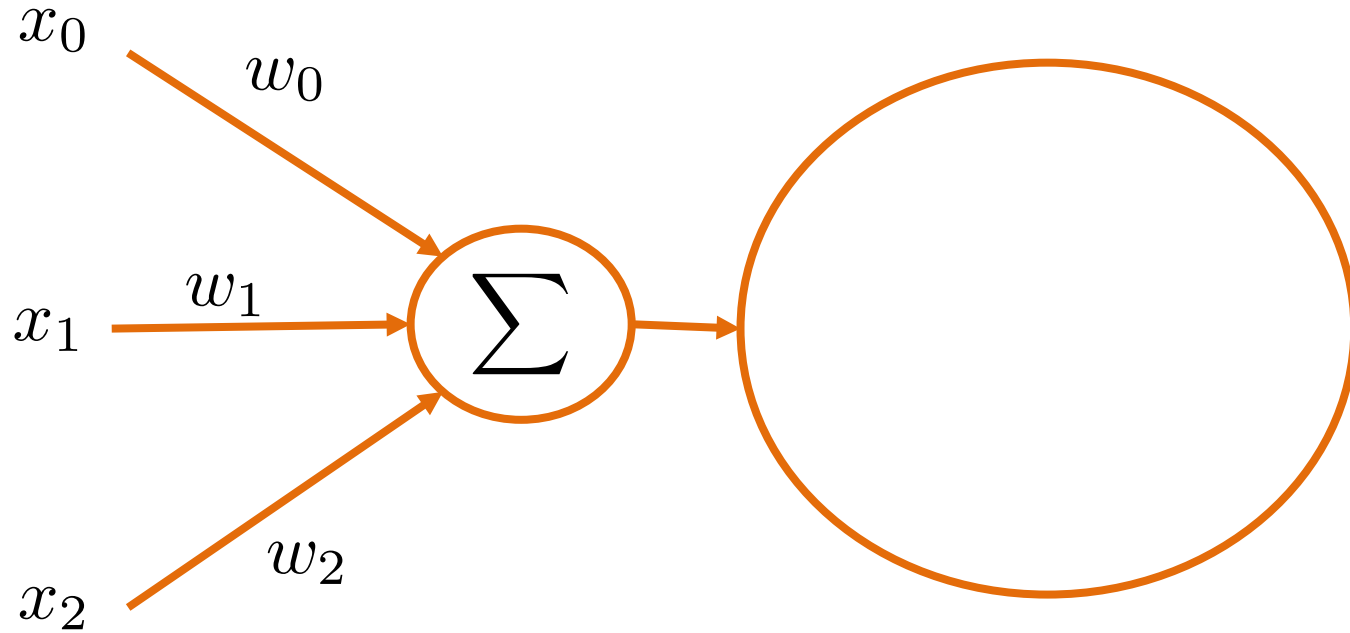
Neurons



Neural networks

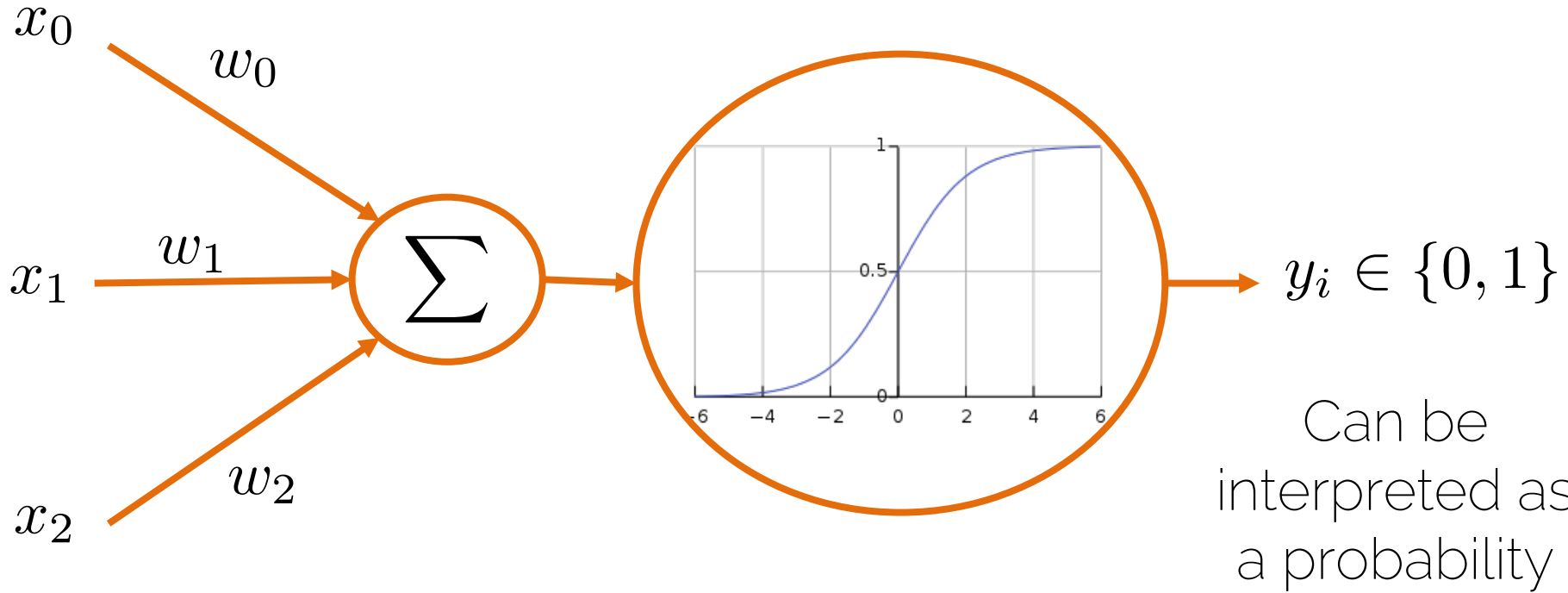


Activation functions or hidden units



Sigmoid

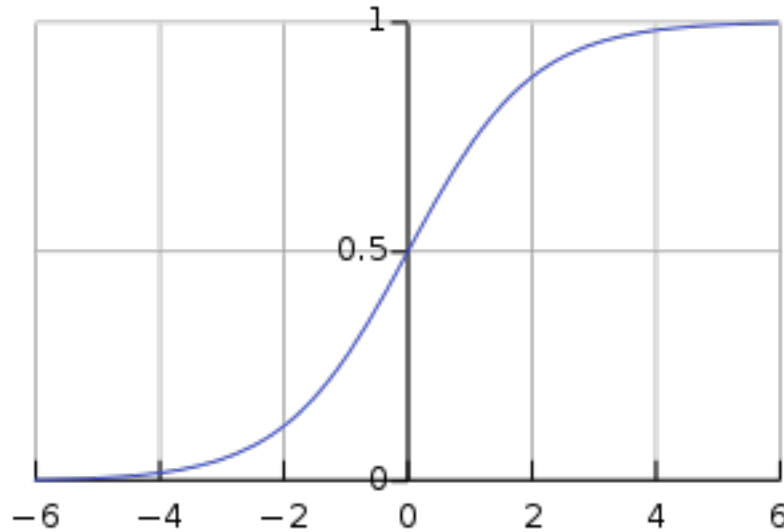
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Sigmoid

