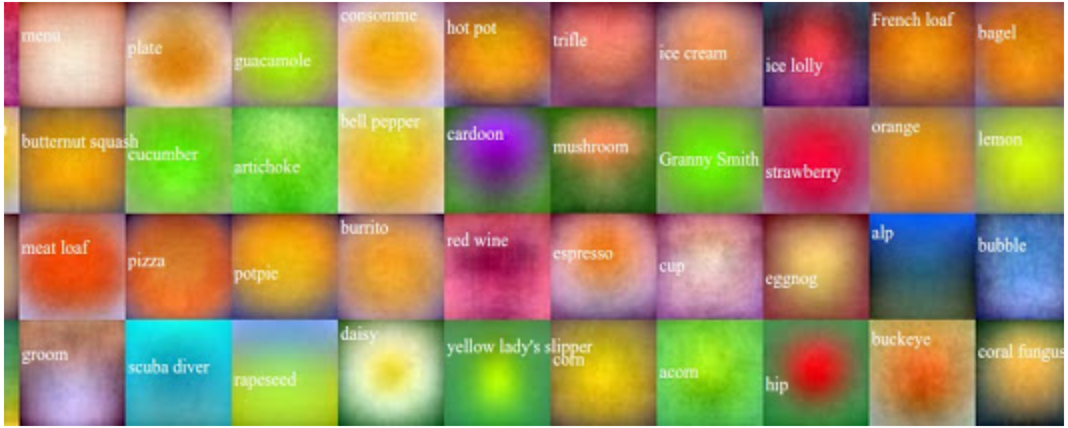
Going Deep into Neural Networks

Beyond linear

- Linear score function $f = Wx$

On CIFAR-10

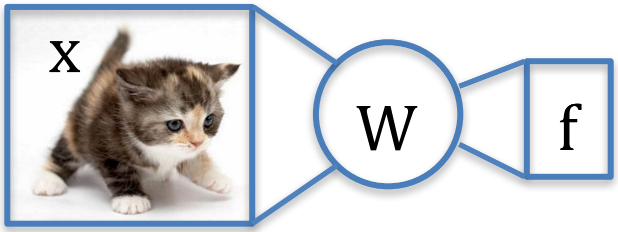


On ImageNet

Credit: Li/Karpathy/Johnson

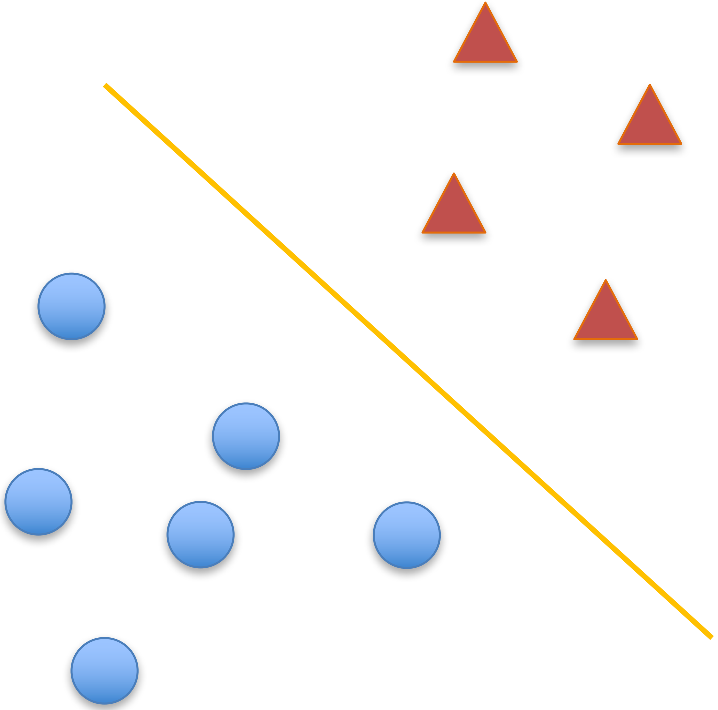
Beyond linear

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



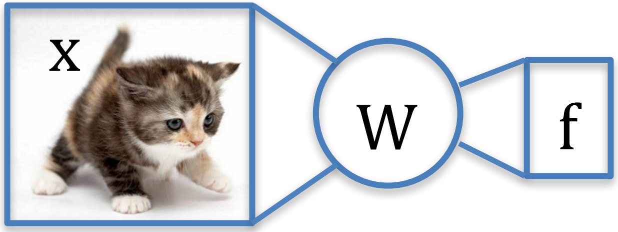
128×128 10

LINEAR TRANSFORMATION



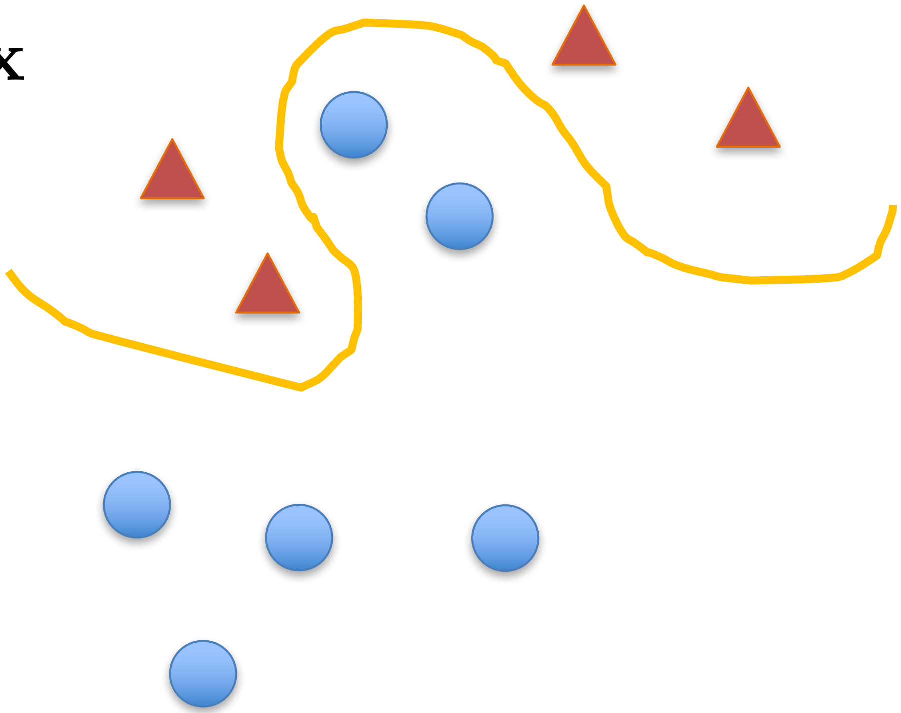
Beyond linear

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



128×128 10

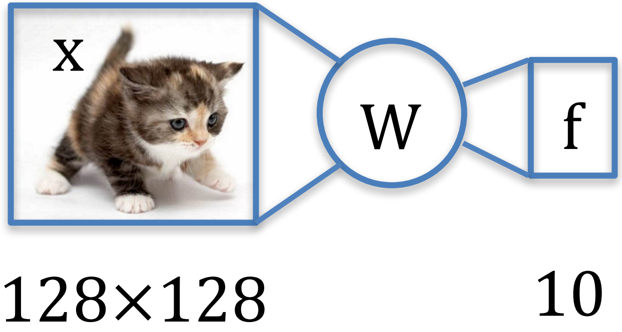
LINEAR TRANSFORMATION



Kernel trick

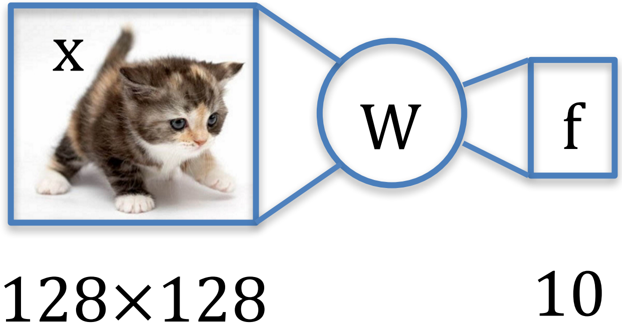
1-layer network: $f = \mathbf{W}\mathbf{x}$

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



Neural networks

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



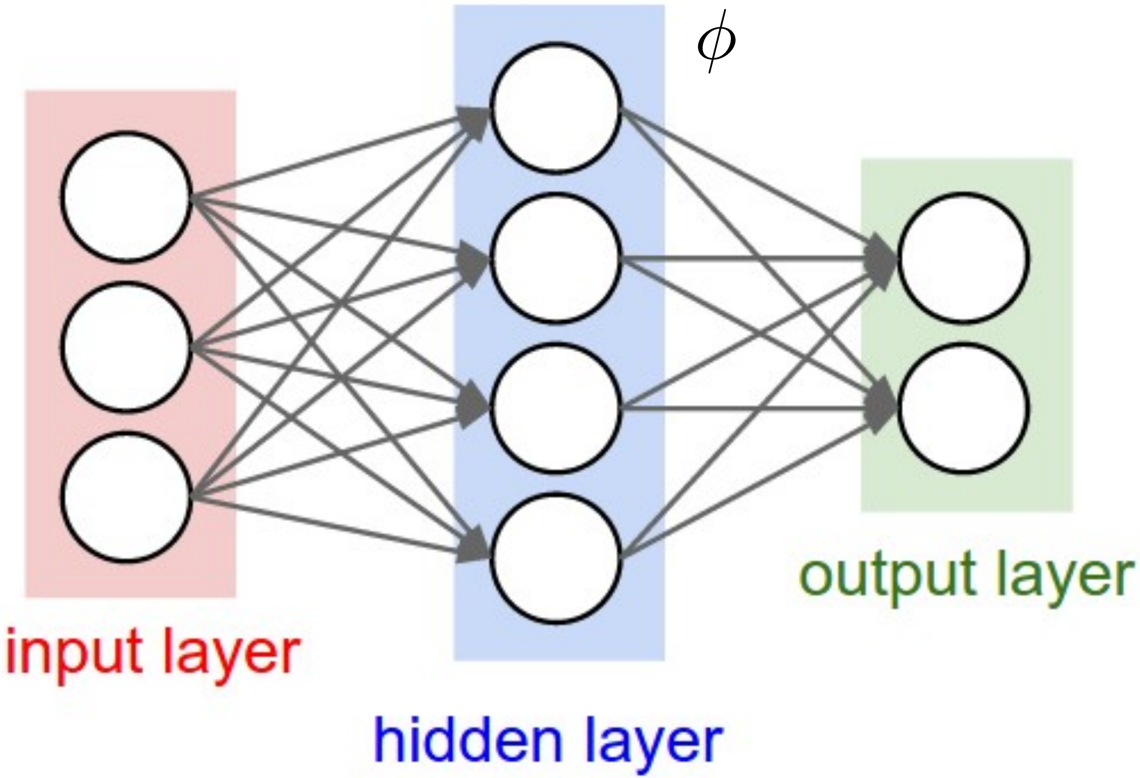
$$f = \mathbf{W}\phi(\mathbf{x}; \boldsymbol{\theta})$$

