CS 4803 / 7643: Deep Learning

Topics:
- Regularization
- Neural Networks

Dhruv Batra

Georgia Tech

Administrativia

- PS1/HW1 out
  - Due: 09/09, 11:59pm
  - Asks about topics being covered now
  - Caveat: one more (extra credit) problem to be added
  - Please please please please start early
  - More details next class

Recap from last time

Parametric Approach: Linear Classifier

3072x1

$$f(x,W) = Wx + b \quad 10\text{x}1$$

S

10x1 10x3072

Image

f(**x**,**W**) → **10** numbers giving class scores

Array of **32x32x3** numbers
(3072 numbers total)

x

W
parameters
or weights

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

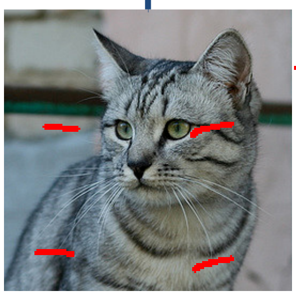
Stretch pixels into column

Input image

56
231
24
2

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Stretch pixels into column



Input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W}\ \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \begin{array}{l} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{array}$$

3x4

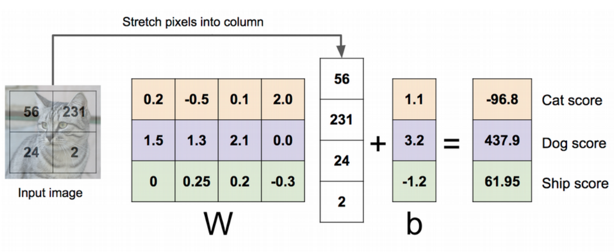
$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{x_i} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} \longleftrightarrow \boxed{\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 & 1.1 \\ 1.5 & 1.3 & 2.1 & 0.0 & 3.2 \\ 0 & 0.25 & 0.2 & -0.3 & -1.2 \end{bmatrix}}_{W \qquad b}}_{\text{new, single W}} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \\ 1 \end{bmatrix}}_{x_i}$$

Image Credit: Andrej Karpathy, CS231n

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

f(x,W) = Wx +b

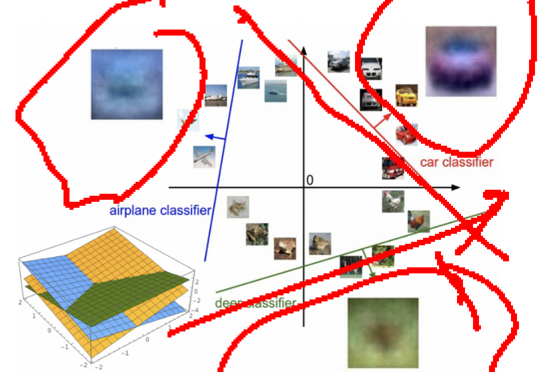


Visual Viewpoint

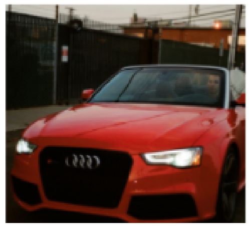
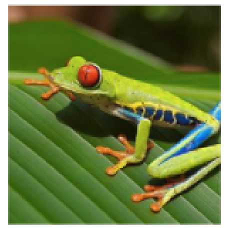
One template per class

Geometric Viewpoint

Hyperplanes cutting up space



Recall from last time: Linear Classifier

airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.

2. Come up with a way of efficiently finding the parameters that minimize the loss function. **(optimization)**

Softmax vs. SVM

$P(y_i | x_i, \vec{w})$

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

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

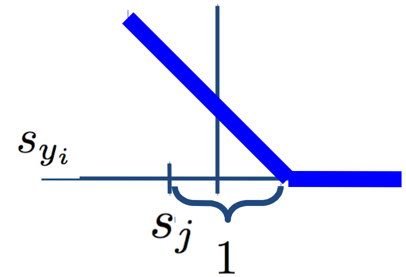
Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Multiclass SVM loss:

"Hinge loss"



$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

Softmax vs. SVM

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

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

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

cat	**3.2**
car	5.1
frog	-1.7

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ Softmax Function

Probabilities must be >= 0

cat	**3.2**		**24.5**
car	5.1	$\xrightarrow{\text{exp}}$	164.0
frog	-1.7		0.18

unnormalized probabilities

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

		exp		normalize	
cat	**3.2**	→	**24.5**	→	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

unnormalized probabilities

probabilities

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

		exp		normalize	
cat	**3.2**	→	**24.5**	→	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00
	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

→ $L_i = -\log(0.13)$
$= \mathbf{2.04}$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits	$\xrightarrow{\text{exp}}$	unnormalized probabilities	$\xrightarrow{\text{normalize}}$	probabilities
cat	**3.2**		**24.5**		**0.13**
car	5.1		164.0		0.87
frog	-1.7		0.18		0.00

$\rightarrow$ $L_i = -\log(0.13)$

$= \mathbf{2.04}$

Maximum Likelihood Estimation
Choose probabilities to maximize the likelihood of the observed data

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits	exp →	unnormalized probabilities	normalize →	probabilities	→ compare ←	Correct probs
cat	**3.2**		**24.5**		**0.13**		**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

min KL

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities		Correct probs
cat	3.2	exp →	24.5	normalize →	0.13	→ compare ←	1.00
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Kullback–Leibler divergence

$$D_{KL}(P\|Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities		Correct probs
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**	→ compare ←	**1.00**
car	5.1		164.0		0.87		0.00
frog	-1.7		0.18		0.00		0.00

Cross Entropy

$$H(P, Q) = H(p) + D_{KL}(P \| Q)$$

Log-Likelihood / KL-Divergence / Cross-Entropy

$D = \{(x_i, y_i)\}$ IID $\sim P^*$

$$\boxed{\hat{w}_{MLE}} = \max P(D \mid w)$$

$$\equiv \max \log P(D \mid w)$$

$$= \max \boxed{\sum_i} \log P(y_i \mid x_i, w)$$

Plan for Today

- Regularization
- Neural Networks

Regularization

$$\underbrace{L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}$$

Data loss: Model predictions should match training data

Regularization

$-\log P(D|\vec{w})$

"model fit"

$-\log P(\vec{w})$

objective

$$\boxed{L(W)} = \underbrace{\boxed{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}} + \underbrace{\boxed{\lambda R(W)}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Regularization

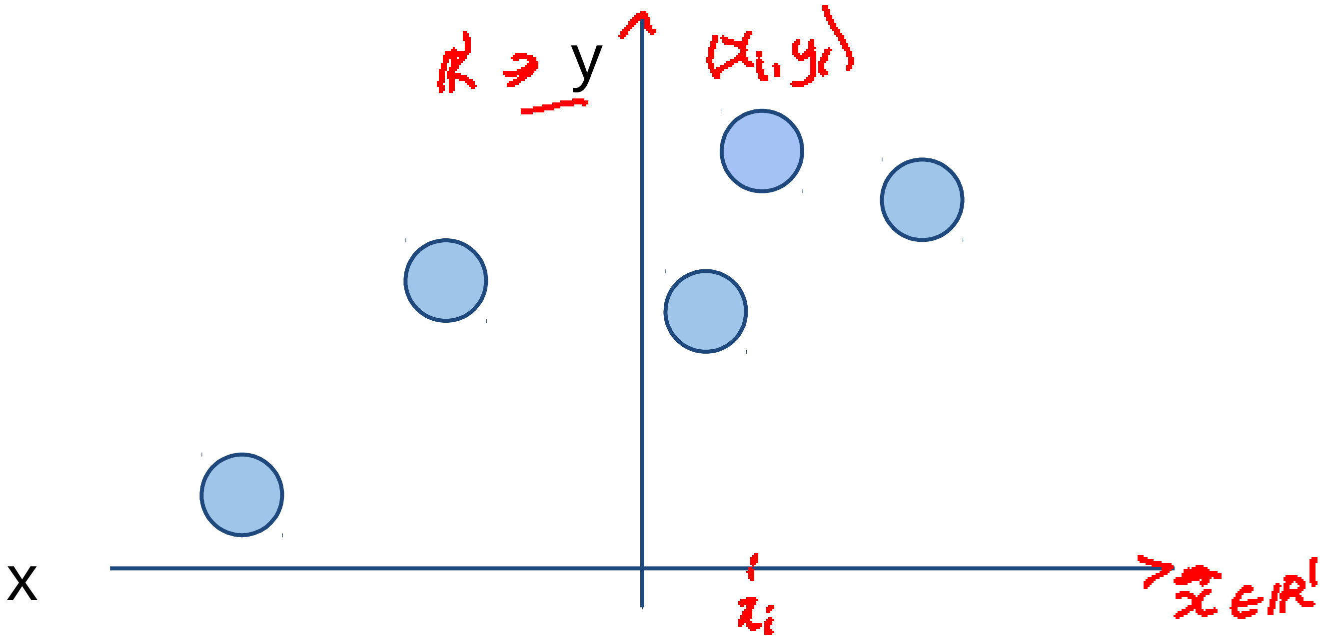
λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