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

Forward



$$\frac{\partial L}{\partial x} = \frac{\partial \sigma}{\partial x} \frac{\partial L}{\partial \sigma}$$



$$\frac{\partial \sigma}{\partial x}$$

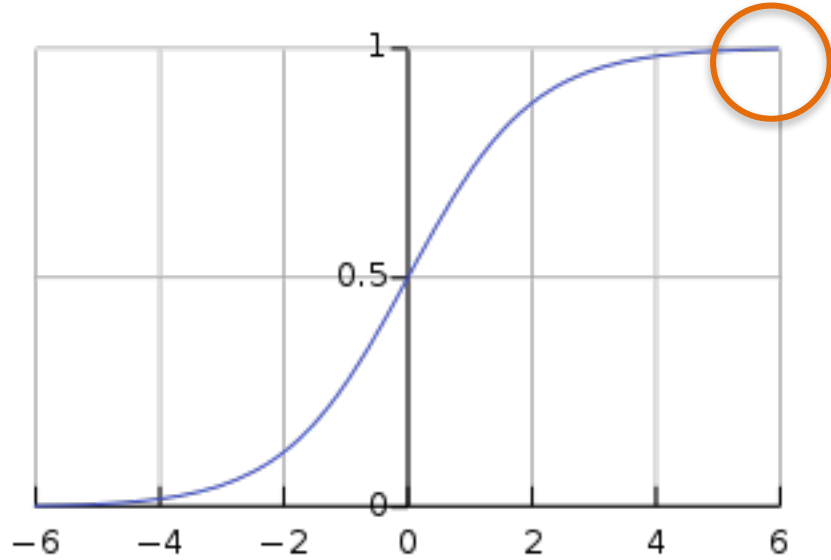


$$\frac{\partial L}{\partial \sigma}$$

Sigmoid

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

Forward



$x = 6$

X Saturated neurons kill the gradient flow

~~$$\frac{\partial L}{\partial x} = \frac{\partial \sigma}{\partial x} \frac{\partial L}{\partial \sigma}$$~~



$$\frac{\partial \sigma}{\partial x}$$

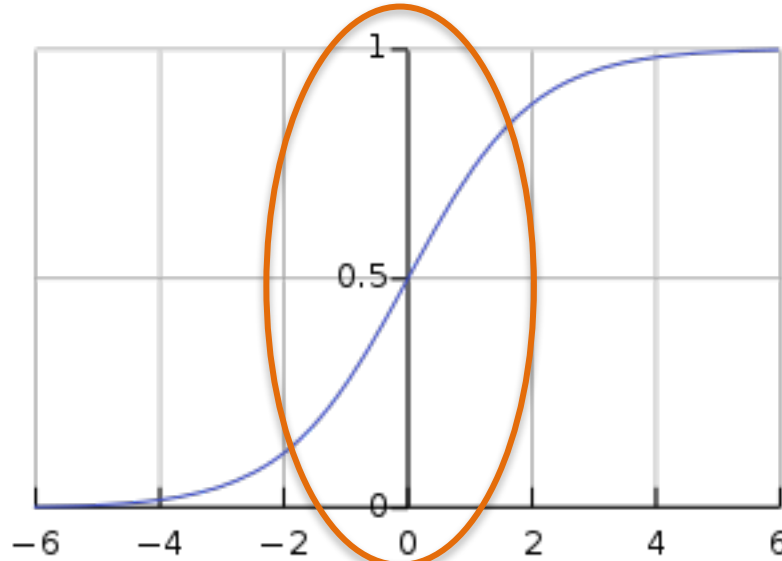


$$\frac{\partial L}{\partial \sigma}$$

Sigmoid

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

Forward



Active region
for gradient
descent

$$\frac{\partial L}{\partial x} = \frac{\partial \sigma}{\partial x} \frac{\partial L}{\partial \sigma}$$



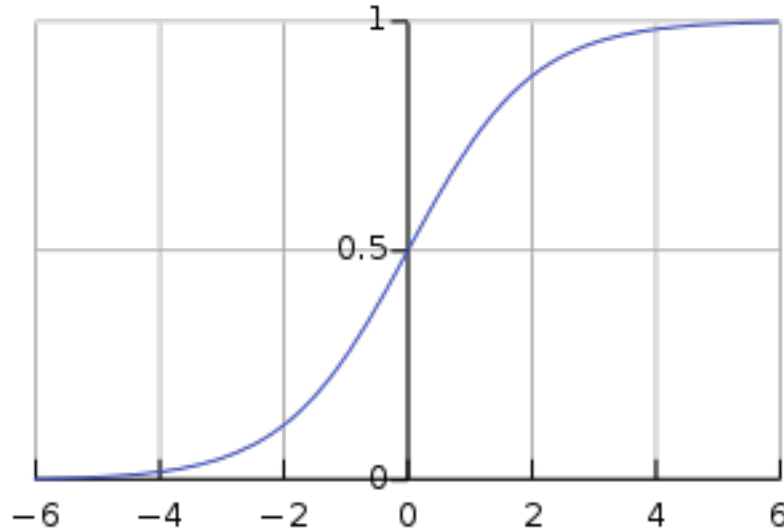
$$\frac{\partial \sigma}{\partial x}$$



$$\frac{\partial L}{\partial \sigma}$$

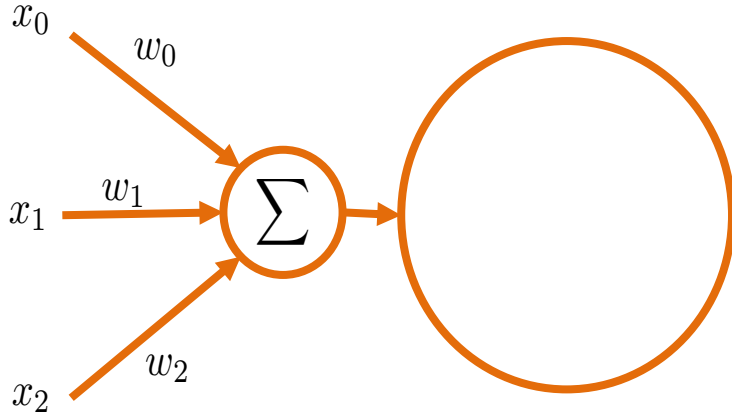
Sigmoid

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



Output is
always
positive

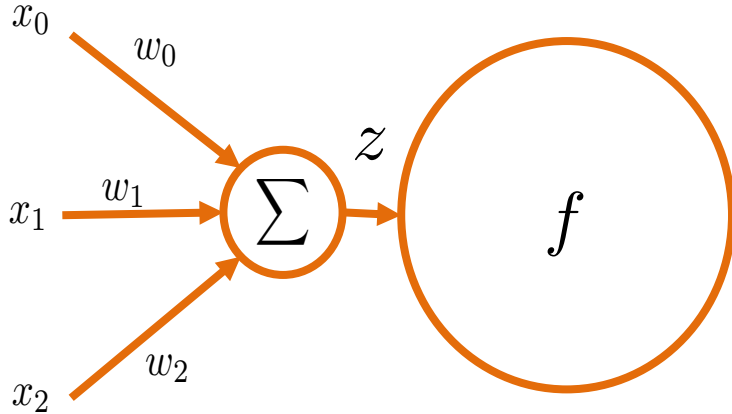
Problem of positive output



$$f \left(\sum_i w_i x_i + b \right)$$

We want to compute the gradient wrt the weights

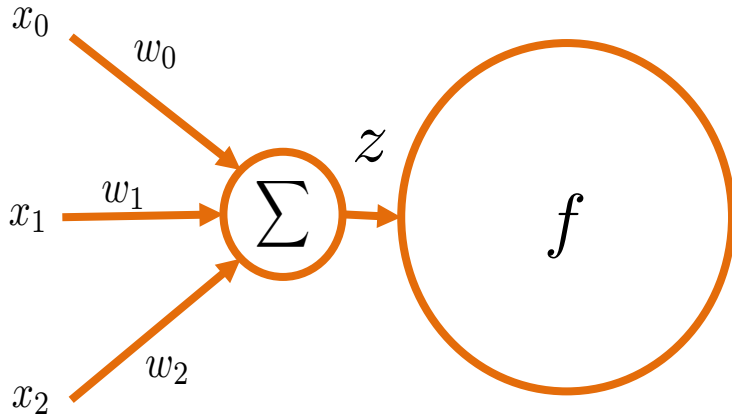
Problem of positive output



$$f \left(\underbrace{\sum_i w_i x_i + b}_{z} \right)$$
$$\frac{\partial z}{\partial w} = x_i > 0$$