kernel

parameters

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

Neural Network



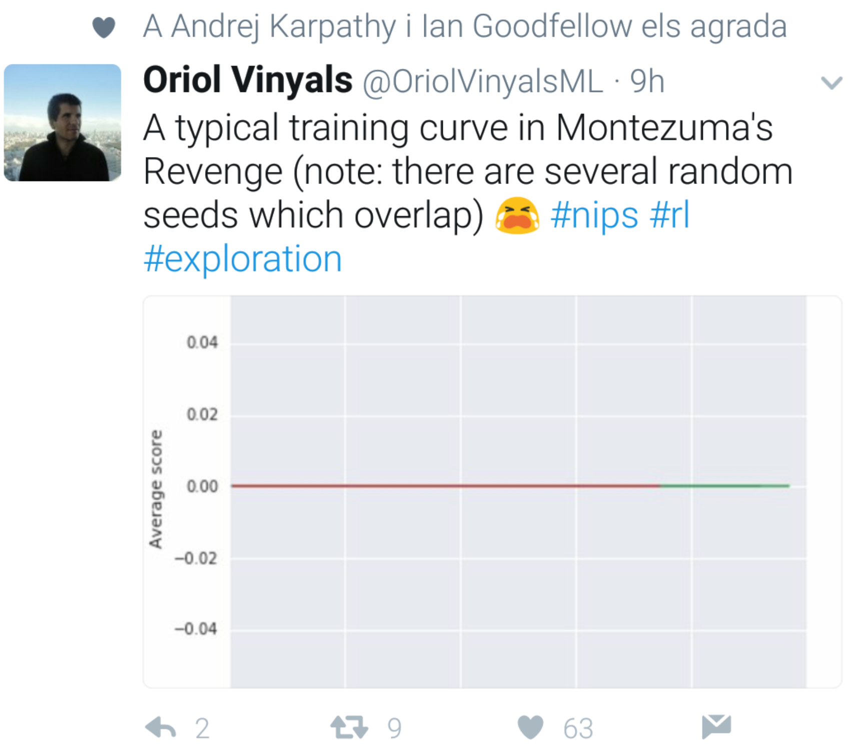
Also SVM is in this category

Credit: Li/Karpathy/Johnson

Neural Network

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

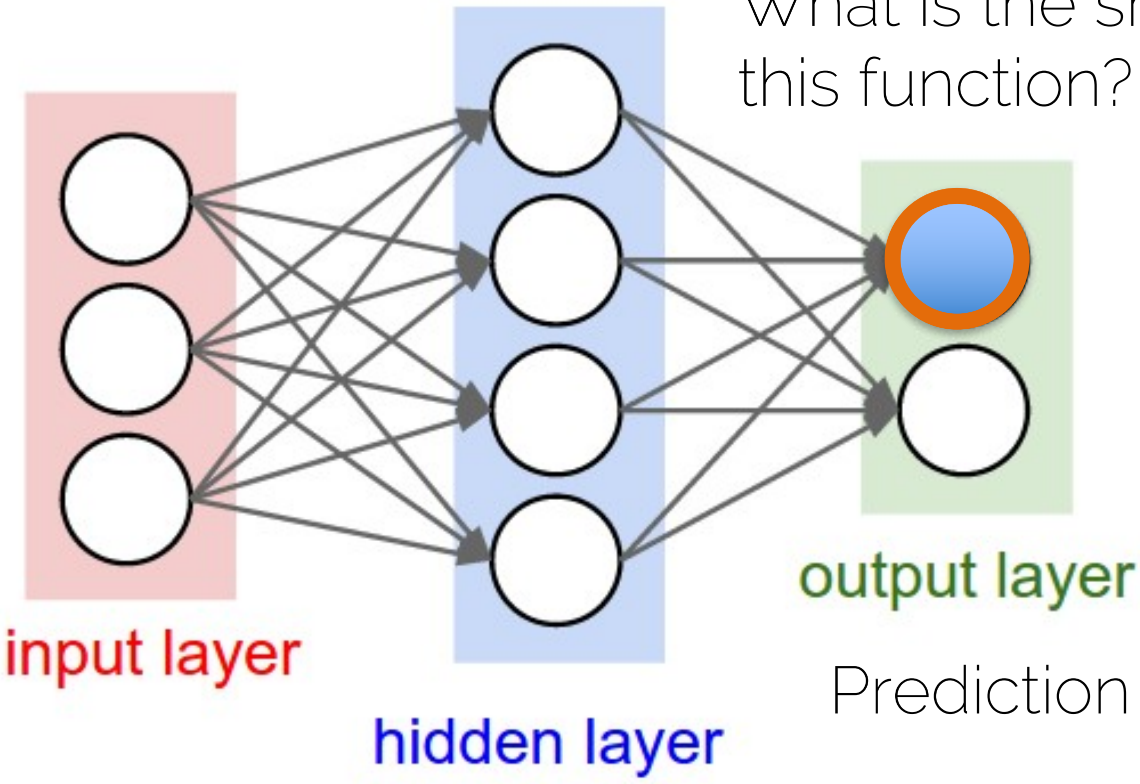
A typical Deep Learner day



Output functions

Neural networks

What is the shape of this function?

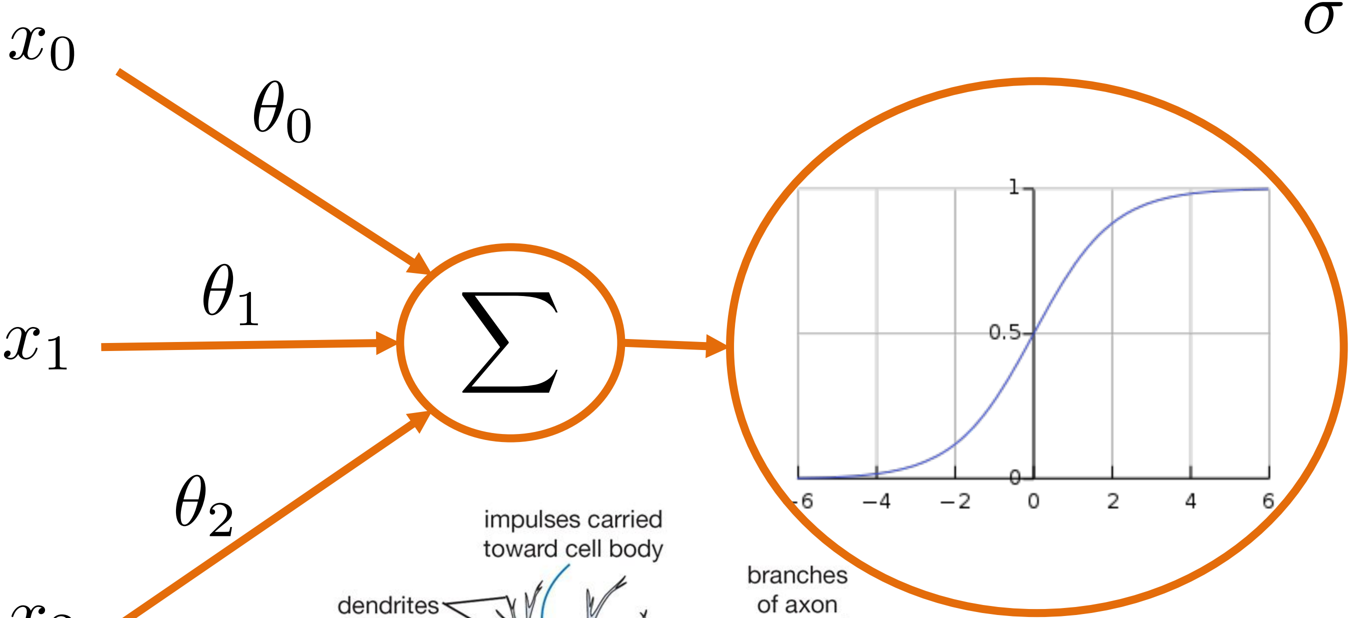


Loss (Softmax, Hinge)

Prediction

Sigmoid for binary predictions

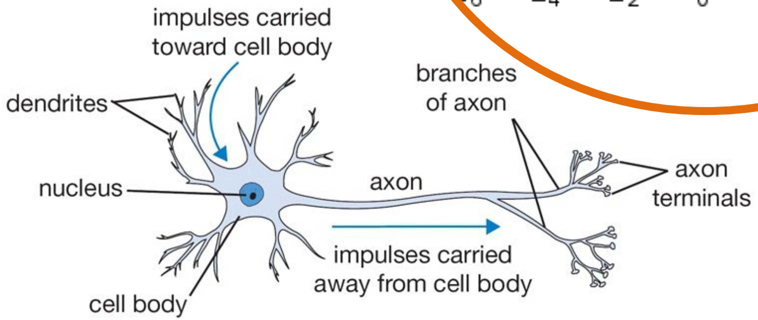
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



1

Can be interpreted as a probability

0



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

Logistic regression

- Probability of a binary output

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

Model for coins

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

Logistic regression

- Probability of a binary output

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

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

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

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

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

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

Logistic regression

- Probability of a binary output

$$\Pi_i = \frac{1}{1 + e^{-\mathbf{x}_i \boldsymbol{\theta}}}$$

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

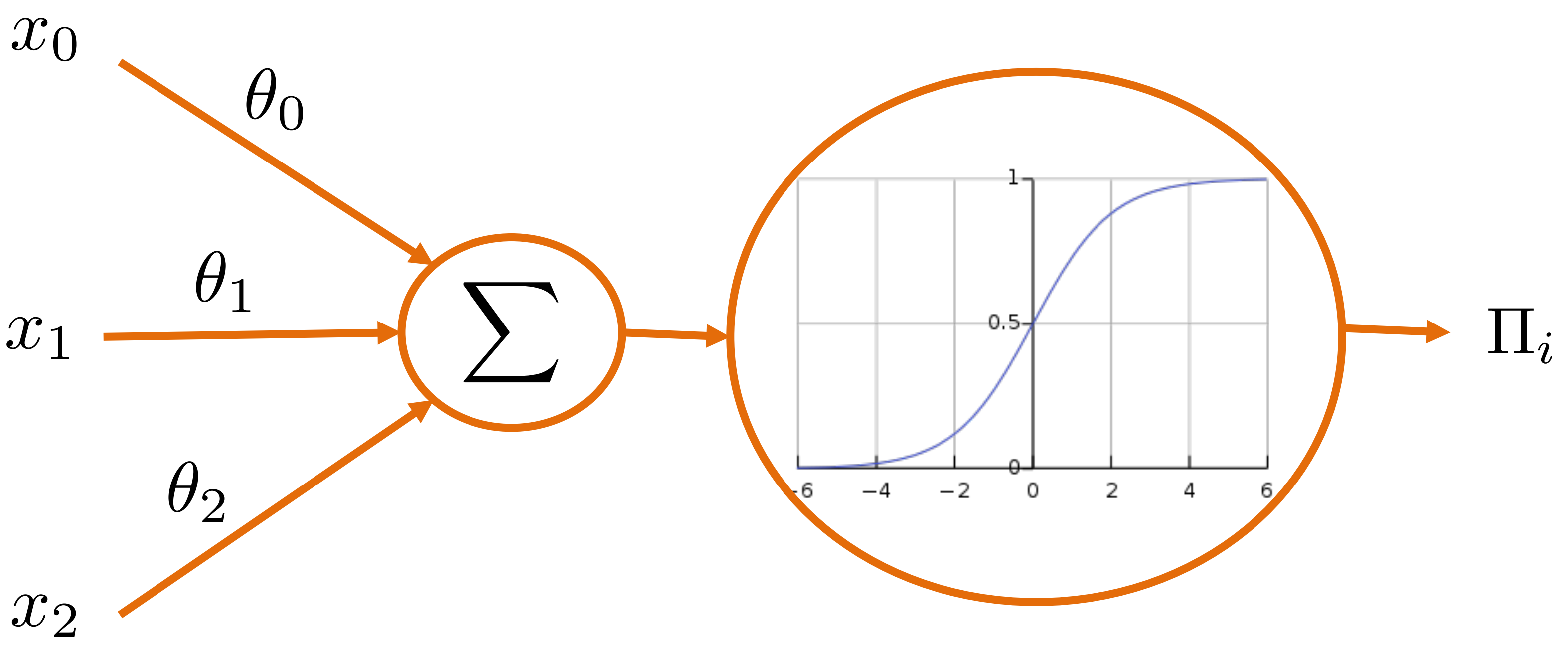
$$C(\boldsymbol{\theta}) = -\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})$$

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

Referred to as cross-entropy

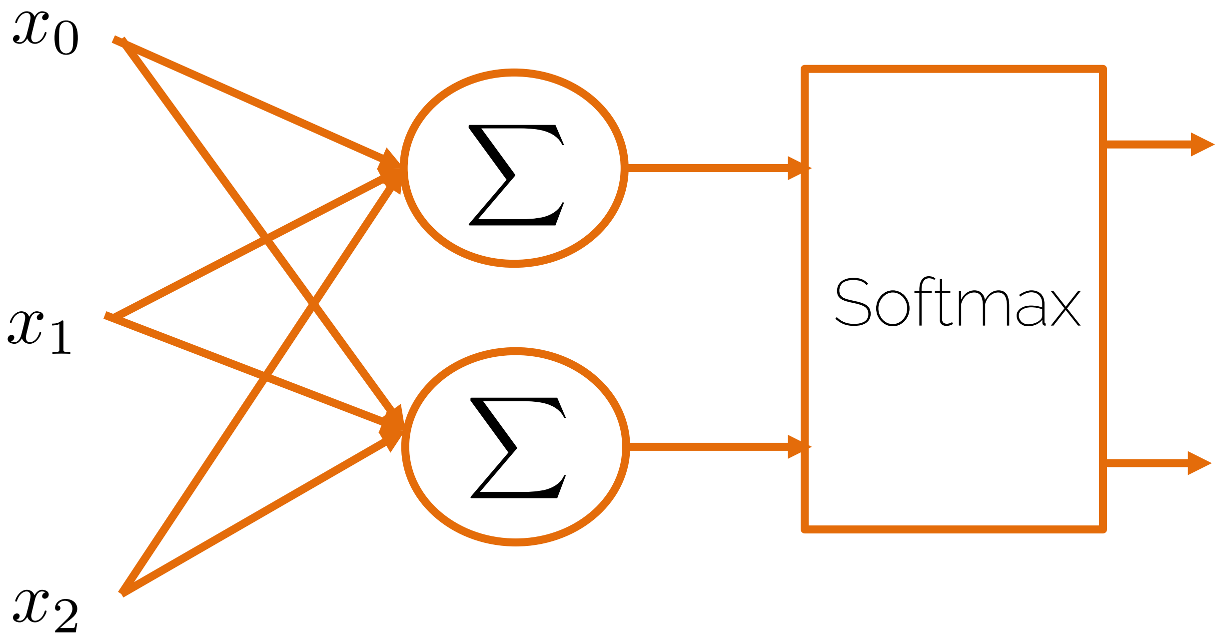
Softmax formulation

- What if we have multiple classes?