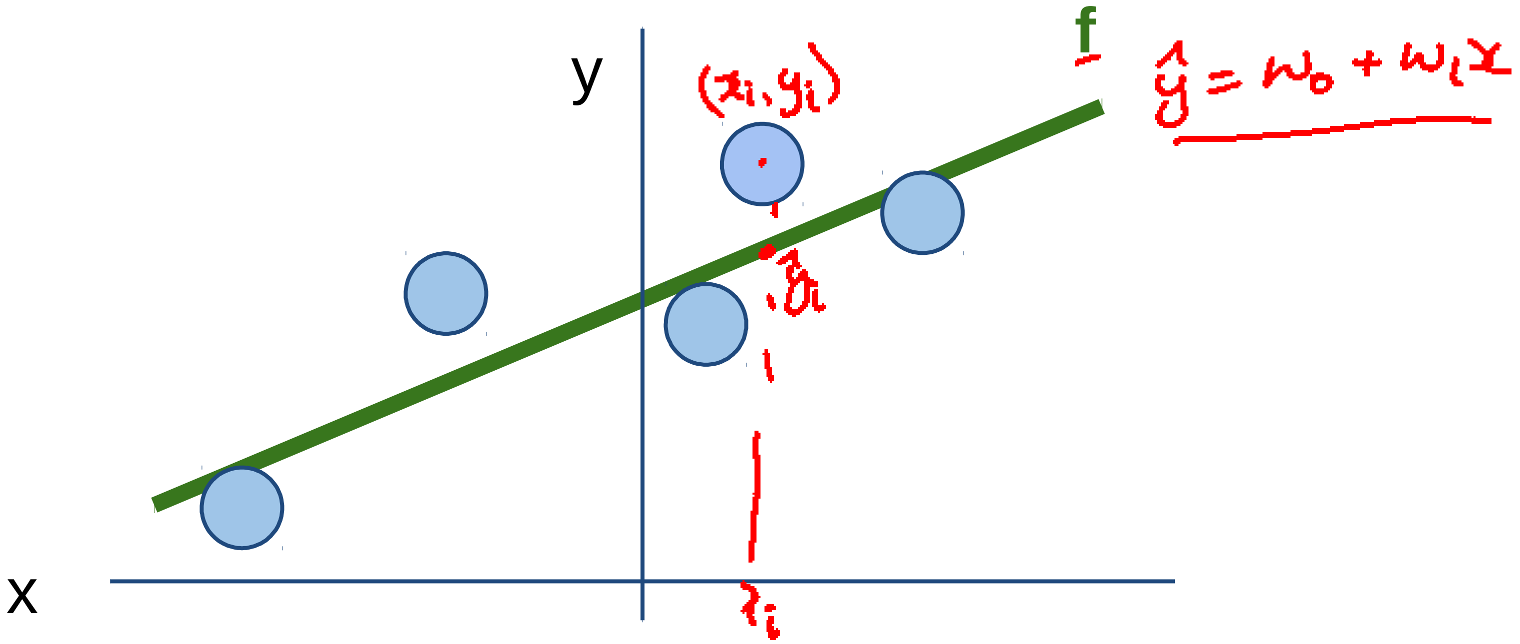
Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Regularization Intuition in Polynomial Regression



Polynomial Regression



Polynomial Regression

$$\hat{y} = w_0 + w_1 x \leftarrow \text{1st order poly}$$

$$= w_0 + w_1 x + w_2 x^2$$

$$= w_0 + \ldots + w_d x^d \leftarrow d^{th} \text{ poly}$$

$$= [w_0 \ldots w_d] \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^d \end{bmatrix}$$

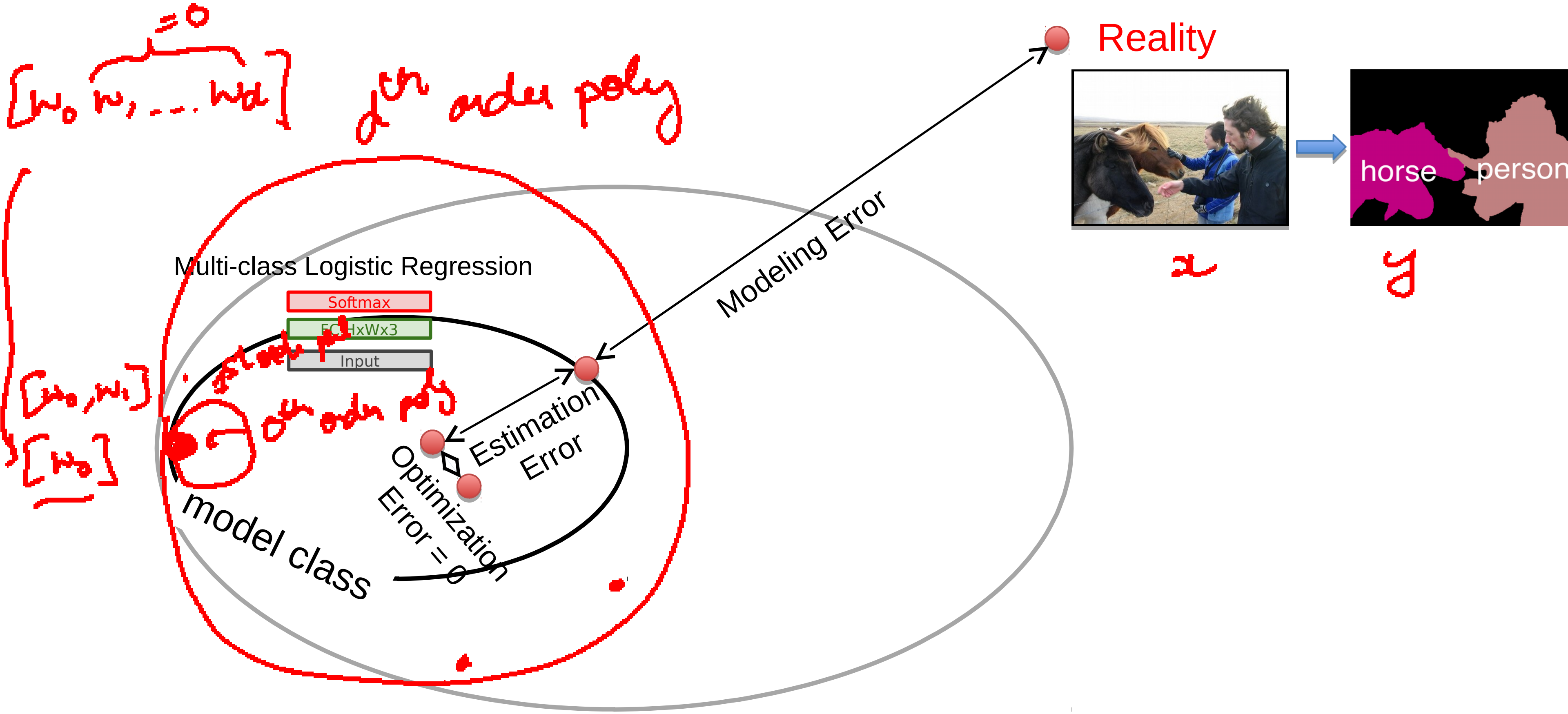
$$= \vec{w}^T \vec{\phi}(x)$$

linear in $\vec{w}$

$$\min_{\vec{w}} L(w, D) = \frac{1}{n} \sum_{i=1}^{N} (y_i - \vec{w}^T \phi(x_i))^2$$