We want to compute the gradient wrt the weights

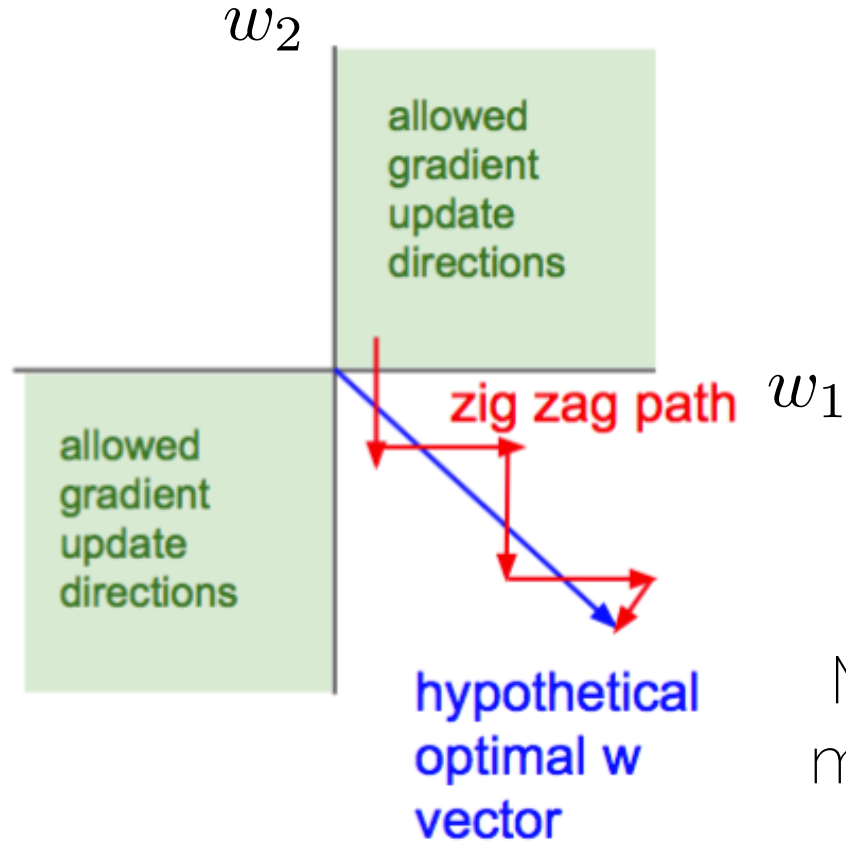
Problem of positive output



$$\frac{\partial f}{\partial z} \rightarrow f \left(\underbrace{\sum_i w_i x_i + b}_{\frac{\partial z}{\partial w} = x_i > 0} \right)$$

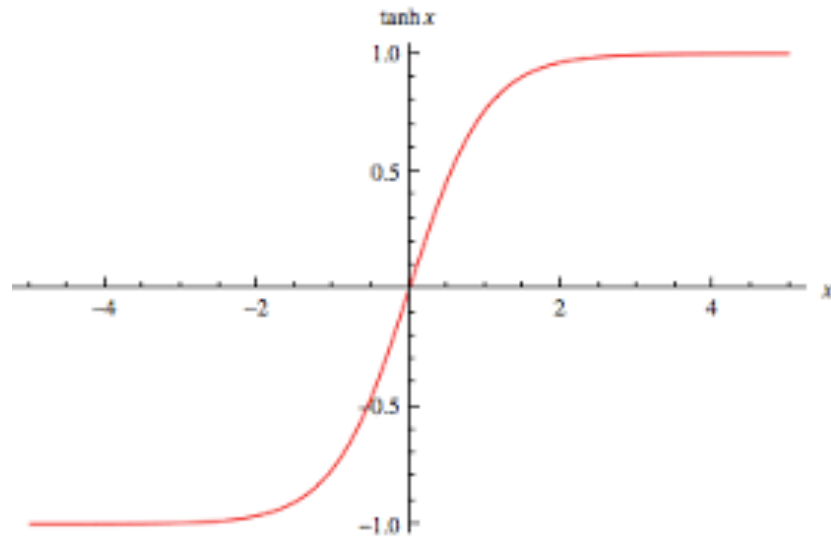
It is going to be either positive or negative for all weights

Problem of positive output



More on zero-mean data later

tanh



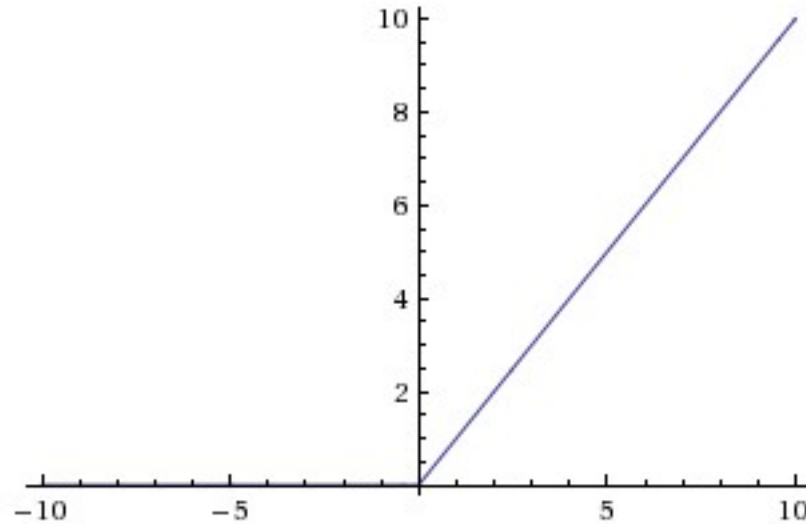
✗ Still saturates

✗ Still saturates

✓ Zero-centered

Rectified Linear Units (ReLU)

$$\sigma(x) = \max(0, x)$$



Large and consistent gradients ✓

✓ Fast convergence

✓ Does not saturate

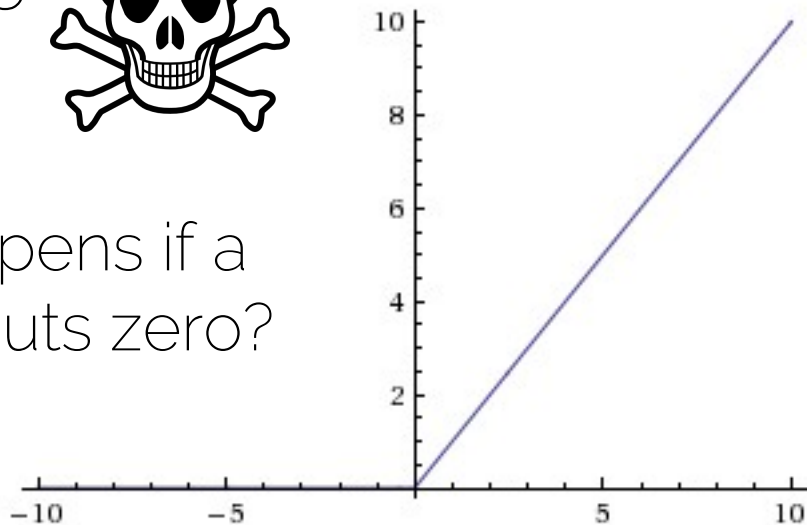
Rectified Linear Units (ReLU)



Dead ReLU



What happens if a ReLU outputs zero?