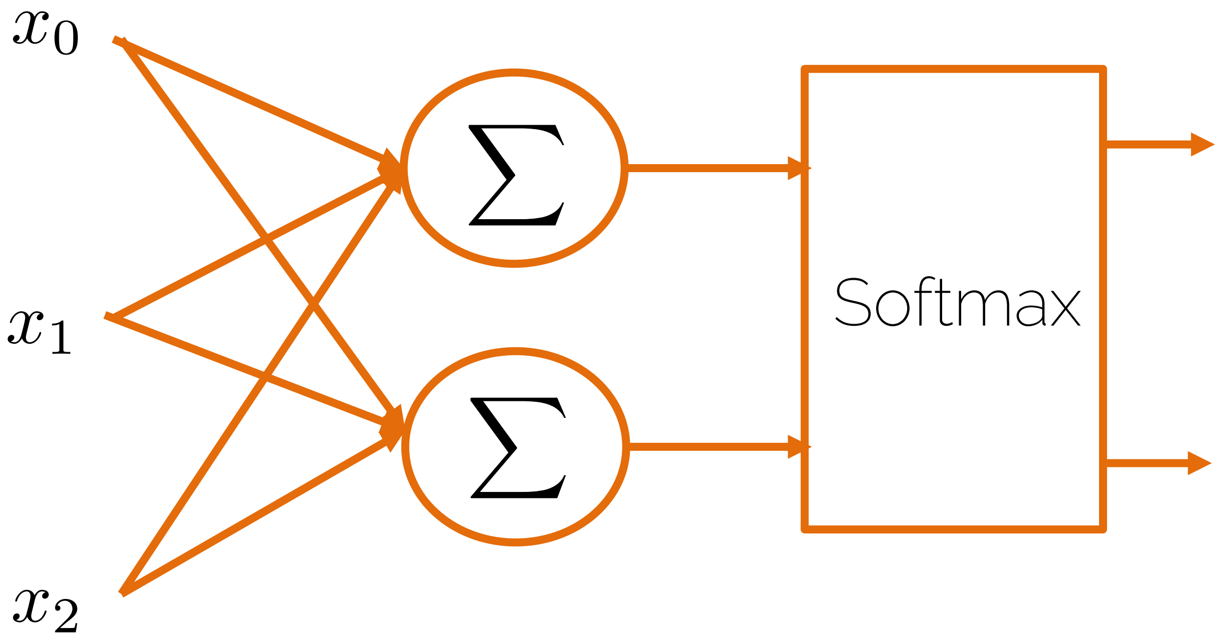
Softmax formulation

- What if we have multiple classes?



Softmax formulation

- What if we have multiple classes?



$$\Pi_1 = \frac{e^{\mathbf{x}_i \boldsymbol{\theta}_1}}{e^{\mathbf{x}_i \boldsymbol{\theta}_1} + e^{\mathbf{x}_i \boldsymbol{\theta}_2}}$$

$$\Pi_2 = \frac{e^{\mathbf{x}_i \boldsymbol{\theta}_2}}{e^{\mathbf{x}_i \boldsymbol{\theta}_1} + e^{\mathbf{x}_i \boldsymbol{\theta}_2}}$$

Softmax formulation

- Softmax

exp

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

normalize

- Softmax loss (ML)

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

Loss Functions

Naïve Losses

L2 Loss: $L^2 = \sum_{i=1}^{n}\left(y_i - f(x_i)\right)^2$

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

L1 Loss: $L^1 = \sum_{i=1}^{n}|y_i - f(x_i)|$

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

x_i:

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

y_i:

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

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

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

Cross-Entropy (Softmax)

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

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

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

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

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

3.2
5.1
-1.7

Loss

Cross-Entropy (Softmax)

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

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

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

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

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

	exp	
3.2	→	24.5
5.1	→	164.0
-1.7	→	0.18

Loss

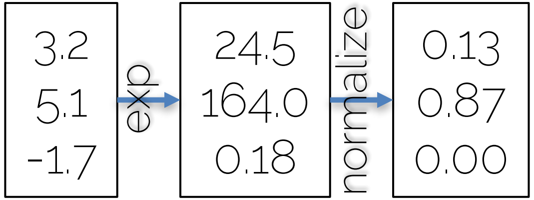
Cross-Entropy (Softmax)

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

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

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

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 $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

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

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

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

3.2 | 5.1 | -1.7 —exp→ 24.5 | 164.0 | 0.18 —normalize→ 0.13 | 0.87 | 0.00 —-log(x)→ 2.04 | 0.14 | 6.94

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

Cross-Entropy (Softmax)

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

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)

3.2 5.1 -1.7	exp →	24.5 164.0 0.18	normalize →	0.13 0.87 0.00	-log(x) →	2.04 0.14 6.94

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i =$$

$$= \frac{L_1 + L_2 + L_3}{3} =$$

$$= \frac{2.04 + 0.14 + 6.94}{3} =$$

$$= \mathbf{9.12}$$

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

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

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

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

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

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

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		

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

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

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	

$L_2 =$
$\max(0, 1.3 - 4.9 + 1) +$
$\max(0, 2.0 - 4.9 + 1) =$
$= \max(0, -2.6) + \max(0, -1.9) =$
$= 0 + 0 =$
$= \mathbf{0}$

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

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

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

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):

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i = $$

$$= \frac{L_1 + L_2 + L_3}{3} =$$

$$= \frac{2.9 + 0 + 10.9}{3} =$$

$$= \mathbf{4.6}$$

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

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

L^2-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 $L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max\big(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1\big) + \lambda R(W)$

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

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

Example:

$x = [1,1,1,1]$

$w_1 = [1, 0, 0, 0]$

$w_2 = [0.25, 0.25, 0.25, 0.25]$

$w_1^T x = w_2^T x = 1$

$R^2(w_1) = 1$

$R^2(w_2) = 0.25^2 + 0.25^2 + 0.25^2 + 0.25^2 = 0.25$

Hinge Loss vs Softmax

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

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

Hinge Loss vs Softmax

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

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

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: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Given the following scores:

Scores	Hinge loss:
$s = [5, -3, 2]$	$\max(0, -3 - 5 + 1) +$ $\max(0, 2 - 5 + 1) = 0$
$s = [5, 10, 10]$	$\max(0, 10 - 5 + 1) +$ $\max(0, 10 - 5 + 1) = 12$
$s = [5, -20, -20]$	$\max(0, -20 - 5 + 1) +$ $\max(0, -20 - 5 + 1) = 0$

$y_i = 0$

Hinge Loss vs Softmax

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

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

Given the following scores:	Hinge loss:	Softmax loss:
$s = [5, -3, 2]$	$\max(0, -3-5+1) + \max(0, 2-5+1) = 0$	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)))=2.e-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: $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

Given the following scores:

	Hinge loss:	Softmax loss:
$s = [5, -3, 2]$	$\max(0, -3 - 5 + 1) + \max(0, 2 - 5 + 1) = 0$	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)))=2.e-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., $f(x_i, W) = W \cdot [x_0, x_1, \ldots, x_N]^T$

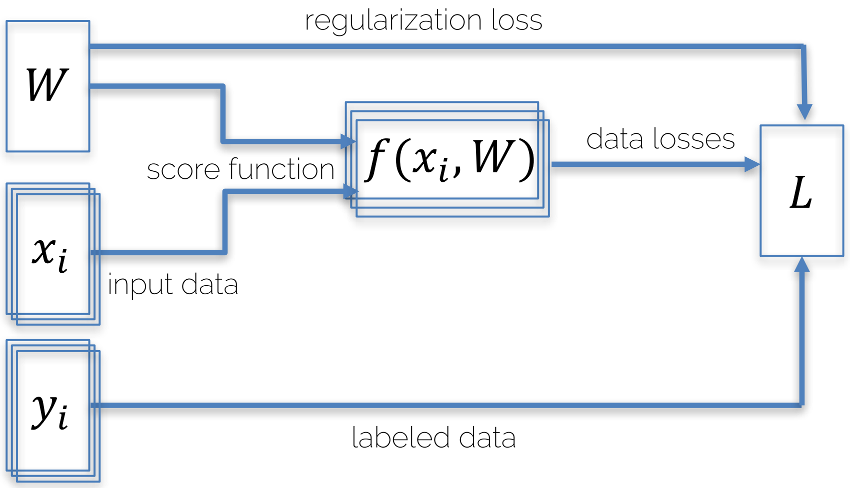
Given a function with weights W,
Training pairs $[x_i; y_i]$ (input and labels)

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

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 $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

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 $L_i = -\log(\frac{e^{s_{y_i}}}{\sum_i e^{s_j}})$

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

⇨

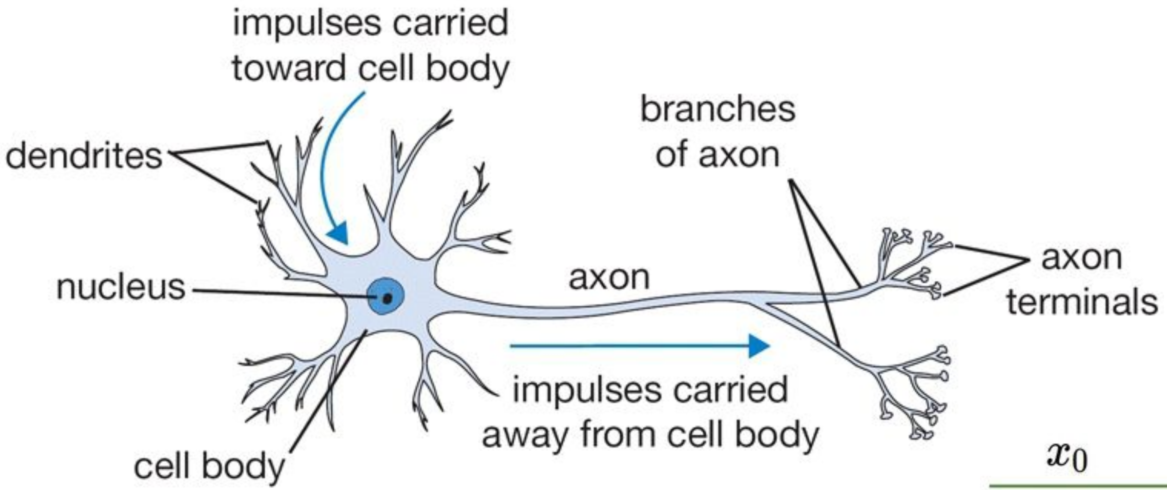
Want to find optimal $\boldsymbol{W}$. I.e., weights are unknowns of optimization problem

Compute gradient w.r.t. $\boldsymbol{W}$.