Polynomial Regression

Error Decomposition

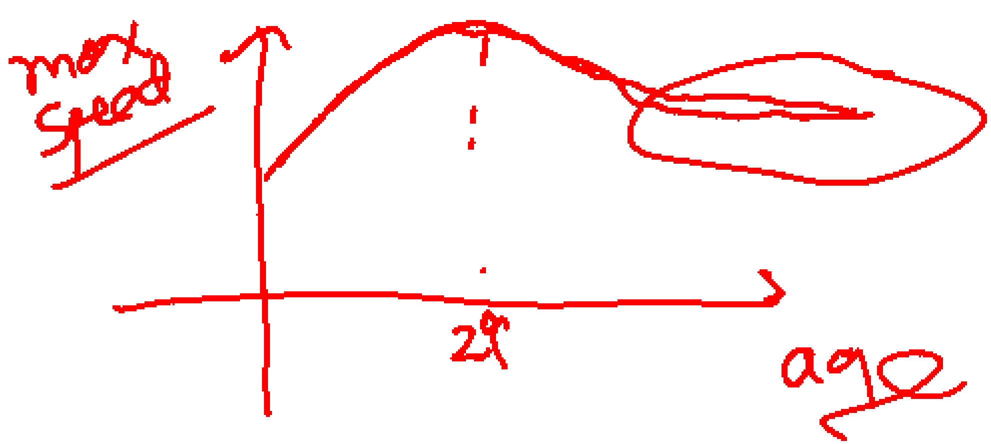


Polynomial Regression

- Demo: https://arachnoid.com/polysolve/

- You are a scientist studying runners.
  - You measure average speeds of the best runners at different ages.

- Data: Age (years), Speed (mph)
  - 10 6
  - 15 9
  - 20 11
  - 25 12
  - 29 13
  - 40 11
  - 50 10
  - 60 9



Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{} + \underbrace{\lambda R(W)}_{}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \underbrace{\sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|}$

Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Simple examples

L2 regularization: $\boxed{R(W)} = \boxed{\sum_k \sum_l W_{k,l}^2}$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

Dropout

Batch normalization

Stochastic depth, fractional pooling, etc

Regularization

λ = regularization strength (hyperparameter)

$$\underbrace{L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Why regularize?

- Express preferences over weights
- Make the model *simple* so it works on test data
- Improve optimization by adding curvature

Recap

- We have some dataset of (x,y)
- We have a **score function:** $s = f(x; W) \overset{\text{e.g.}}{=} Wx$
- We have a **loss function**:

Softmax

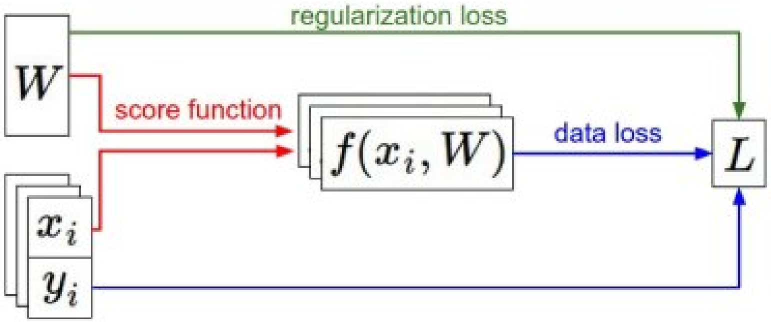
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

SVM

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

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

Full loss



Recap

How do we find the best W?

- We have some dataset of (x,y)
- We have a **score function:** $s = f(x; W) \overset{\text{e.g.}}{=} Wx$
- We have a **loss function**:

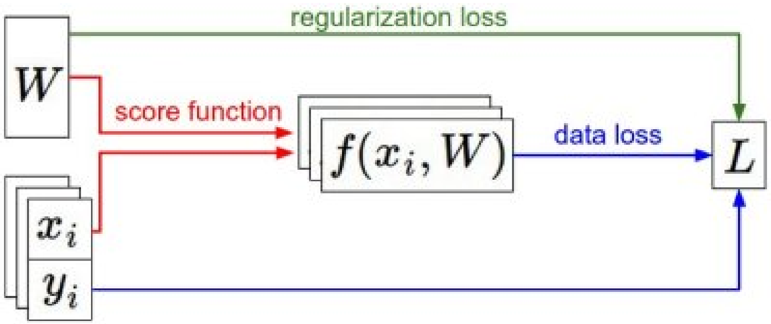
Softmax

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

SVM

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

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



Next: Neural Networks

So far: Linear Classifiers

"Raw Input"
x

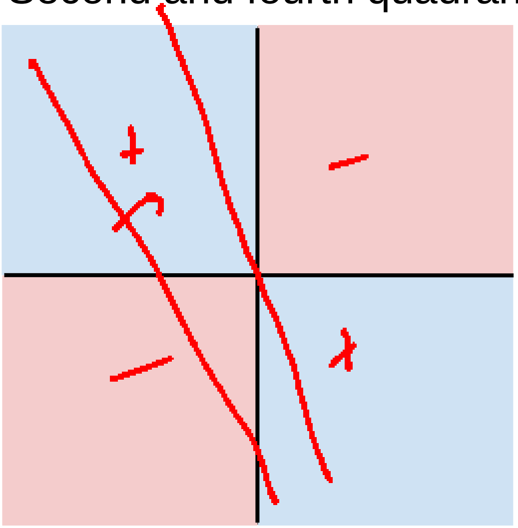
$$f(x) = Wx$$

Class scores

Hard cases for a linear classifier

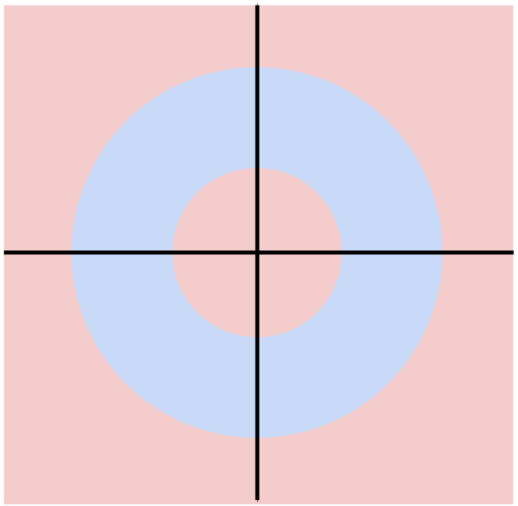
Class 1:
First and third quadrants

Class 2:
Second and fourth quadrants



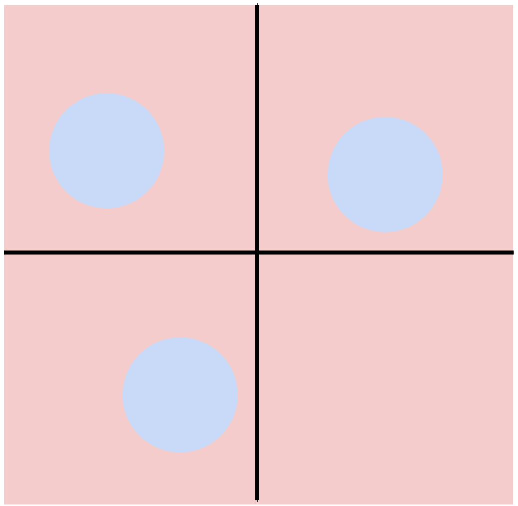
Class 1:
1 <= L2 norm <= 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else