Large and consistent gradients ✓



Fast convergence



Does not saturate

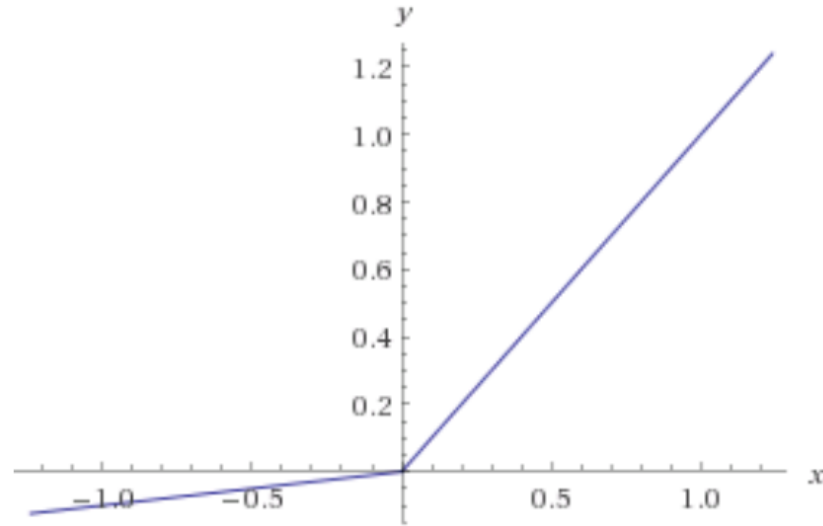
Rectified Linear Units (ReLU)

- Initializing ReLU neurons with slightly positive biases (0.1) makes it likely that they stay active for most inputs