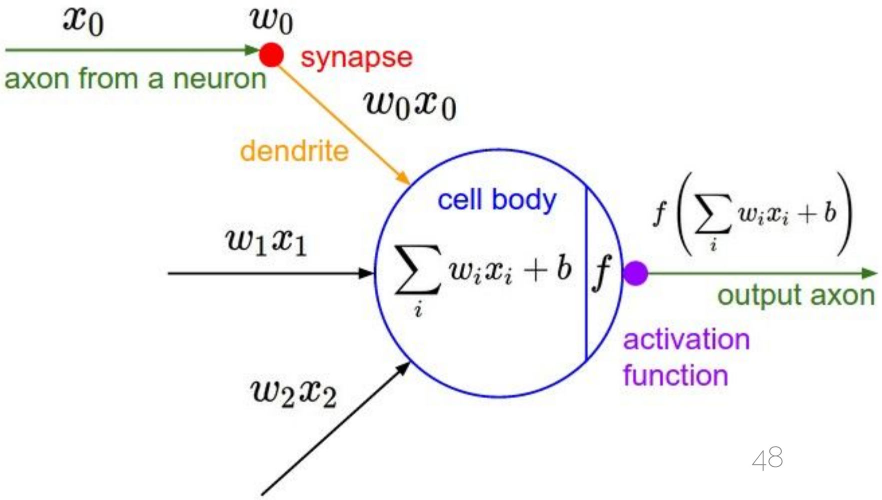
Gradient $\nabla_W L$ is computed via backpropagation

Activation functions

Neurons

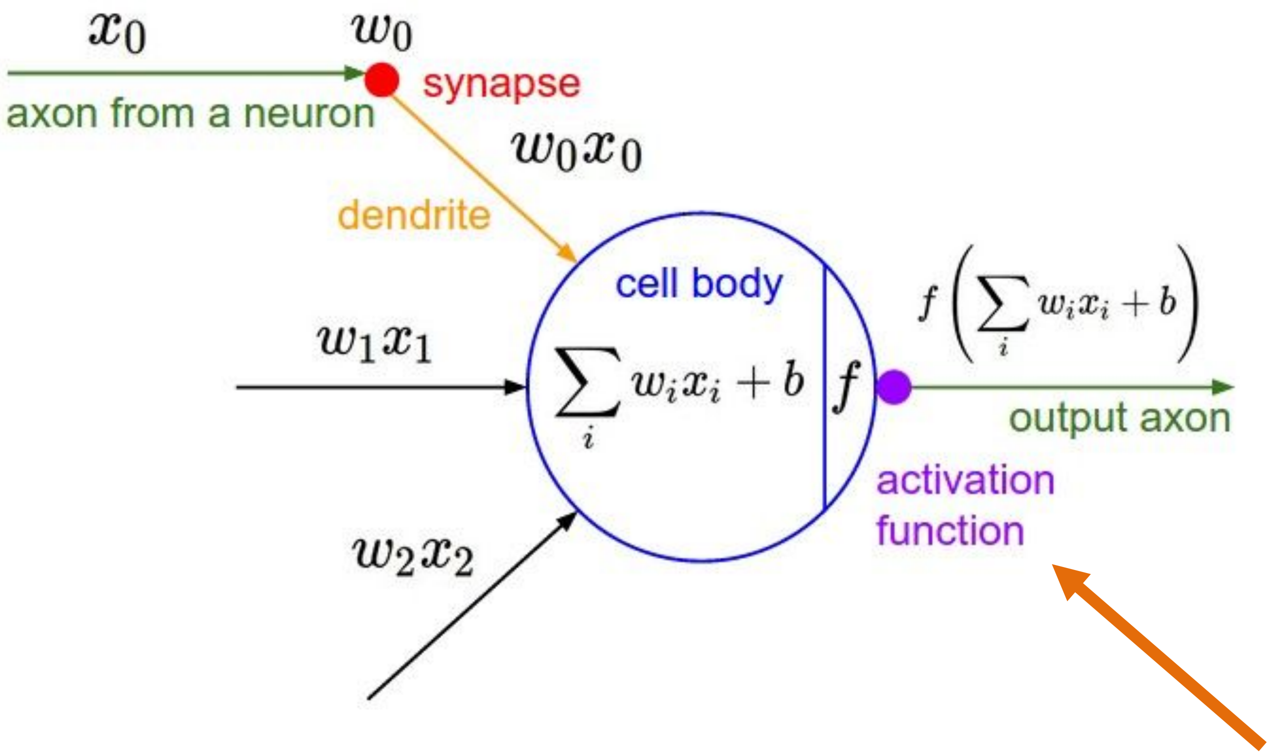


impulses carried
toward cell body
dendrites
branches
of axon
nucleus
axon
axon
terminals
impulses carried
away from cell body
cell body



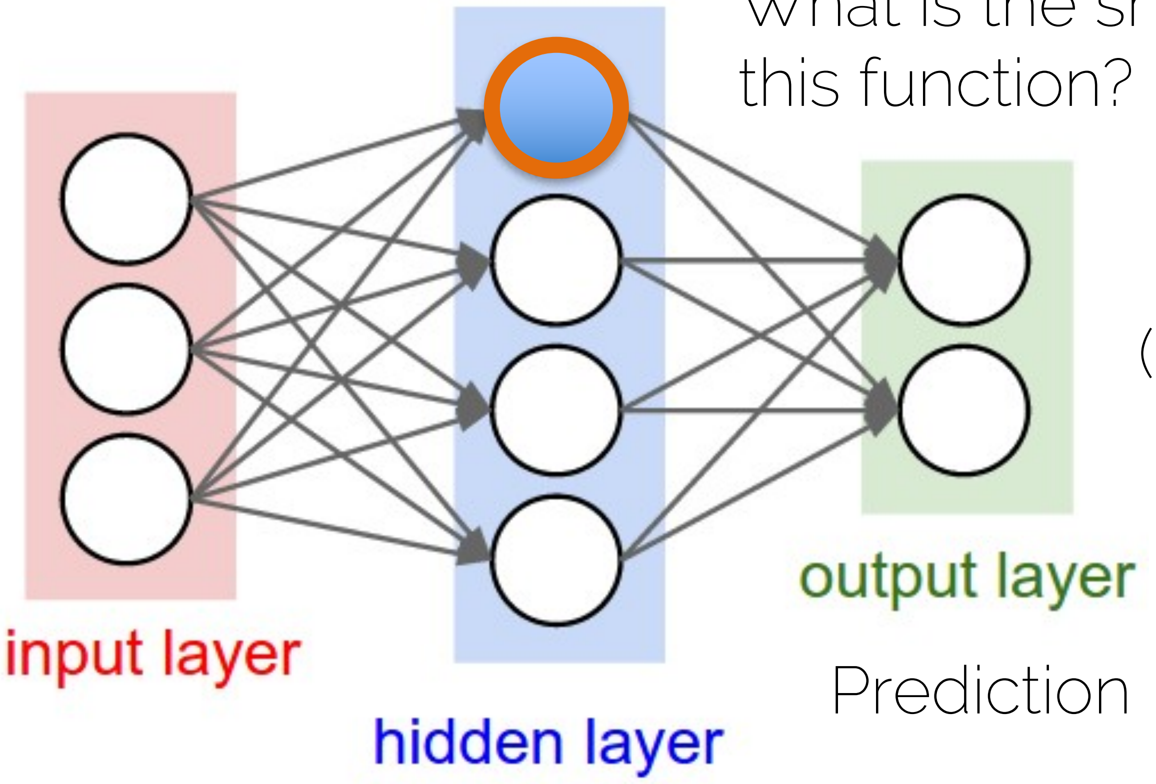
x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation
function
$w_2 x_2$

Neurons



x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation function
$w_2 x_2$

Neural networks

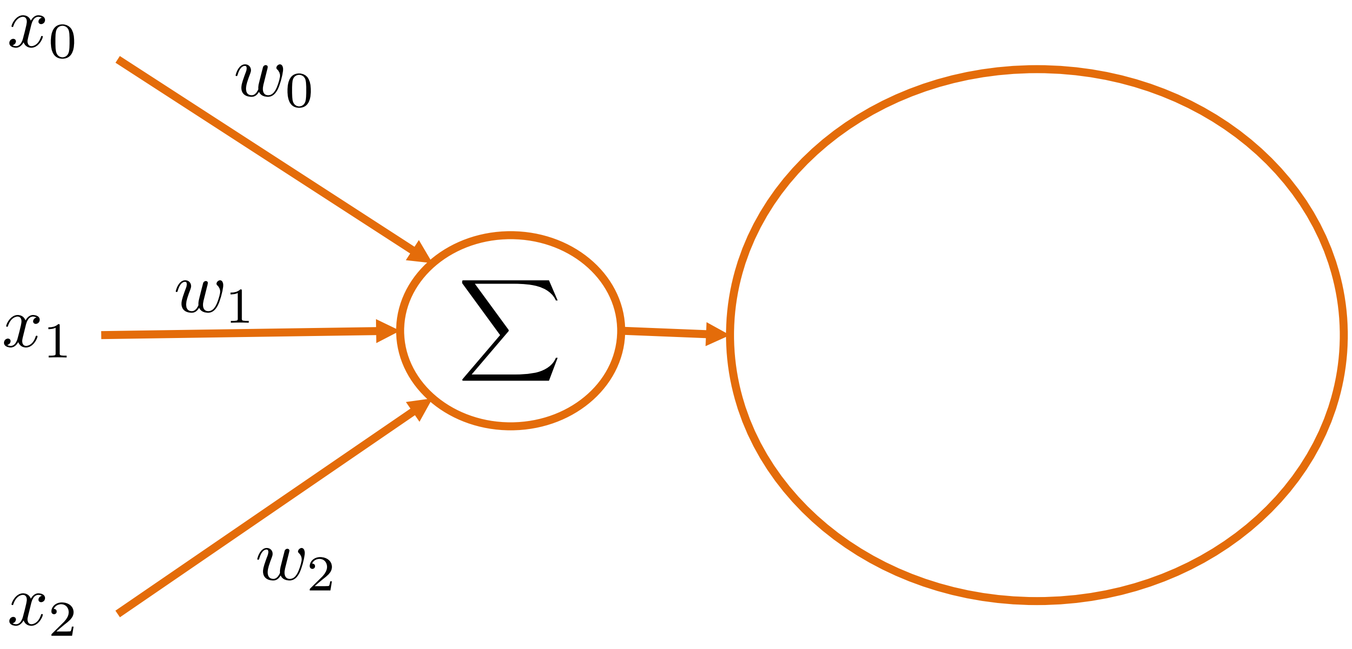


What is the shape of this function?

Loss (Softmax, Hinge)