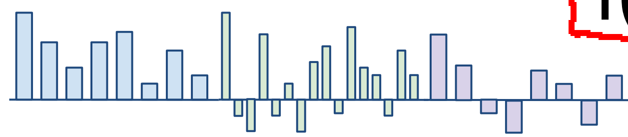


Aside: Image Features

$\phi(x)$

Raw Input

x

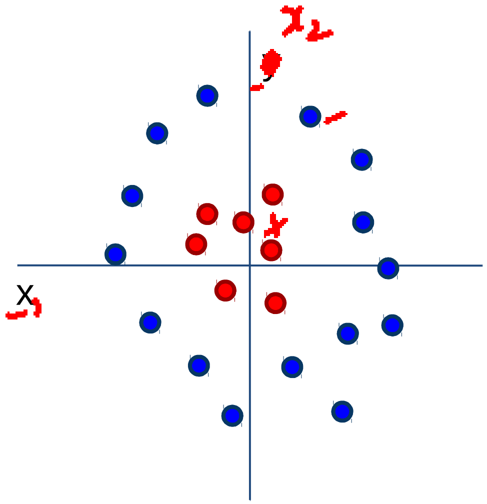


Feature Representation

$f(x) = Wx$ $\phi(x)$

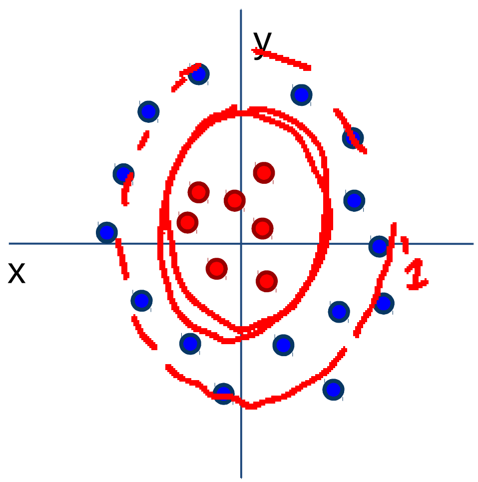
Class scores

Image Features: Motivation

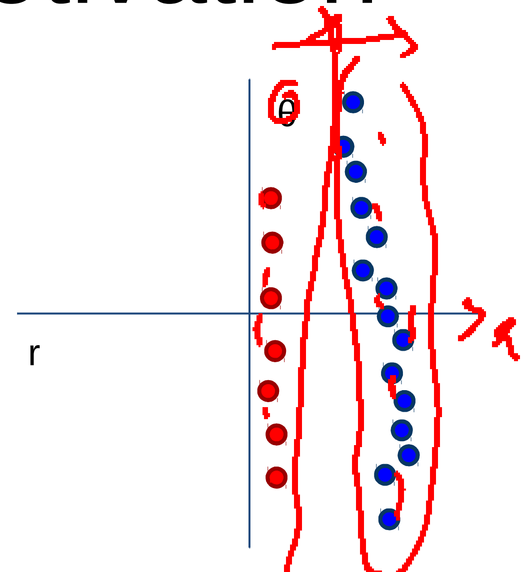


Cannot separate red and blue points with linear classifier

Image Features: Motivation



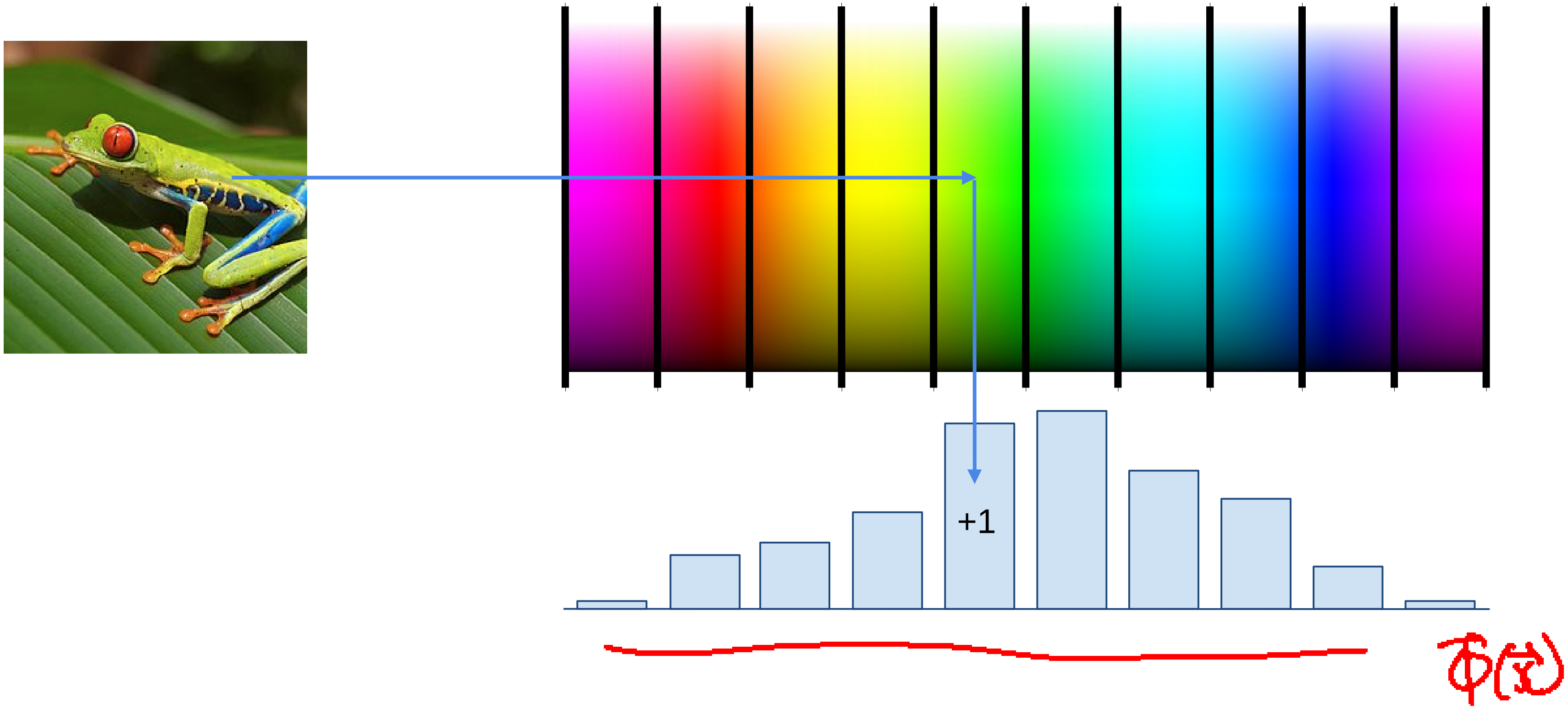
$f(x, y) = (r(x, y), \theta(x, y))$



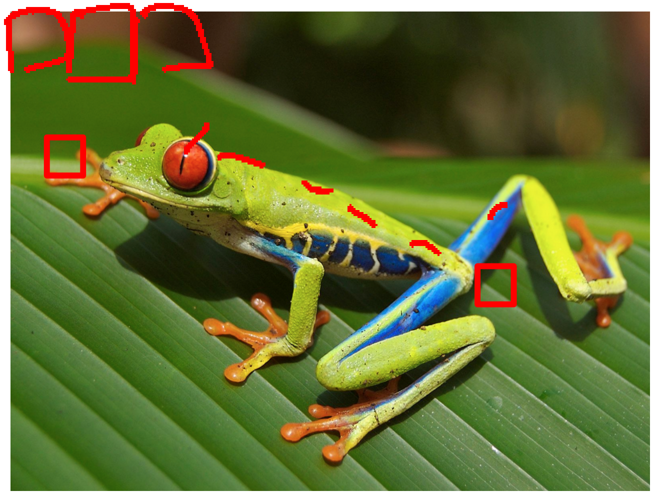
Cannot separate red and blue points with linear classifier