$$f \left(\sum_i w_i x_i + b \right)$$

Leaky ReLU

$$\sigma(x) = \max(0.01x, x)$$



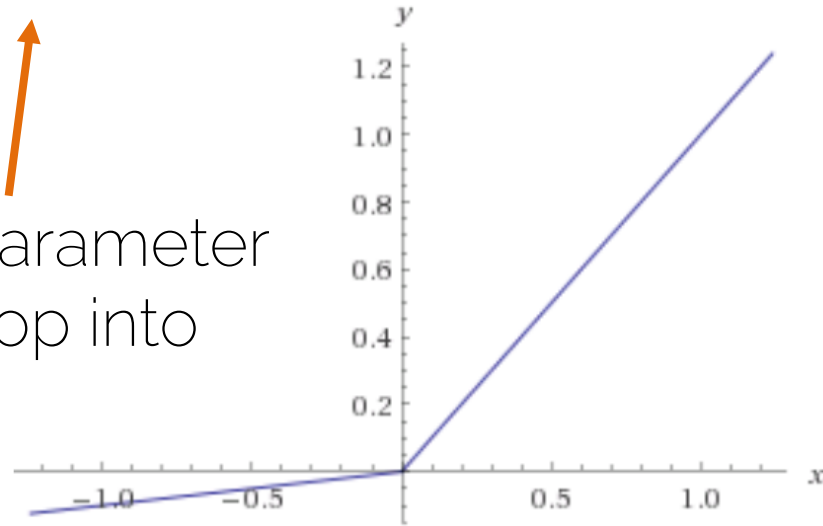
Does not die

Parametric ReLU

$$\sigma(x) = \max(\alpha x, x)$$

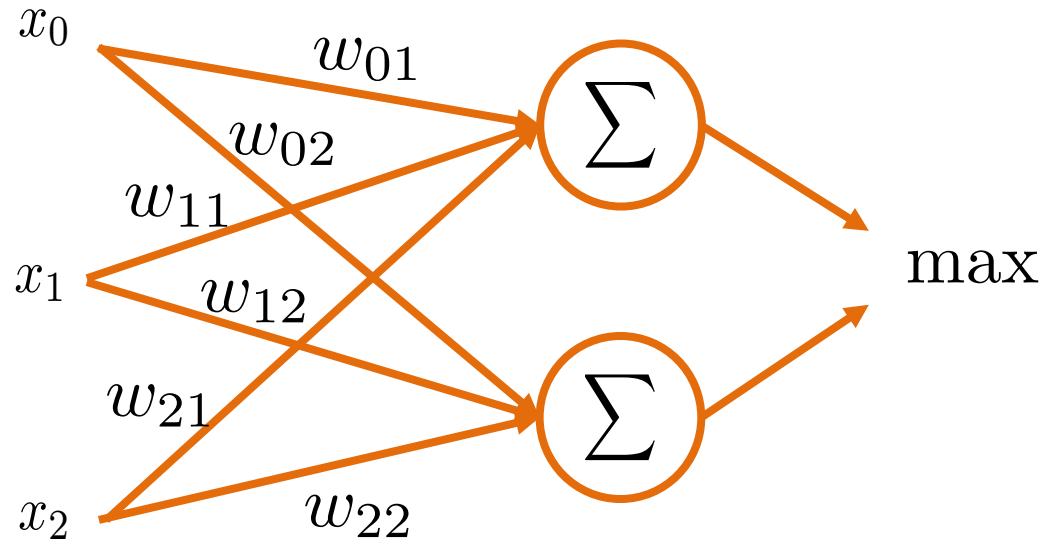


One more parameter
to backprop into

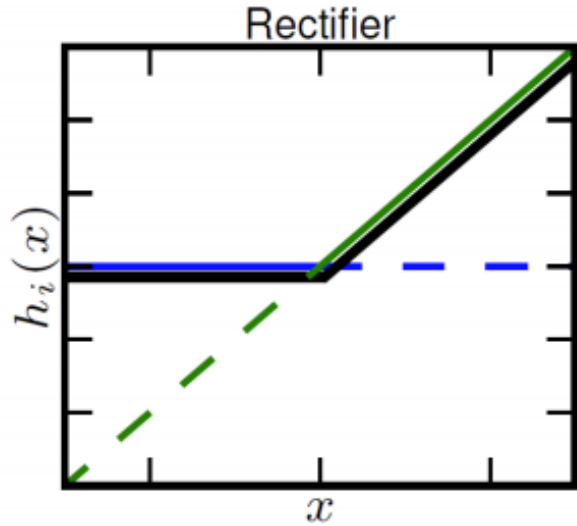


Does not die

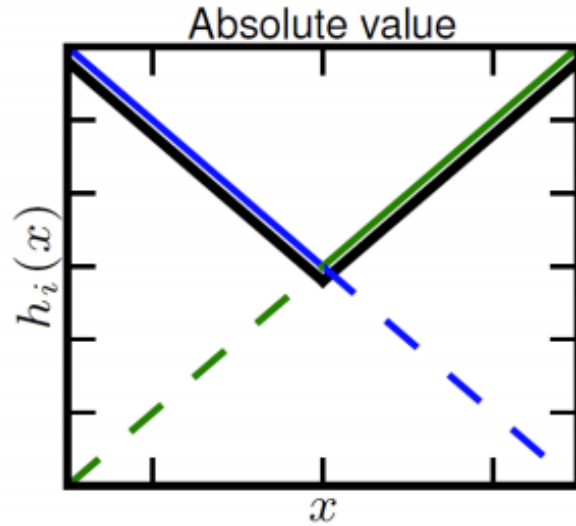
Maxout units



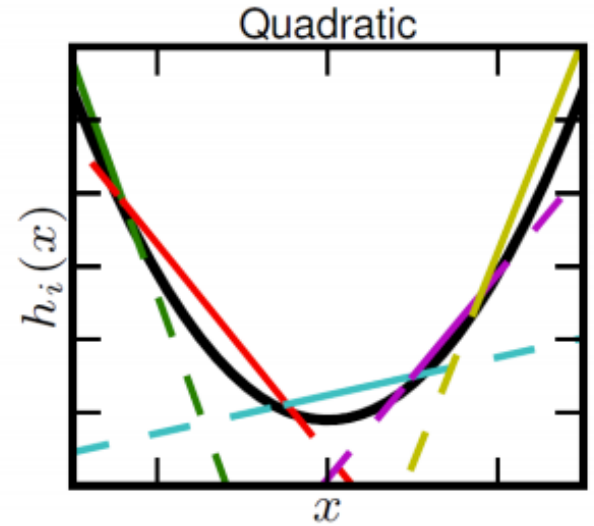
Maxout units



$k=2$



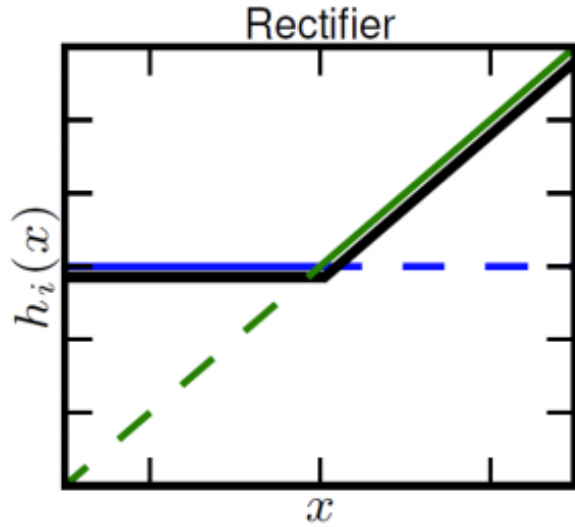
$k=2$



$k=5$

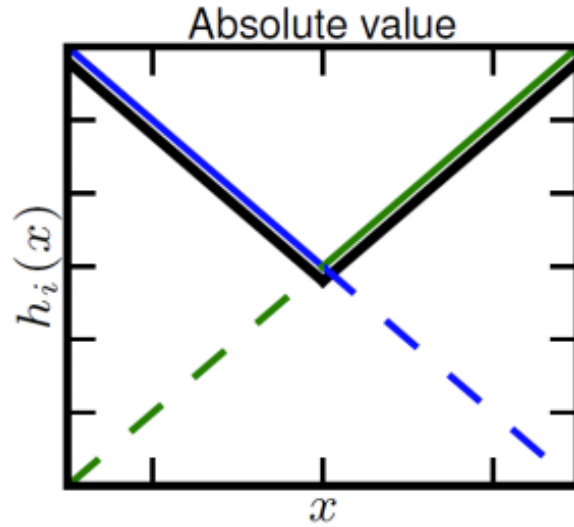
Piecewise linear approximation of a convex function with N pieces

Maxout units



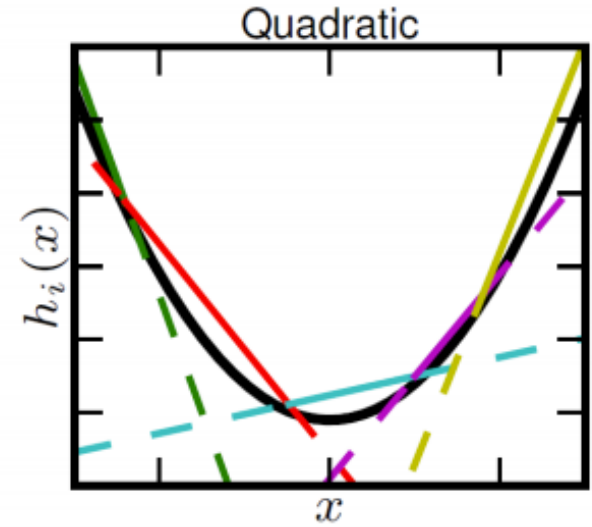
$k=2$

✓ Generalization of ReLUs



$k=2$

✓ Linear regimes



$k=5$

✓ Does not die
✓ Does not saturate

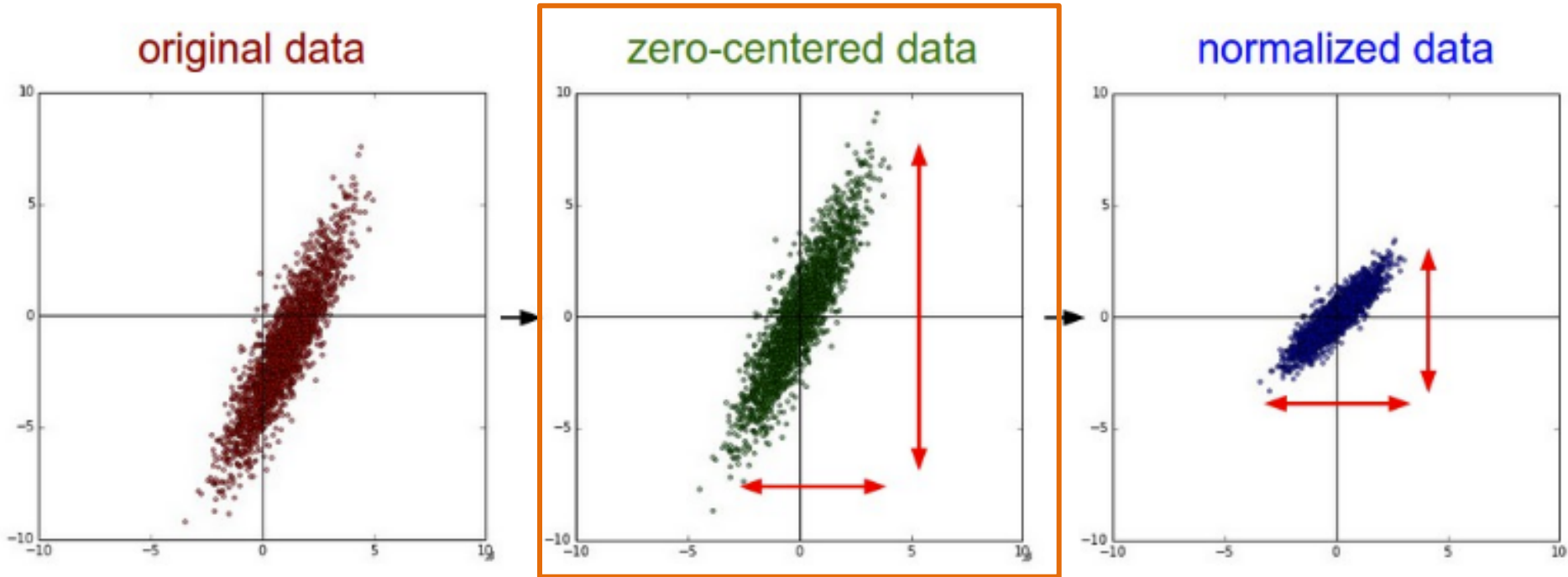
✗ Increase of the number of parameters

Quick guide

- Sigmoid is not really used
- ReLU is the standard choice
- Second choice are the variants of ReLU or Maxout
- Recurrent nets will require tanh or similar