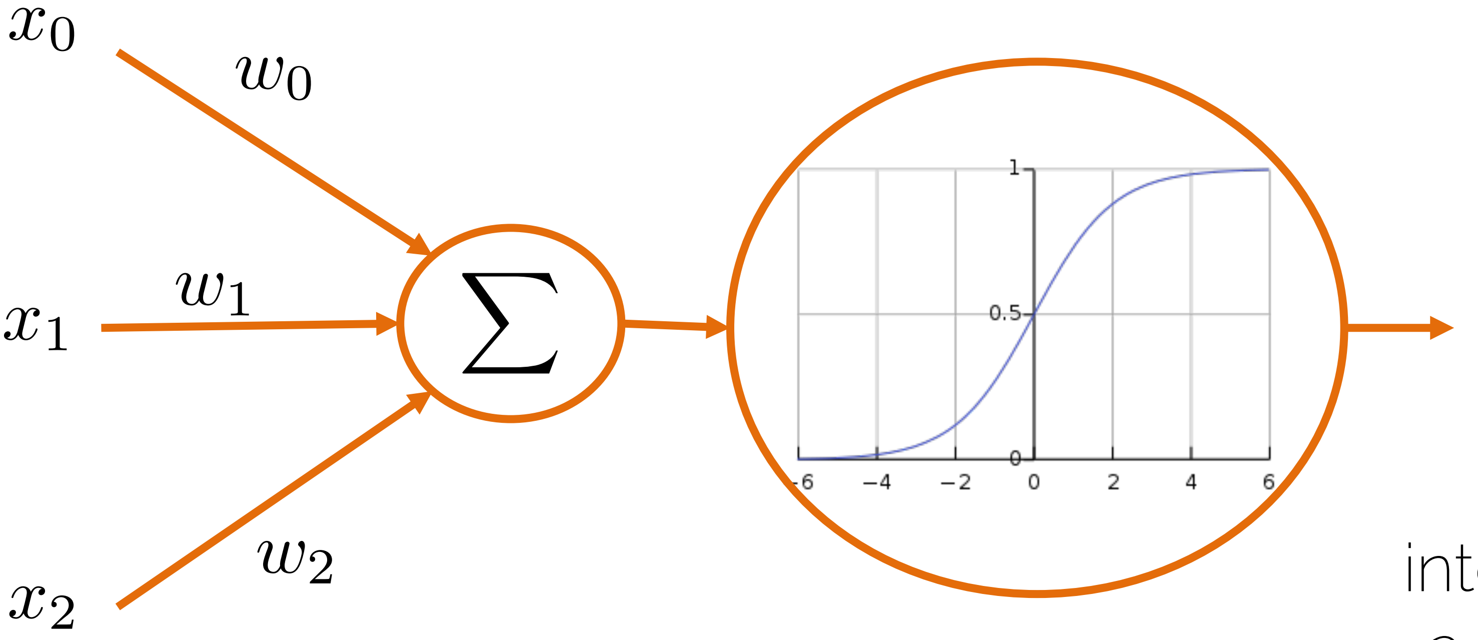
Prediction

Activation functions or hidden units



Sigmoid

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



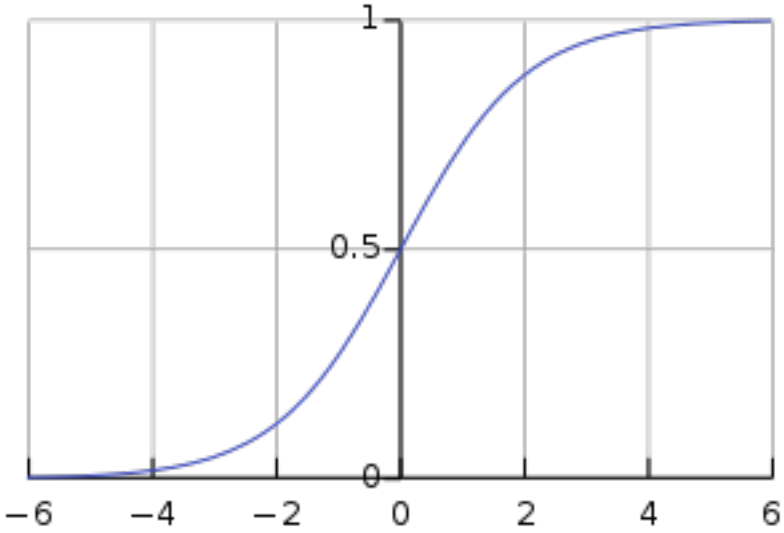
$y_i \in \{0, 1\}$

Can be interpreted as a probability

Sigmoid

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

Forward

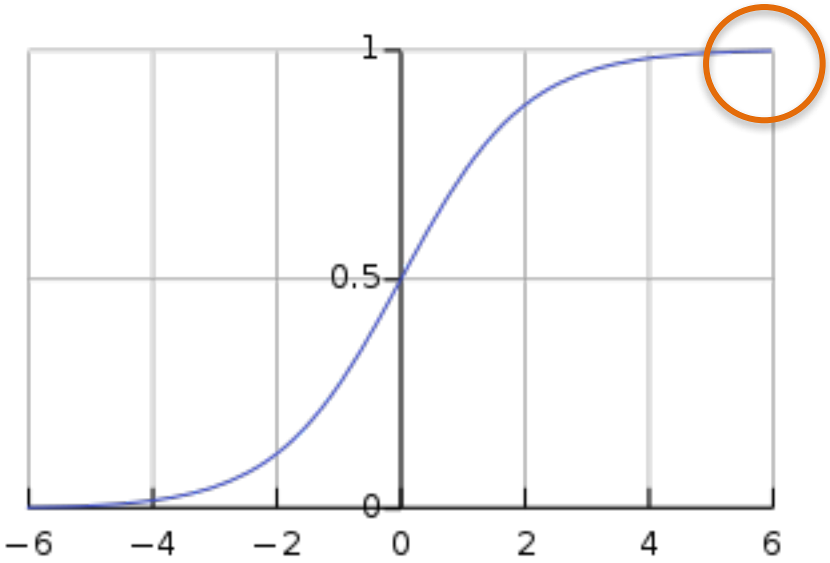


$$\frac{\partial L}{\partial x} = \frac{\partial \sigma}{\partial x}\frac{\partial L}{\partial \sigma} \quad \longleftarrow \quad \frac{\partial \sigma}{\partial x} \quad \longleftarrow \quad \frac{\partial L}{\partial \sigma}$$

Sigmoid

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

Forward



$$x = 6$$

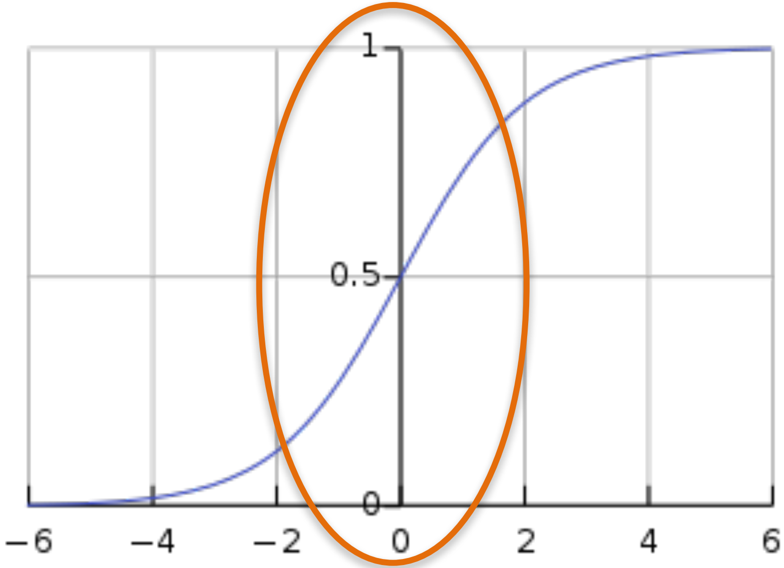
✘ Saturated neurons kill the gradient flow

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

Sigmoid

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

Forward

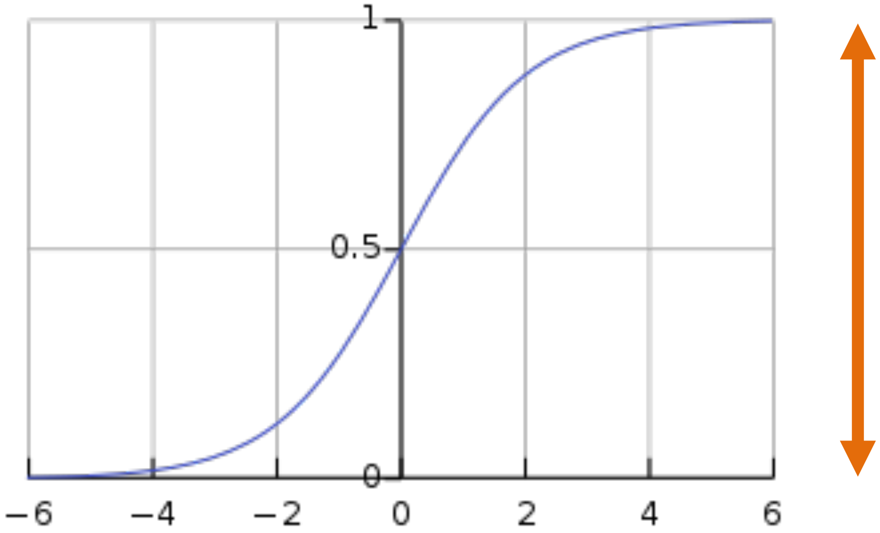


Active region for gradient descent

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

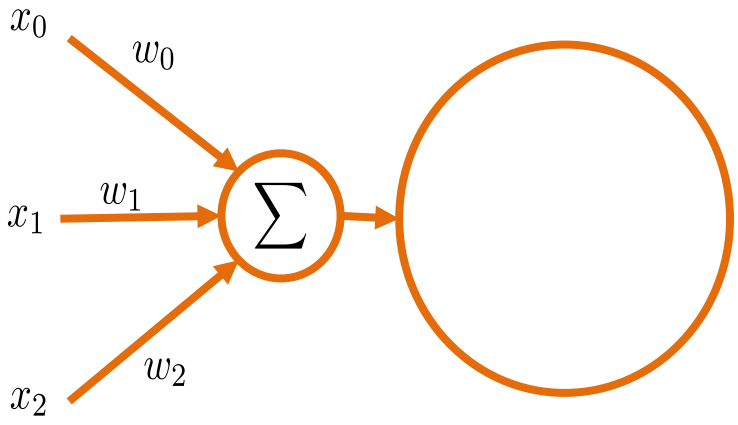
Sigmoid

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



Output is always positive

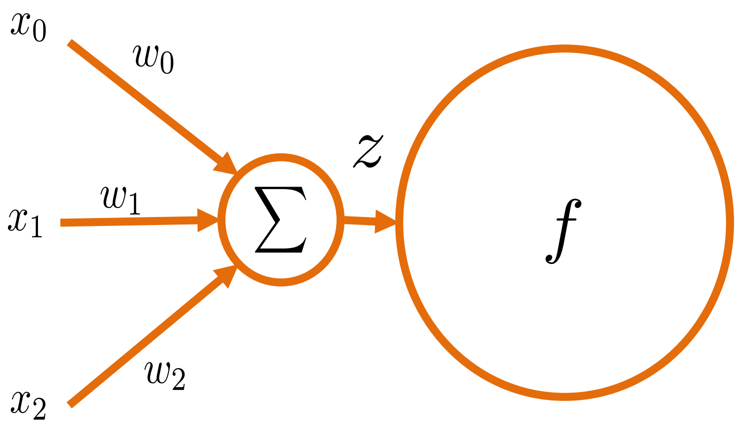
Problem of positive output



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

We want to compute the gradient wrt the weights

Problem of positive output

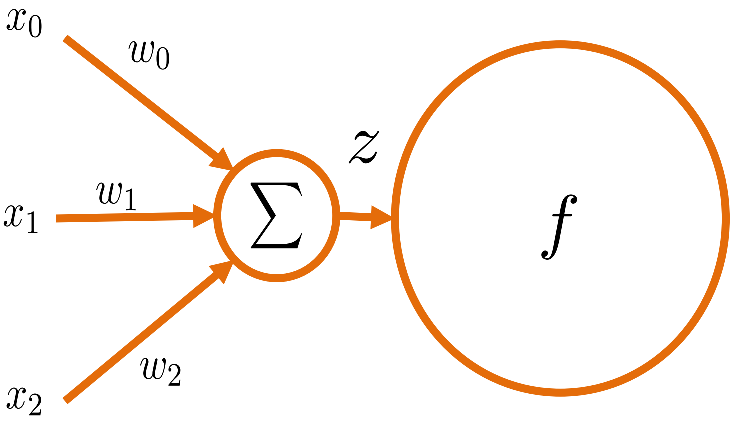


$$f\left(\sum_i w_i x_i + b\right)$$

$$\frac{\partial z}{\partial w} = x_i > 0$$

We want to compute the gradient wrt the weights

Problem of positive output



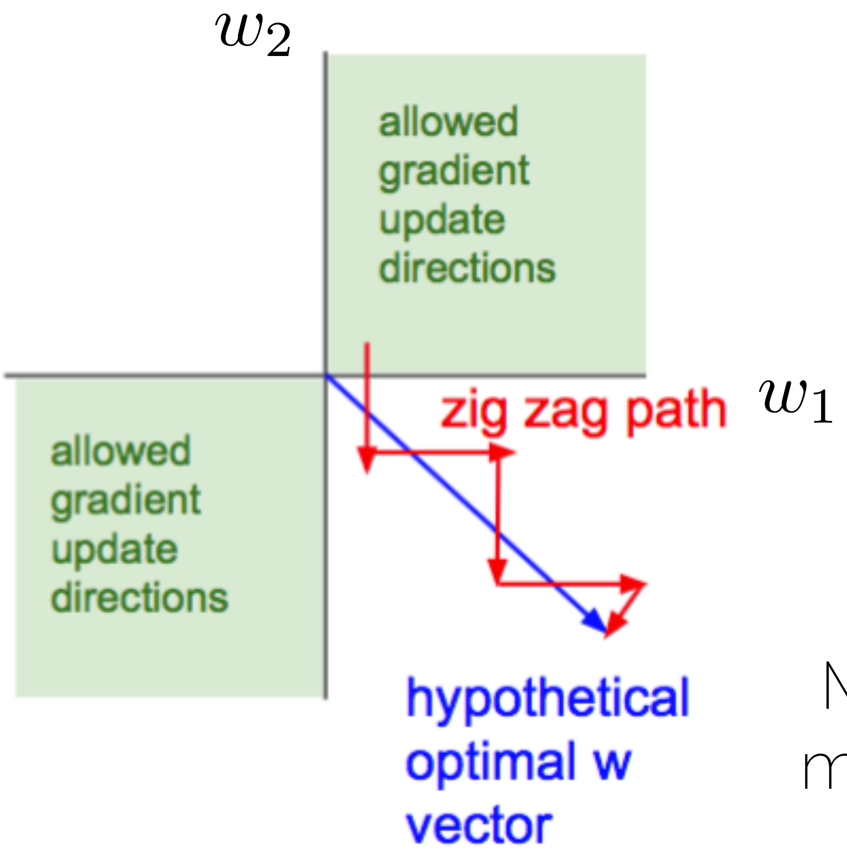
$$f\left(\sum_i w_i x_i + b\right)$$

$$\frac{\partial f}{\partial z}$$

$$\frac{\partial z}{\partial w} = x_i > 0$$

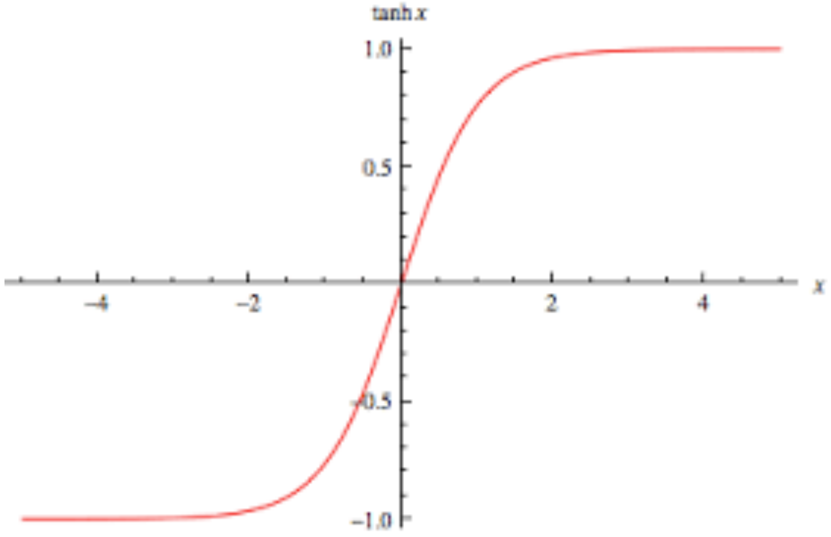
It is going to be either positive or negative for all weights