After applying feature transform, points can be separated by linear classifier

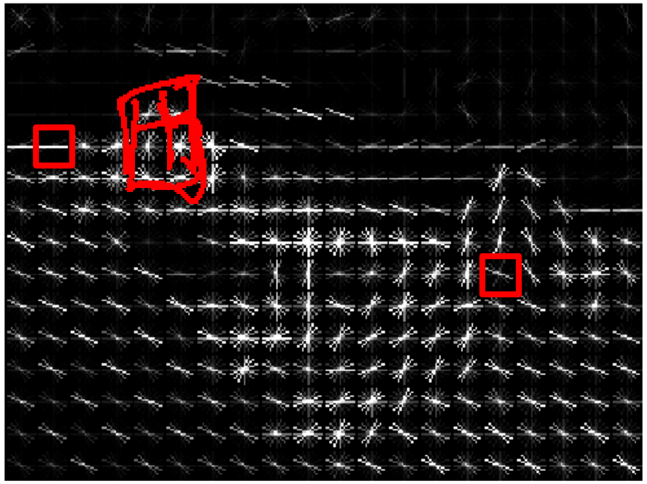
Example: Color Histogram



Example: Histogram of Oriented Gradients (HoG)



Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins



Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Lowe, “Object recognition from local scale-invariant features”, ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

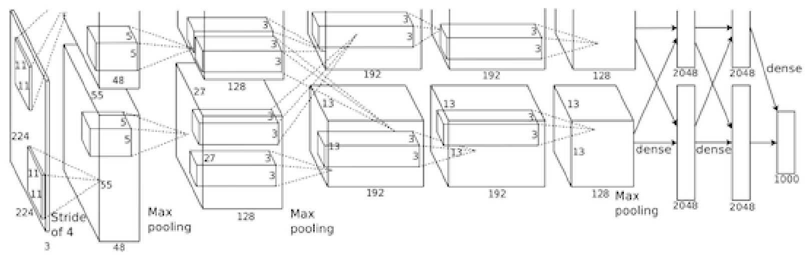
Image features vs Neural Nets

$\phi(x)$

f

10 numbers giving scores for classes

training



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

x

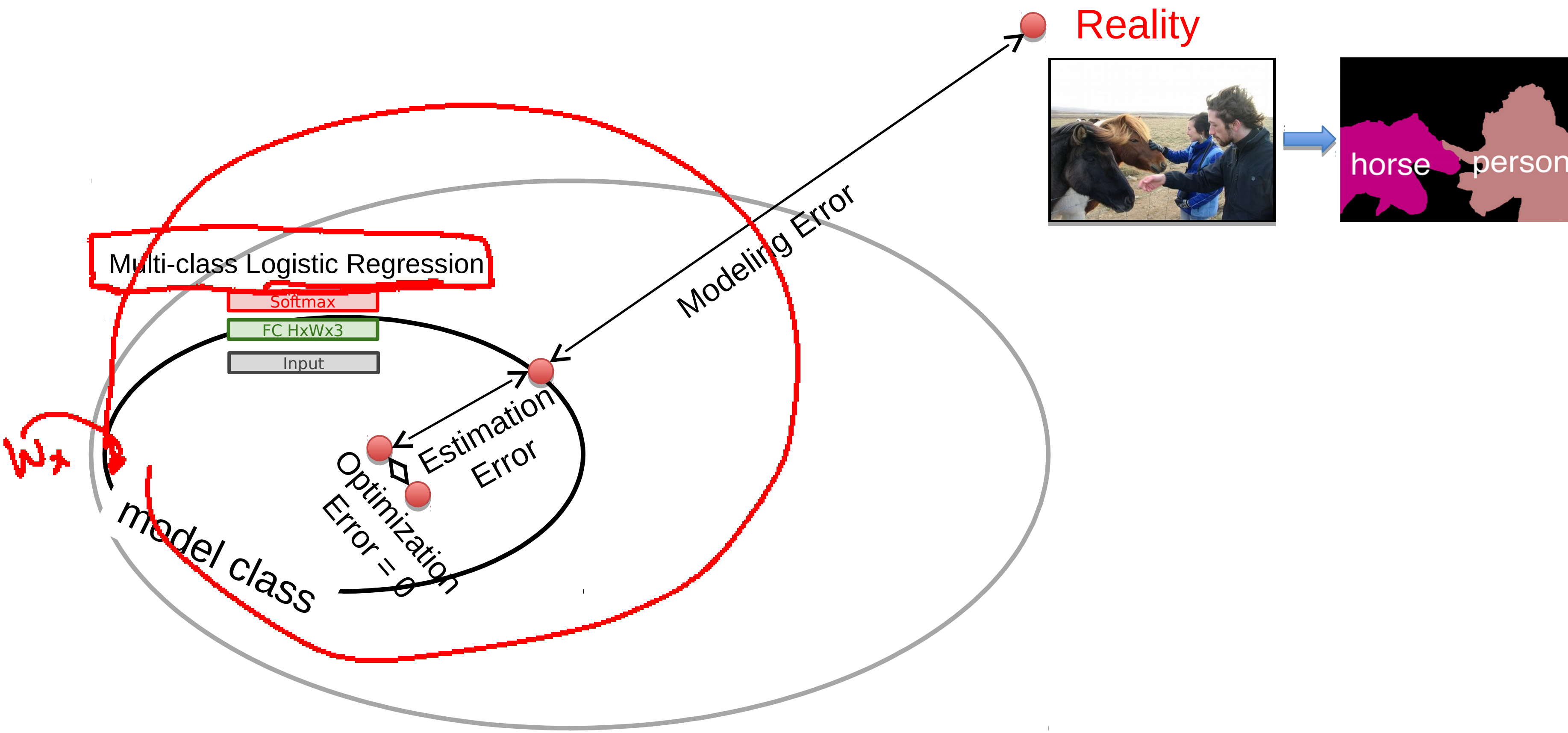
10 numbers giving scores for classes

training

$\phi^1(x)$ $\phi^2(x)$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Error Decomposition



Neural networks: without the brain stuff

$\mathcal{H} = \{h : X \to Y\}$

(**Before**) Linear score function: $\boxed{f} = Wx$

$\ldots \boxed{W_2 W_1} x$

$\boxed{W_3} x$

Neural networks: without the brain stuff

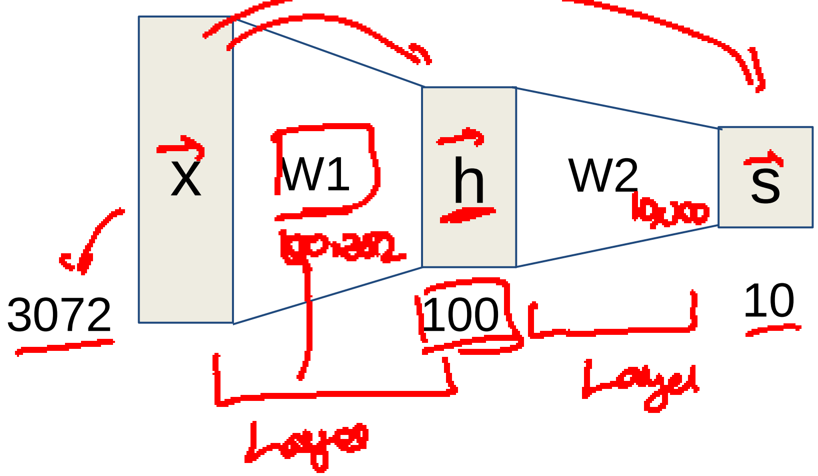
(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 \vec{x})$

Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

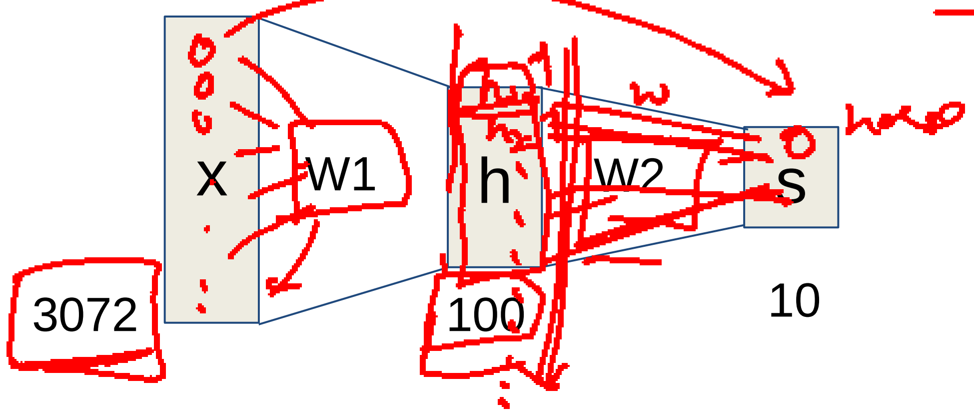
(**Now**) 2-layer Neural Network $f = W_2 \boxed{\max}(0, W_1 x)$

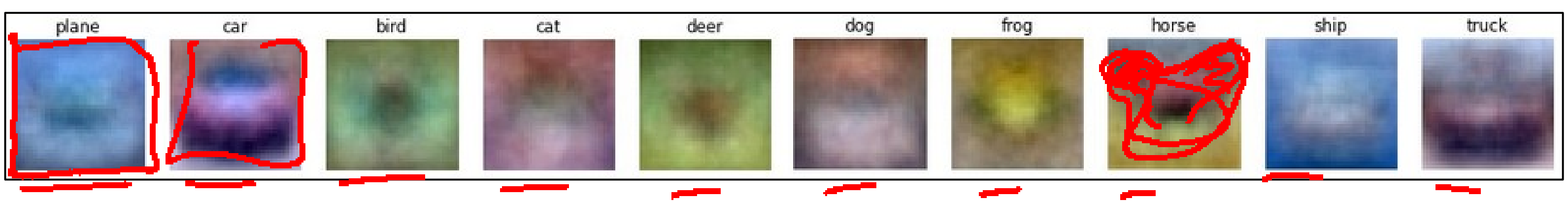


Neural networks: without the brain stuff

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$





W_1

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Neural networks: without the brain stuff

$L(\boxed{\{W_1, \ldots W_3\}}, D)$ $L_i(\hat{y}(x_i), y_i)$

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$\boxed{\vec{s}}$ $\boxed{\vec{s}}$

$\vec{x} \xrightarrow[W_1]{} \vec{h}^{(1)} \xrightarrow[W_2]{} \vec{h}^{(2)} \xrightarrow[W_3]{} \vec{s}$

Multilayer Networks

- Cascaded "neurons"
- The output from one layer is the input to the next
- Each layer has its own sets of weights

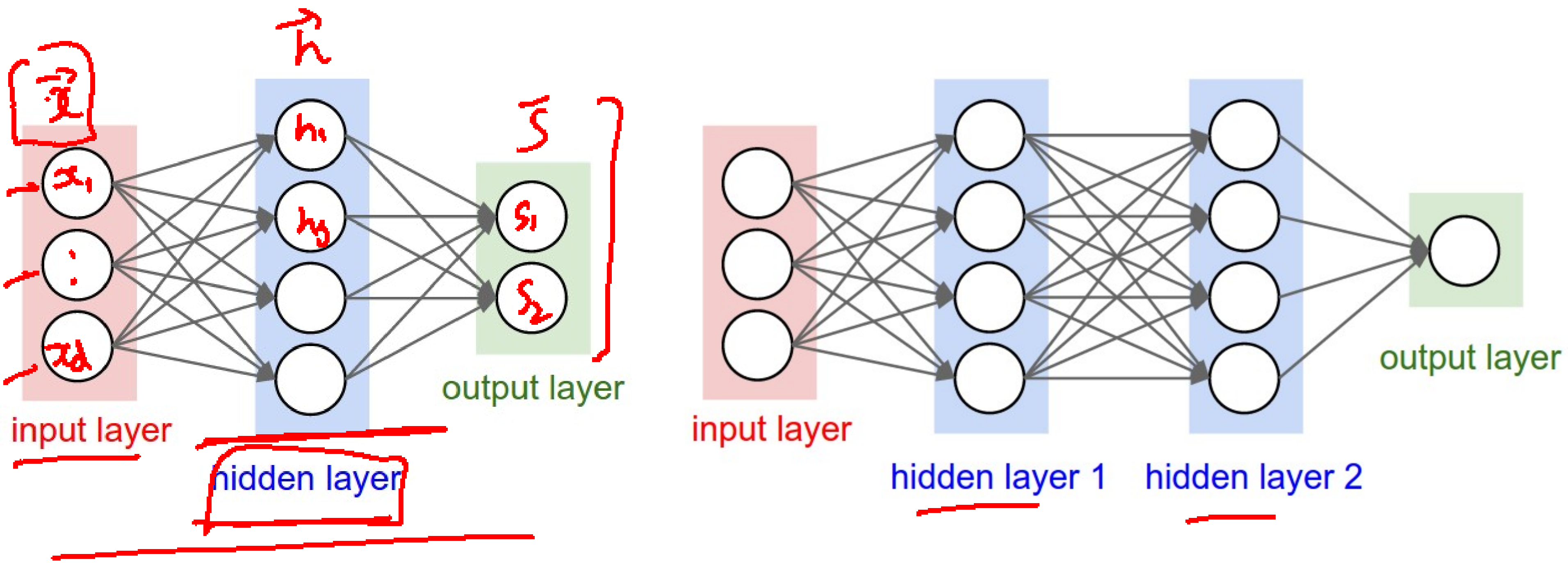
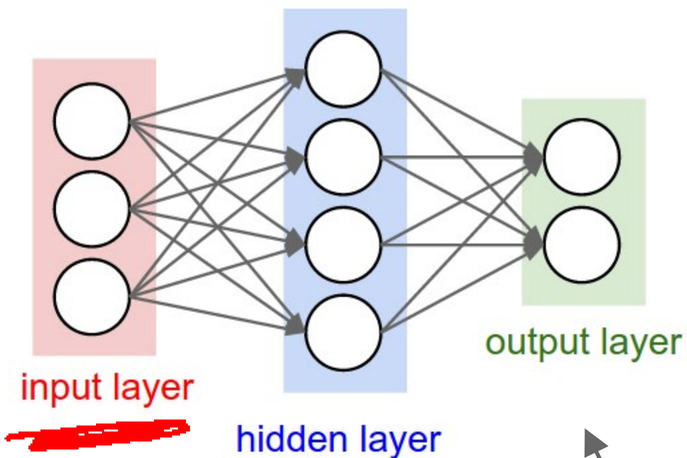
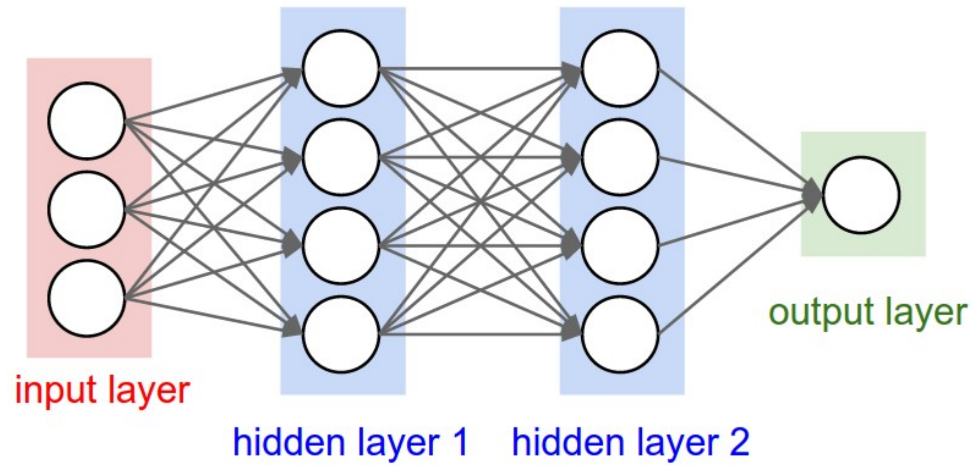