A quick word on data pre-processing

Data pre-processing

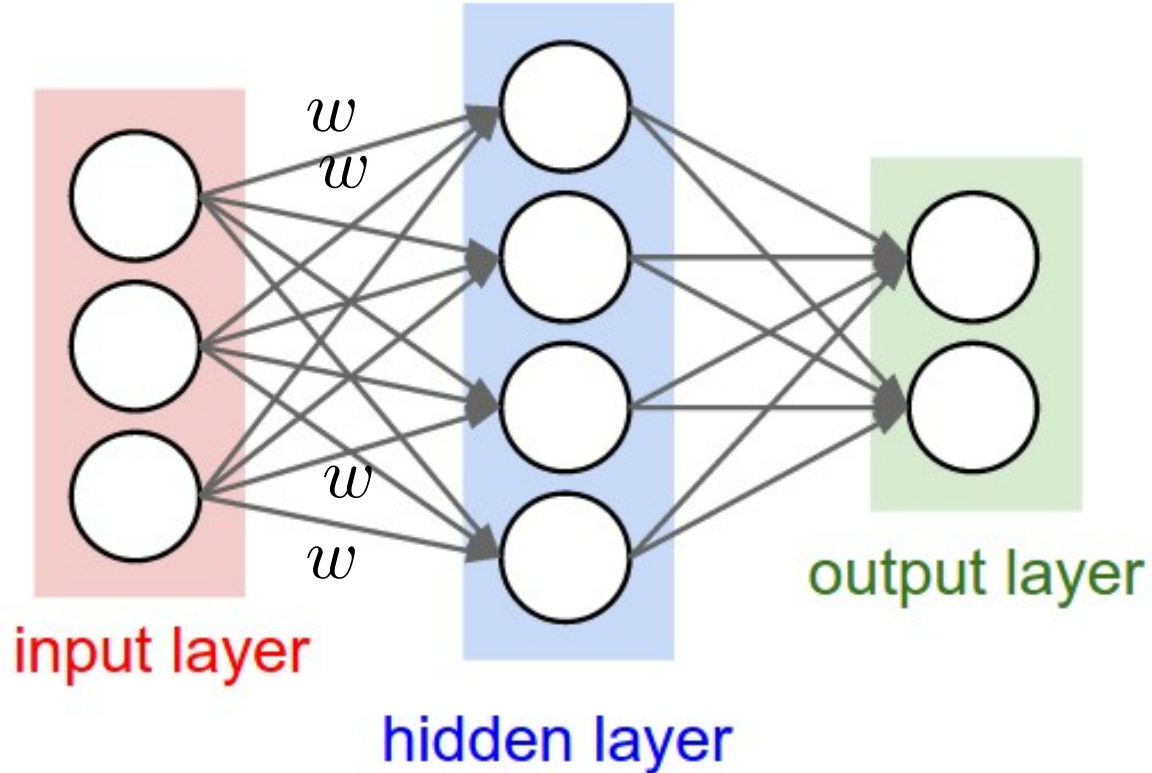


For images subtract the mean image (AlexNet) or per-channel mean (VGG-Net)

Weight initialization

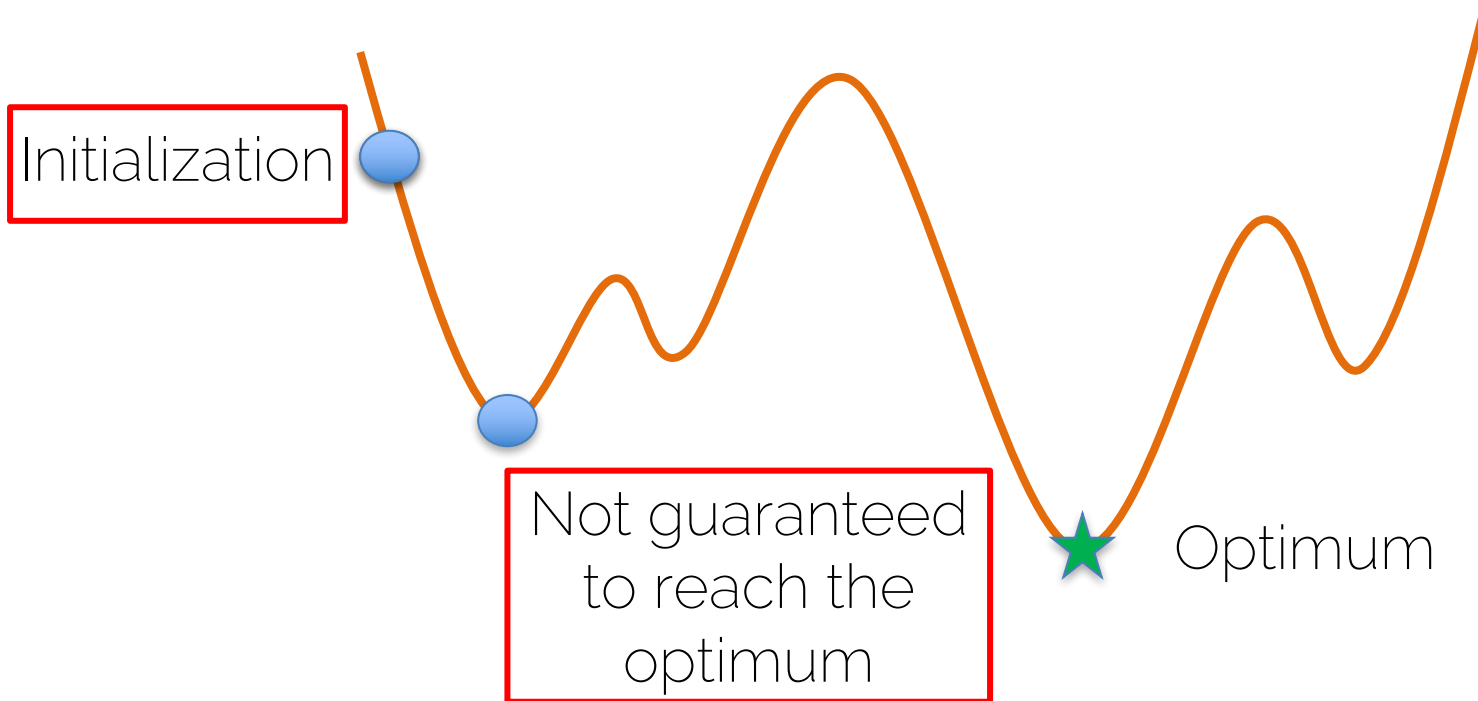
How do I start?