Problem of positive output



More on zero-mean data later

tanh



✗ Still saturates

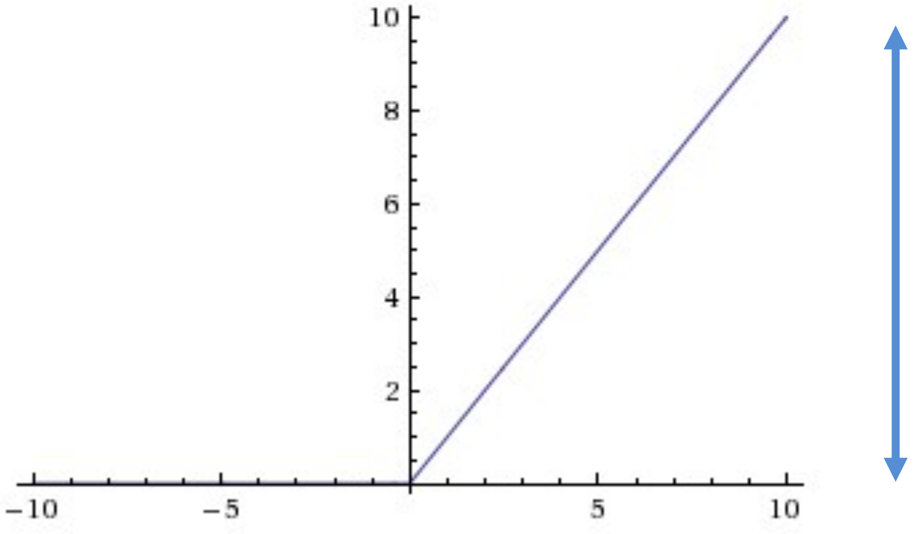
✓ Zero-centered

✗ Still saturates

LeCun 1991

Rectified Linear Units (ReLU)

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



Large and consistent gradients ✓

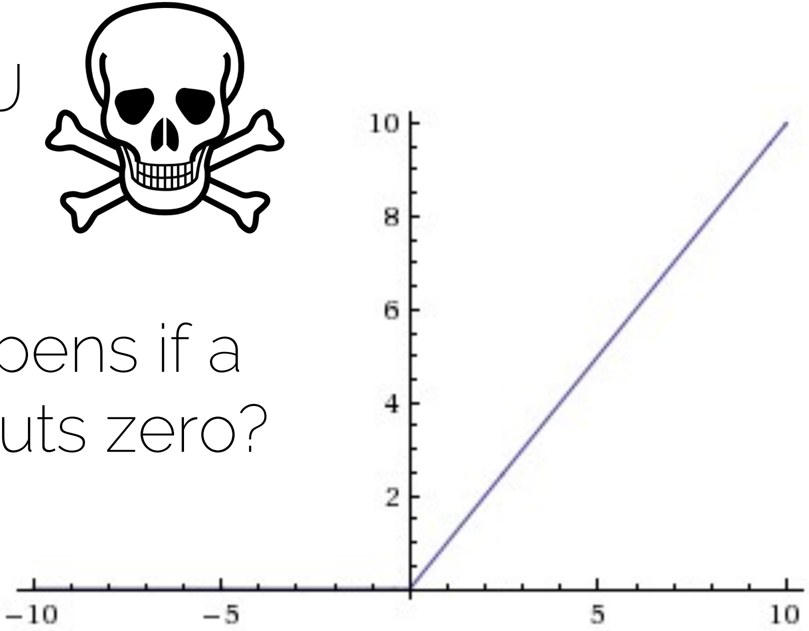
✓ Fast convergence

✓ Does not saturate

Krizhevsky 2012

Rectified Linear Units (ReLU)

✗ Dead ReLU



What happens if a ReLU outputs zero?

Large and consistent gradients ✓

✓ Fast convergence

✓ Does not saturate

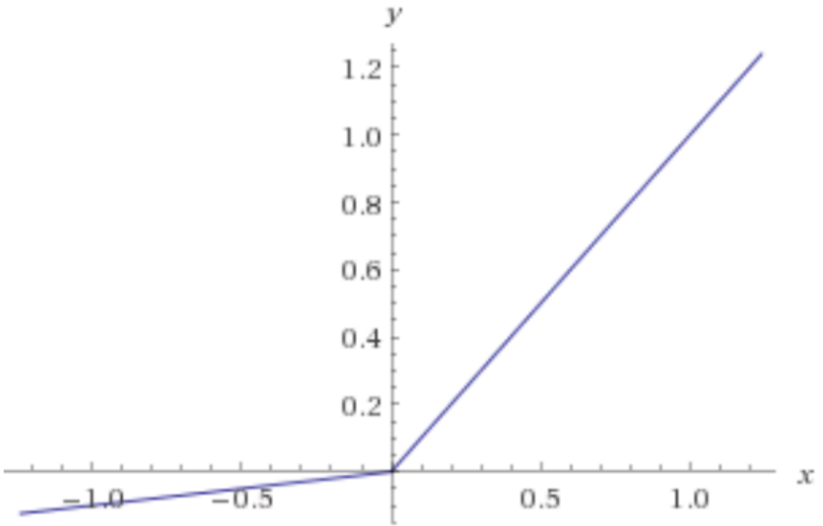
Rectified Linear Units (ReLU)