Image Credit: Andrej Karpathy, CS231n

Neural networks: Architectures

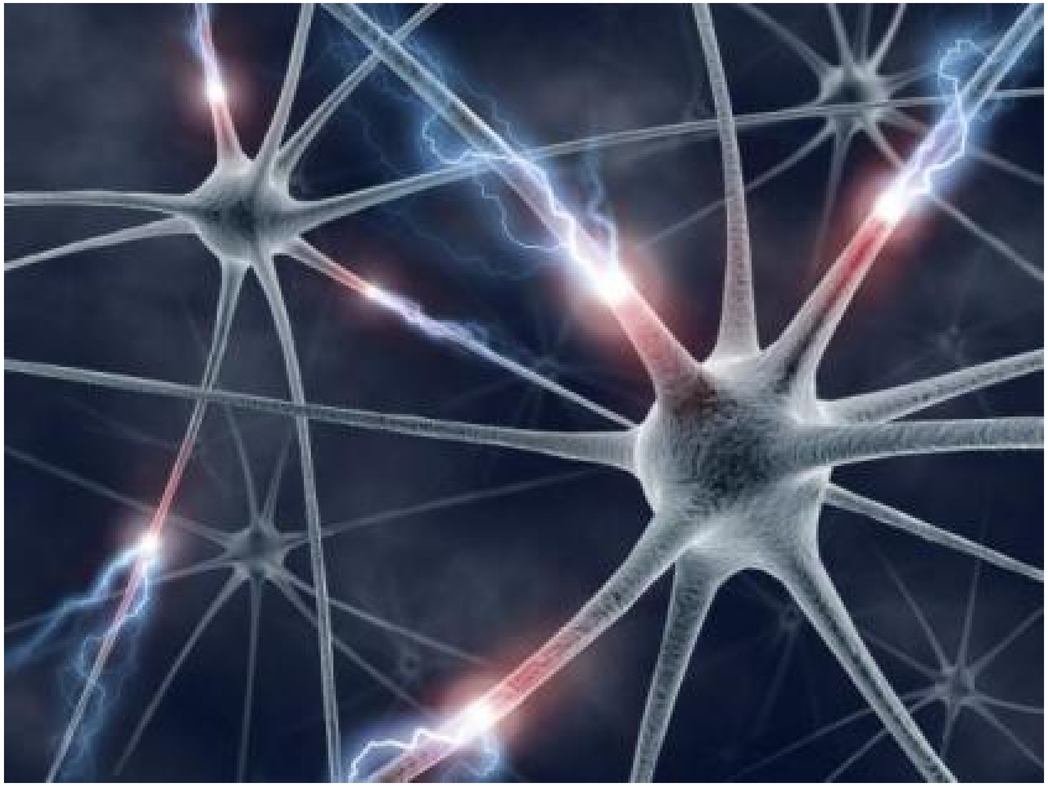


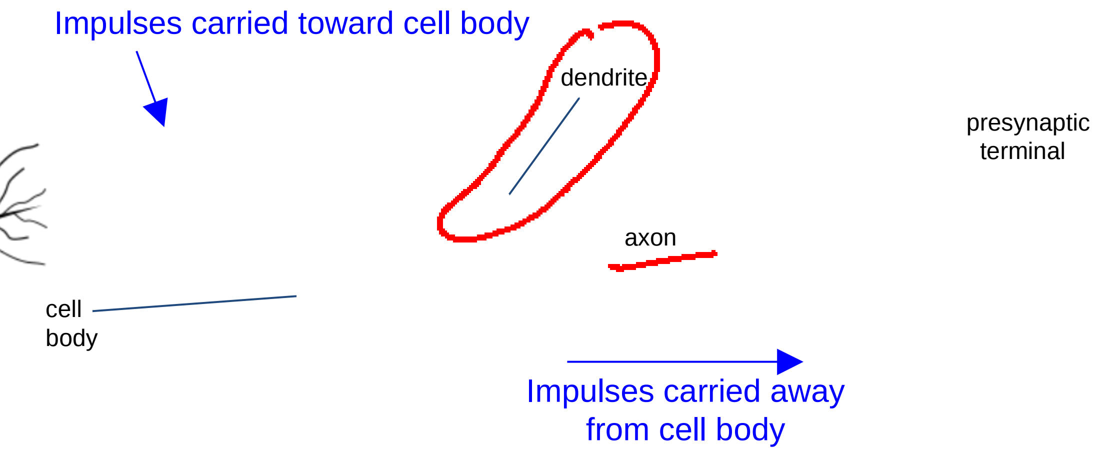


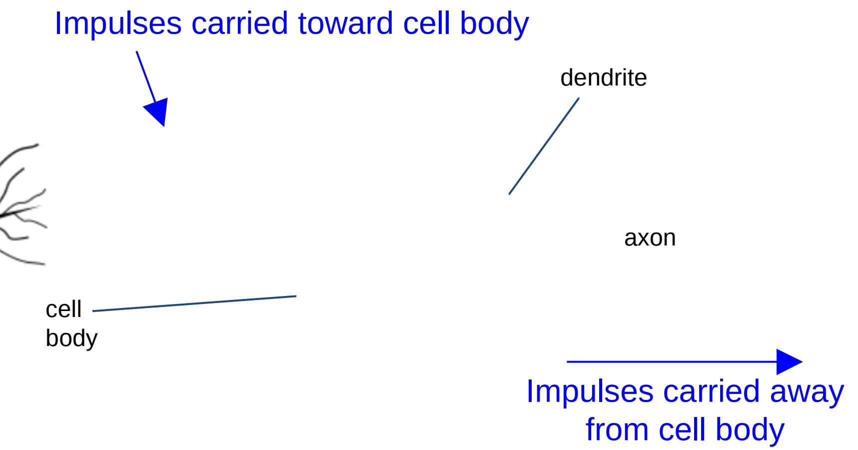
"2-layer Neural Net", or
"1-hidden-layer Neural Net"

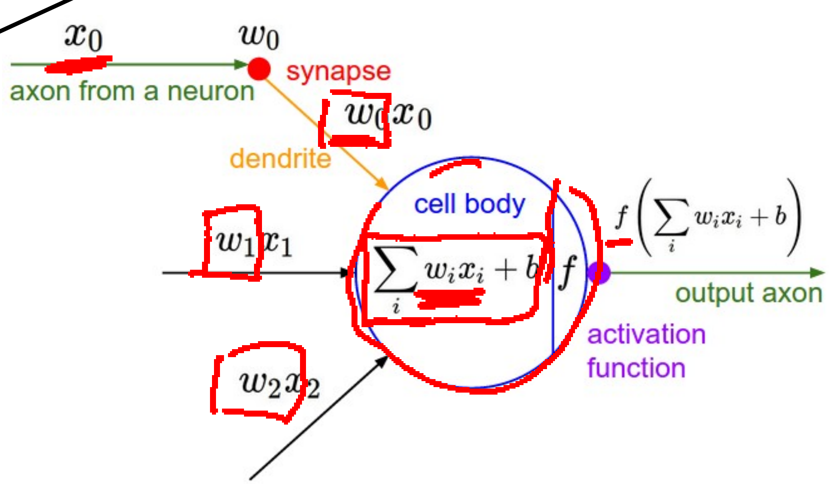
"Fully-connected" layers

"3-layer Neural Net", or
"2-hidden-layer Neural Net"