Forward



Initialization is extremely important

$$\mathbf{x}^* = \arg \min f(\mathbf{x})$$



How do I start?

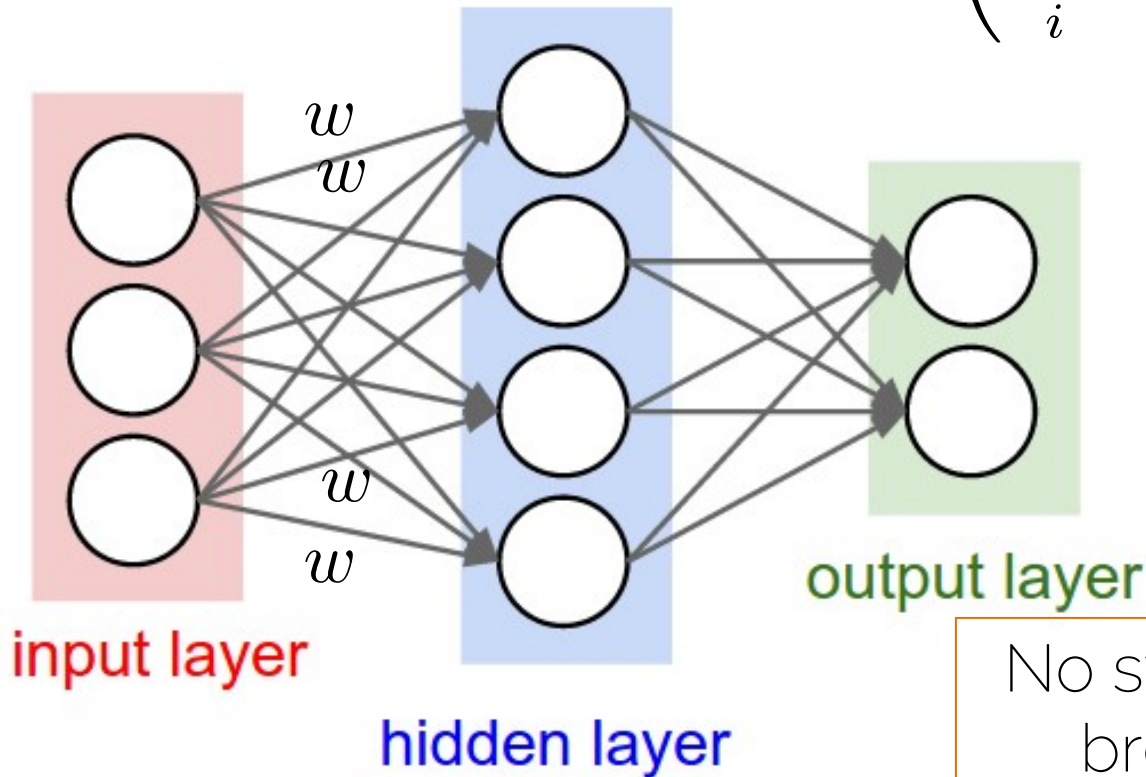
Forward



$$f \left(\sum_i w_i x_i + b \right)$$

$w = 0$

What happens to the gradients?

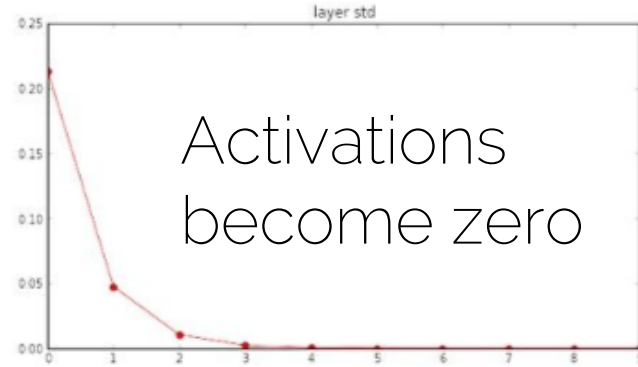
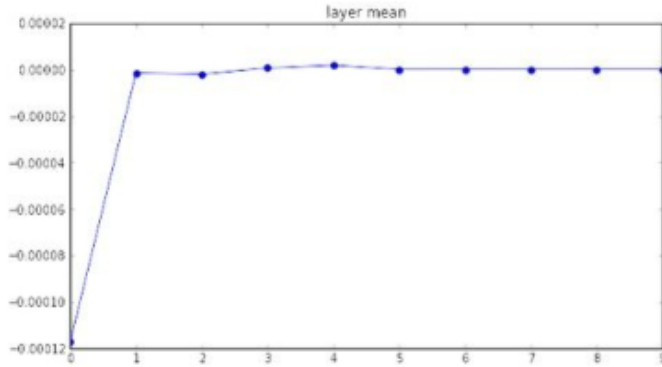


No symmetry breaking⁷⁷

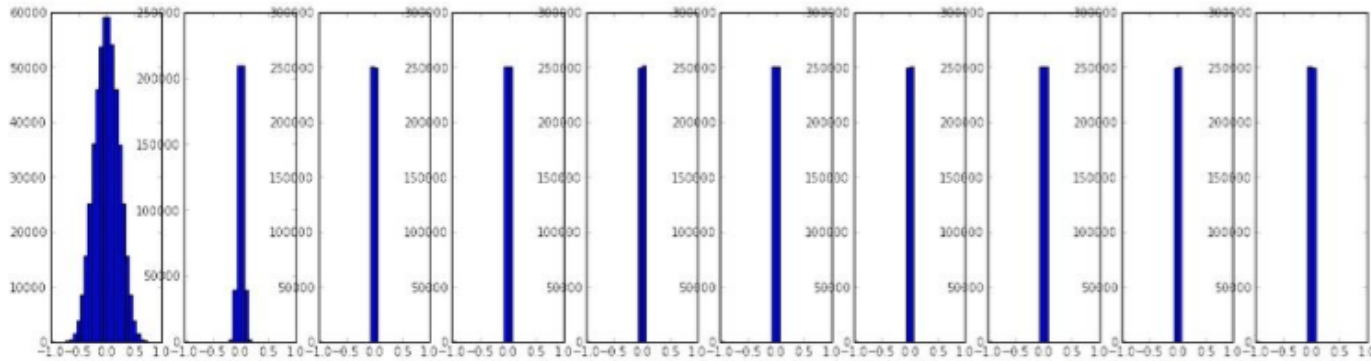
Small random numbers

- Gaussian with zero mean and standard deviation 0.01
- Let us see what happens:
 - Network with 10 layers with 500 neurons each
 - Tanh as activation functions
 - Input unit Gaussian data