- 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

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



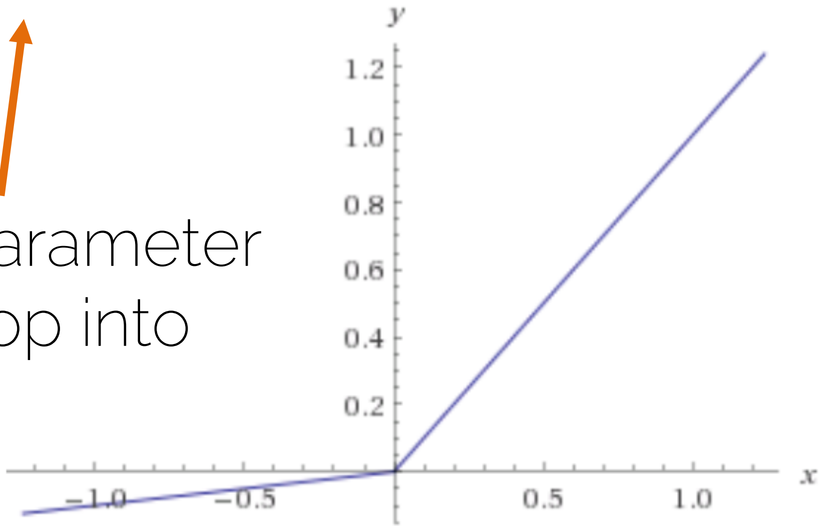
✓ Does not die

Mass 2013

Parametric ReLU

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

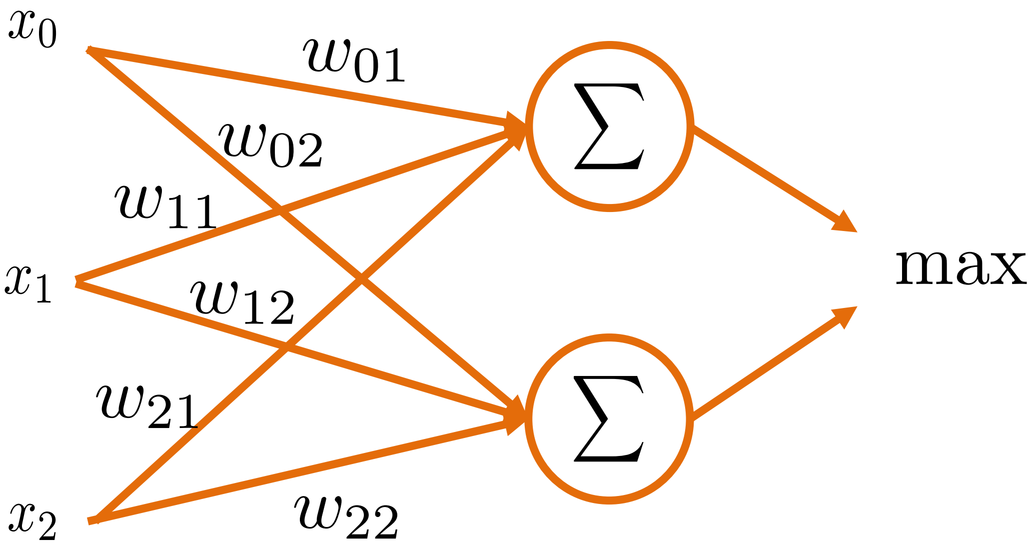
One more parameter
to backprop into



✓ Does not die

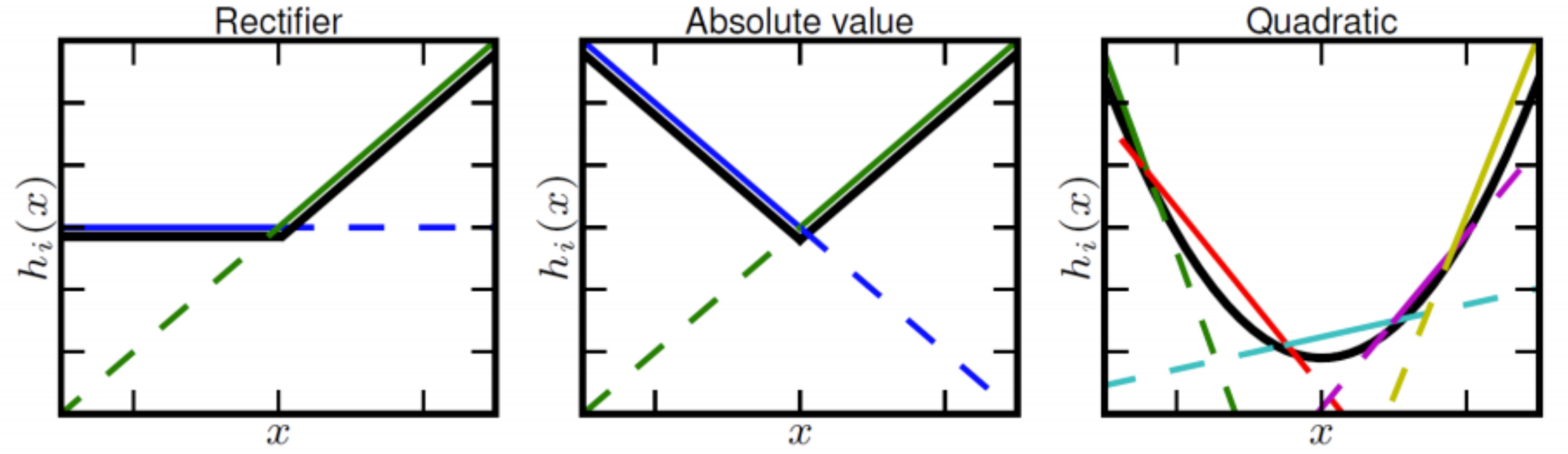
He 2015

Maxout units



Goodfellow 2013

Maxout units

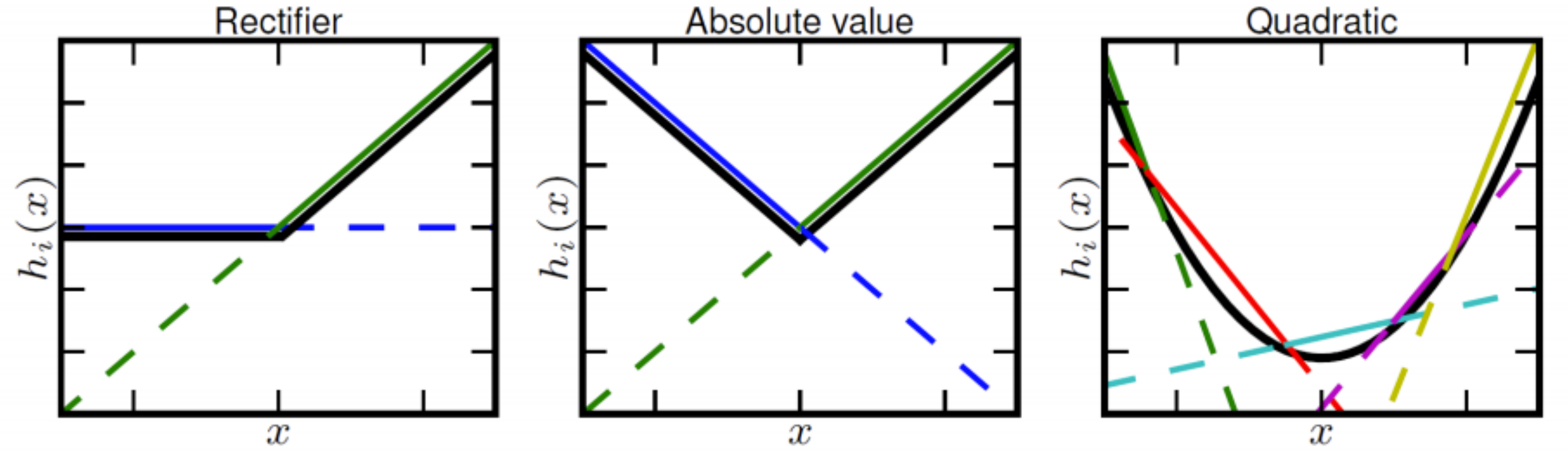


k=2 **k=2** **k=5**

Piecewise linear approximation of
a convex function with N pieces

Goodfellow 2013

Maxout units



k=2 | k=2 | k=5

✓ Generalization of ReLUs ✓ Linear regimes ✓ Does not die ✓ Does not saturate

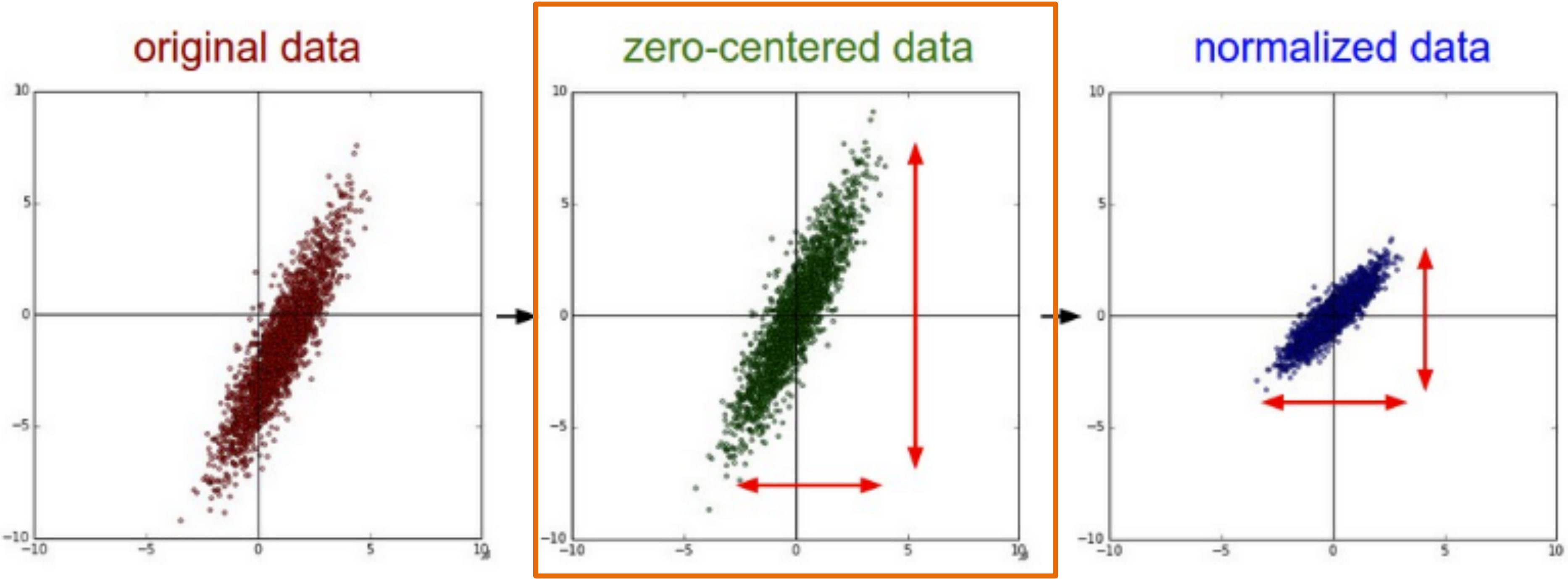
✗ Increase of the number of parameters

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

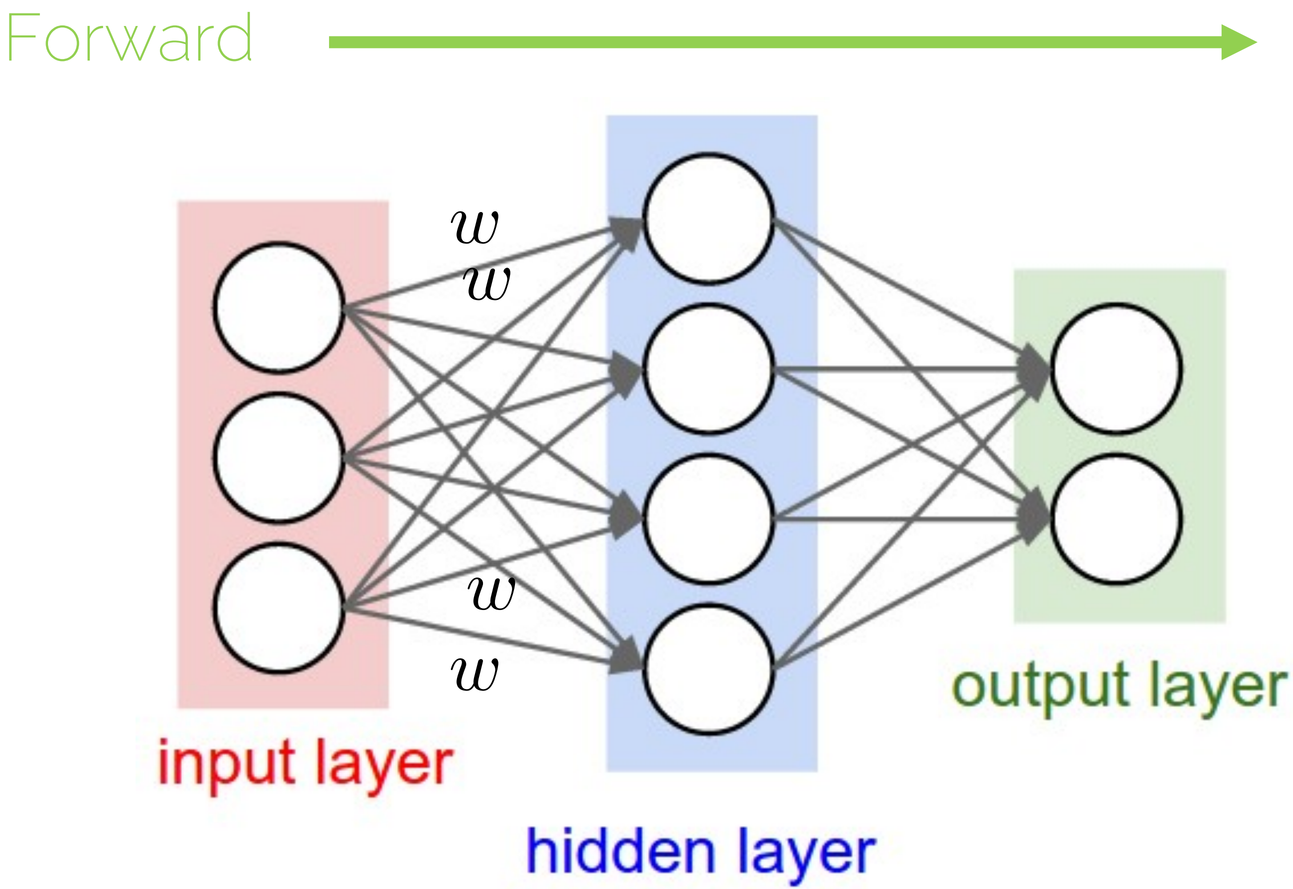
Data pre-processing



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

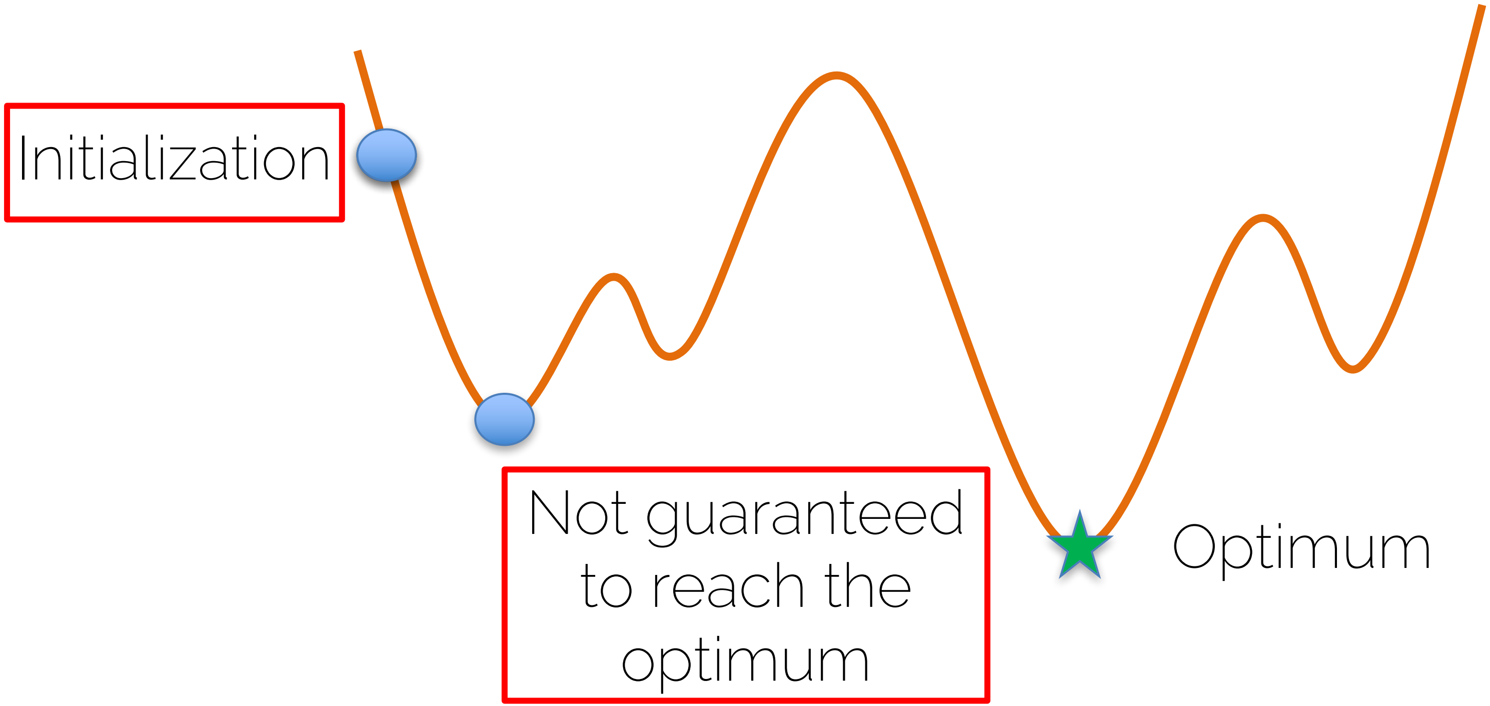
Weight initialization

How do I start?