Small random numbers



Input

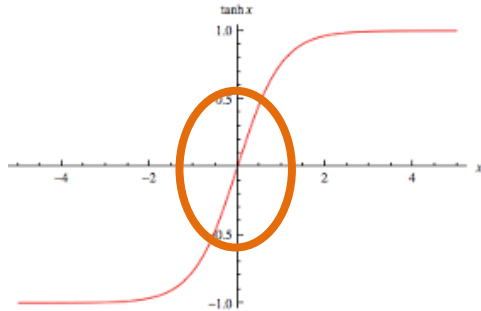
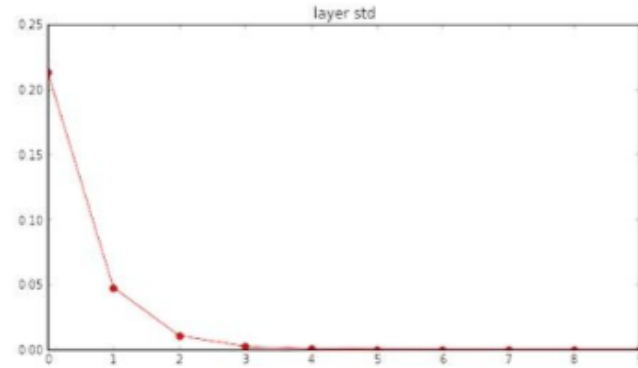
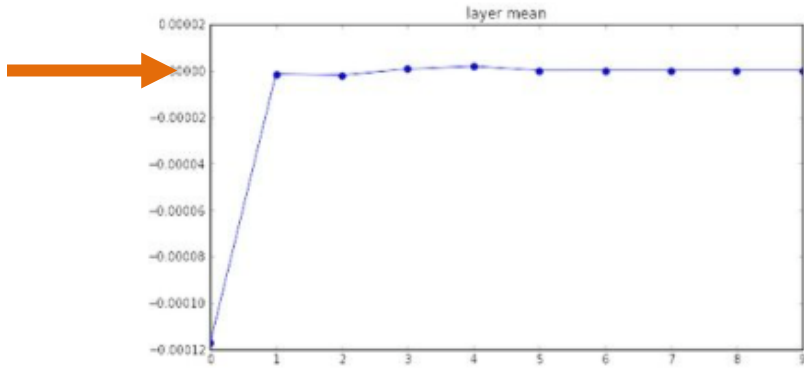


Last layer

Forward



Small random numbers

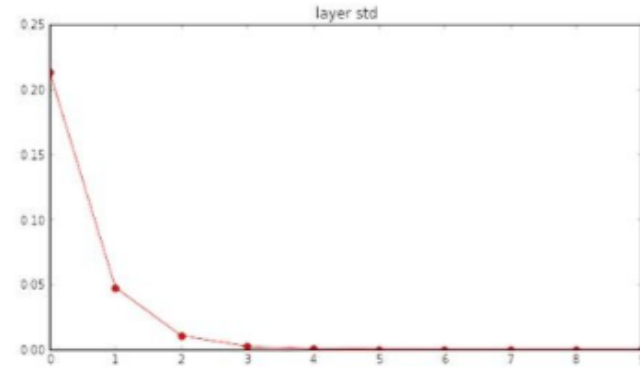
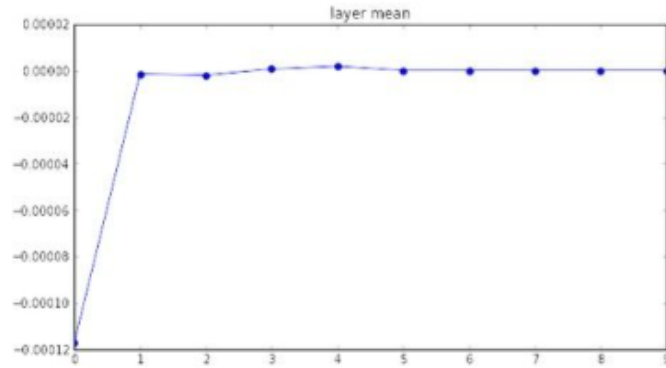


$$f \left(\sum_i \text{small } w_i c_i + b \right)$$

Forward



Small random numbers



Gradients
vanish

$$f \left(\sum_i w_i x_i + b \right)$$

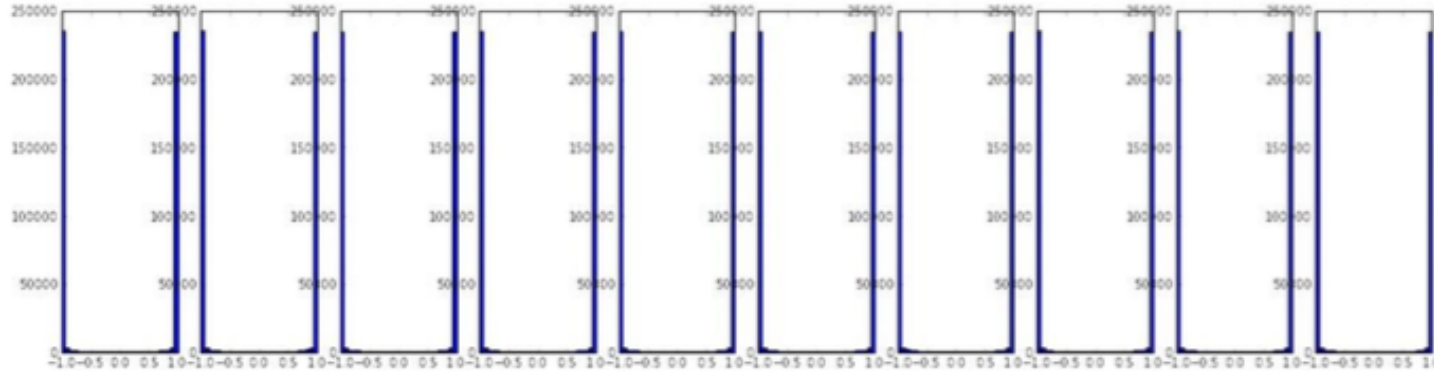
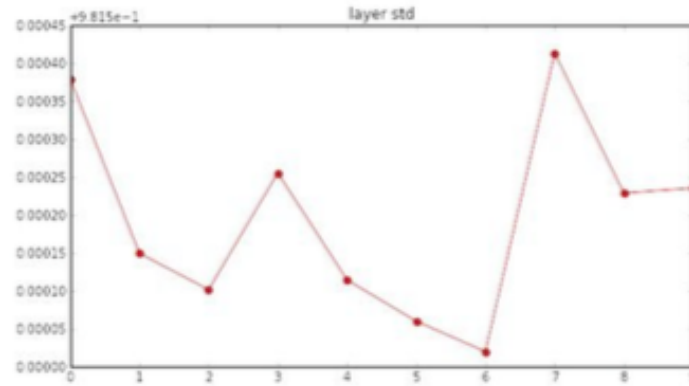
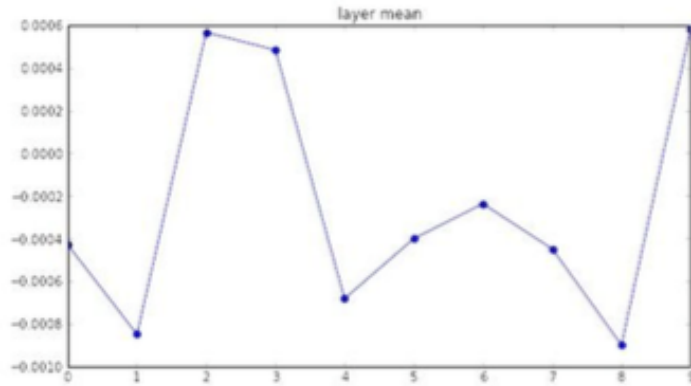
Backward



Big random numbers

- Gaussian with zero mean and standard deviation 1
- Let us see what happens:
 - Network with 10 layers with 500 neurons each
 - Tanh as activation functions
 - Input unit Gaussian data

Big random numbers



Everything is saturated

How to solve this?

- Next lecture!

Administrative Things

- Next Thursday Q&A Session
- Upcoming lectures:
 - More about neural networks (regularization, BN)
 - CNN 3 lecture block