Initialization is extremely important

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

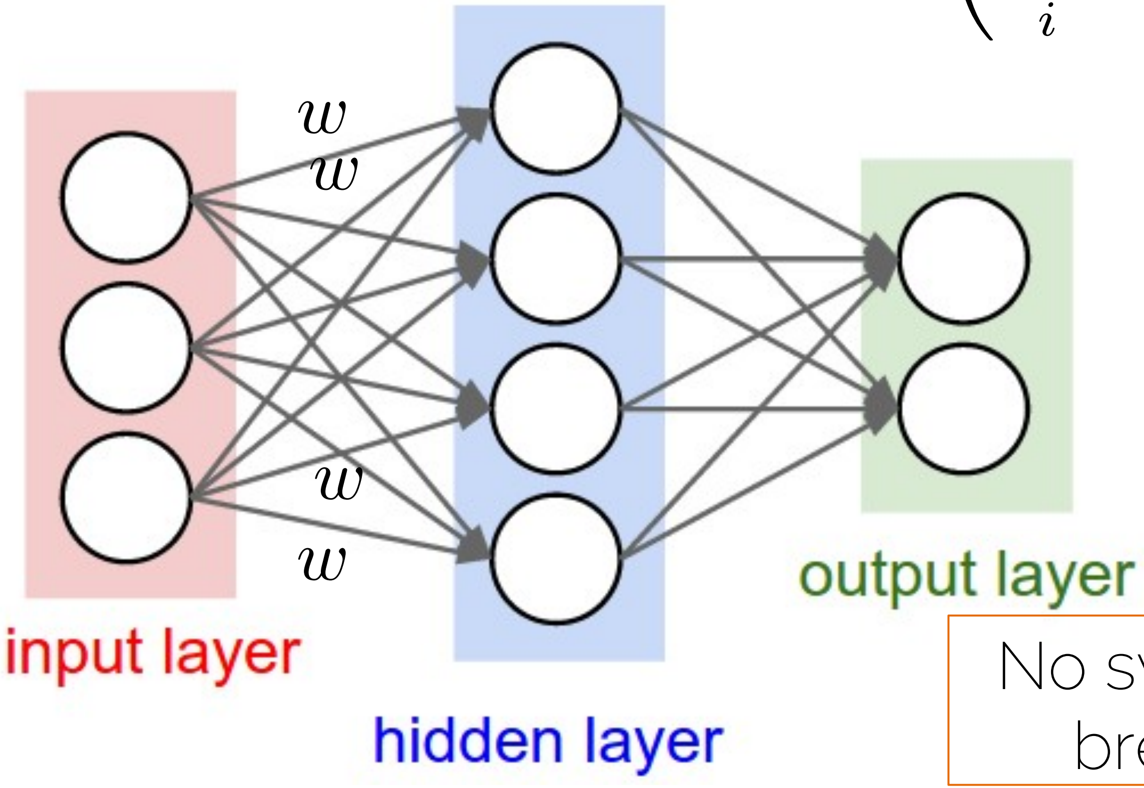


How do I start?

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

$w = 0$

What happens to the gradients?



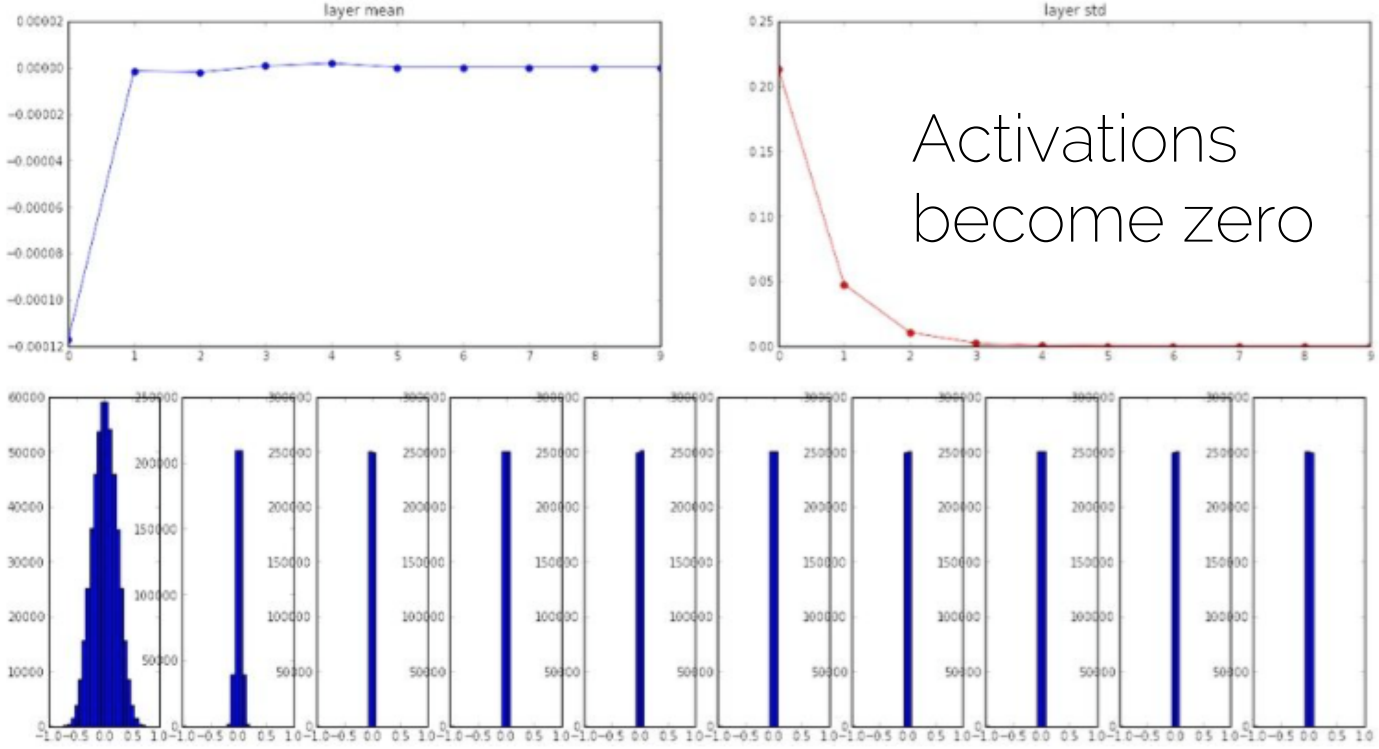
No symmetry breaking

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



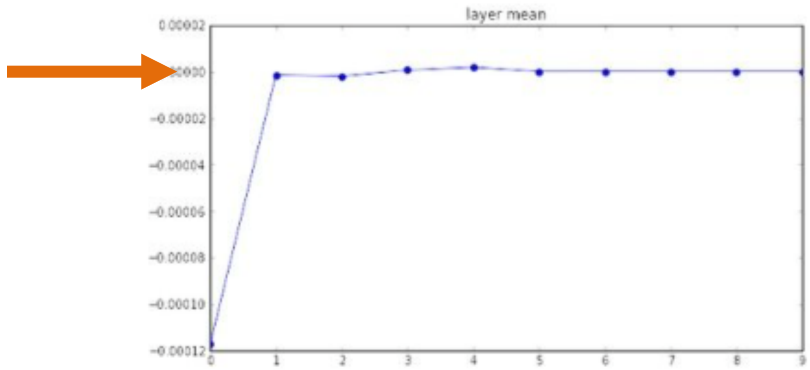
Activations become zero

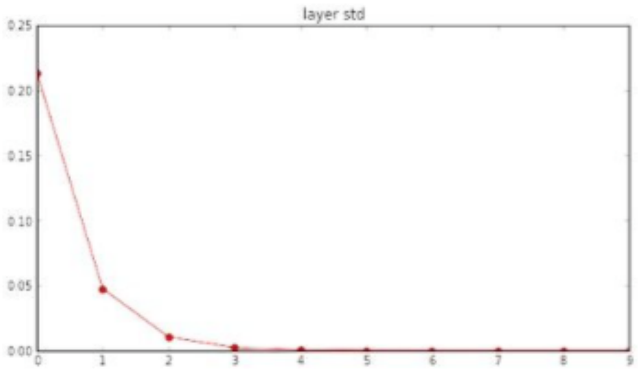
Input

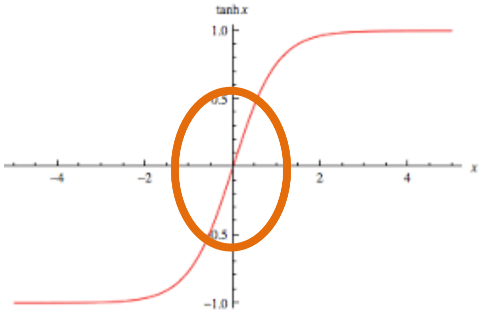
Last layer

Forward

Small random numbers





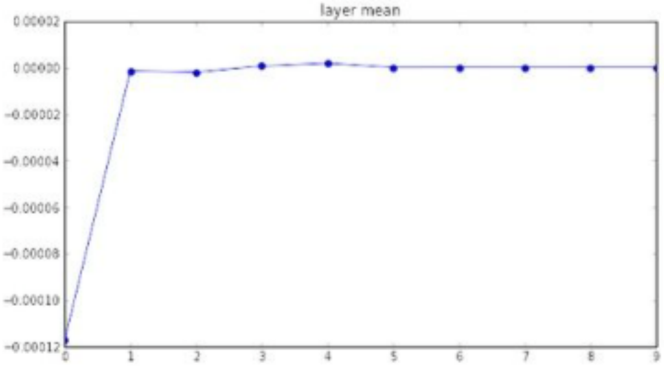


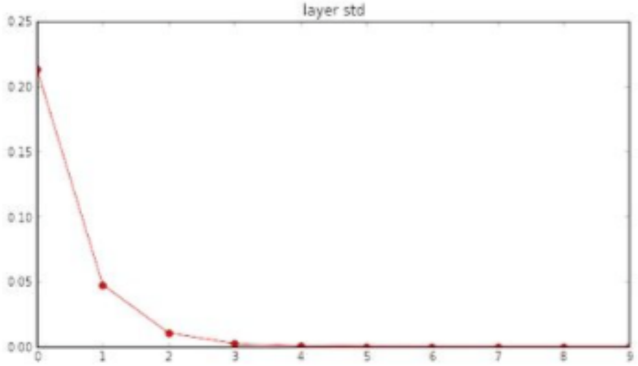
small

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

Forward

Small random numbers





Gradients vanish

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

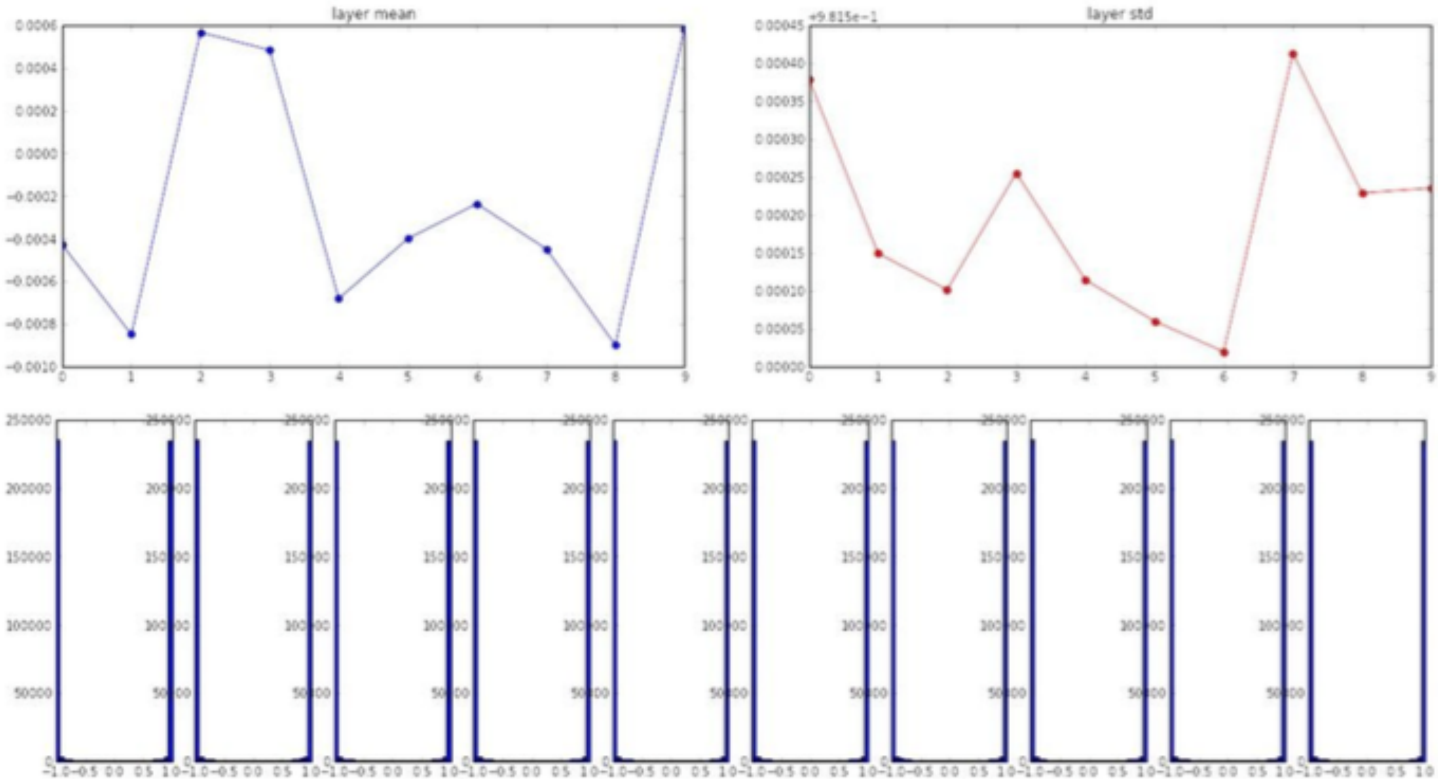
Backward

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