Impulses carried toward cell body
dendrite
presynaptic
terminal
axon
cell
body
Impulses carried away
from cell body


Impulses carried toward cell body
dendrite
presynaptic terminal
axon
cell body
Impulses carried away from cell body


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

Impulses carried toward cell body

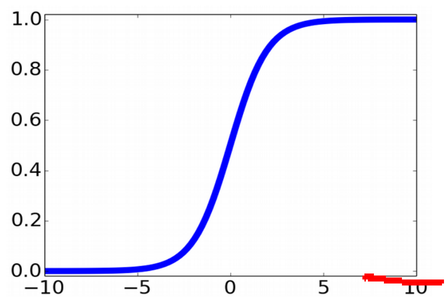
dendrite

presynaptic terminal

axon

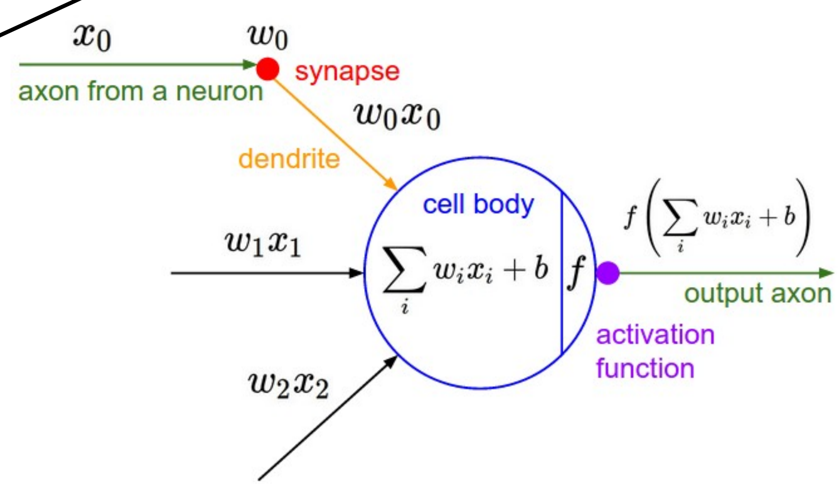
cell body

Impulses carried away from cell body



sigmoid activation function

$$\frac{1}{1+e^{-x}}$$



Be very careful with your brain analogies!

Biological Neurons:

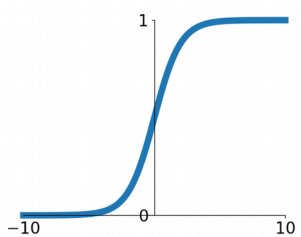
- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate

[Dendritic Computation. London and Hausser]

Activation functions

Sigmoid

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

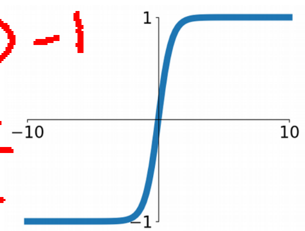


tanh

$\tanh(x)$

$2\sigma(2x) - 1$

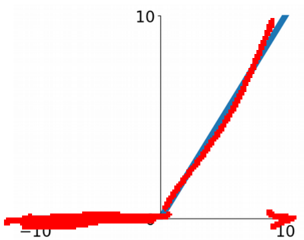
$\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$



ReLU

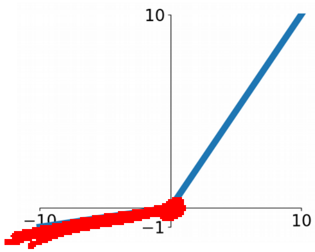
$\max(0, x)$

$\vec{h}$



Leaky ReLU

$\max(0.1x, x)$



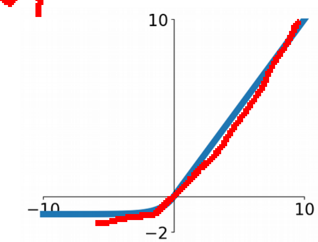
Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

$w_2^T x$ $w_1^T x$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

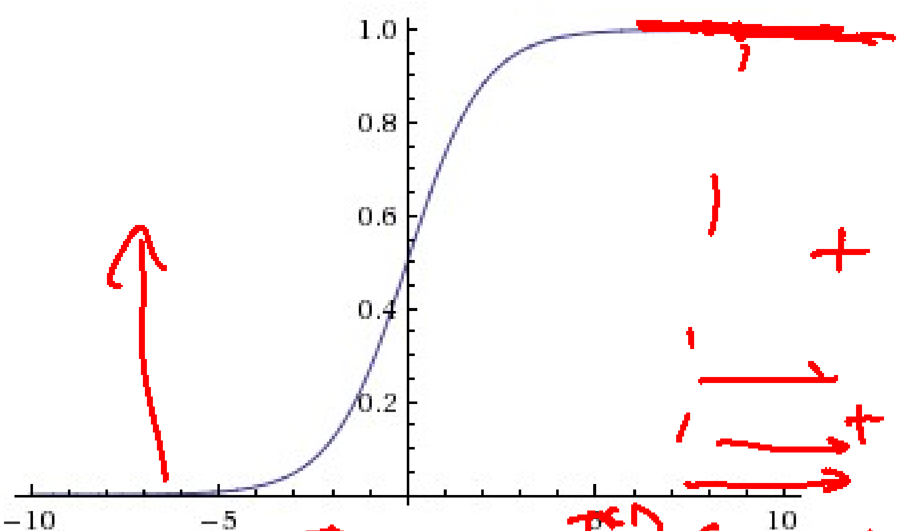


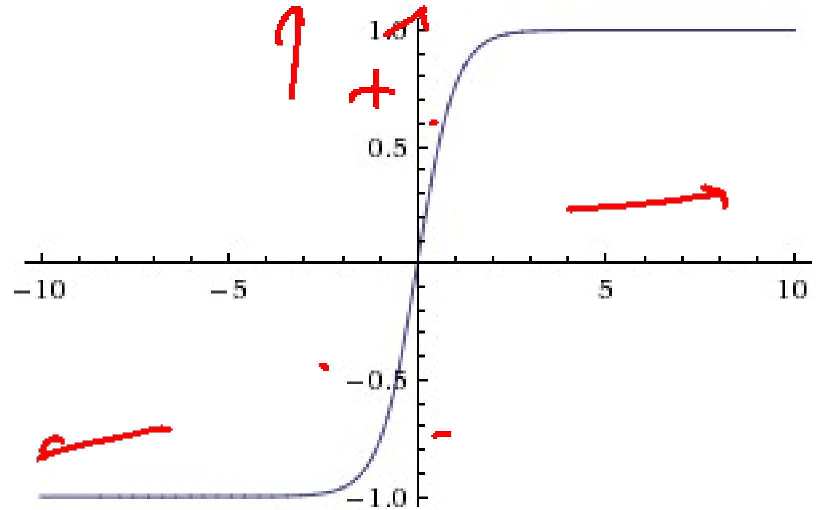
S

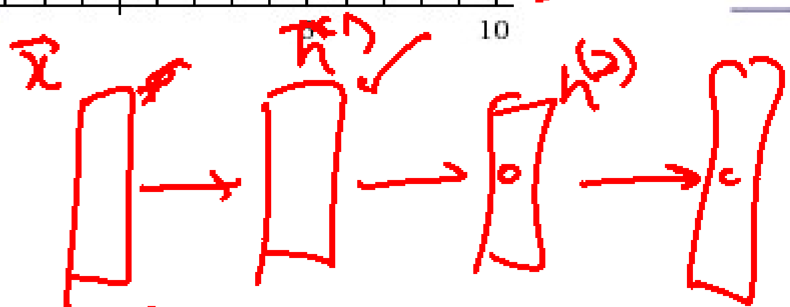
Activation Functions

$2\sigma(2a) - 1$

- sigmoid vs tanh







$(\sigma(\Sigma\sigma(\Sigma h_i)))$ - -

A quick note

$\sigma\left(\sum w_{ji} h_i + b\right)$

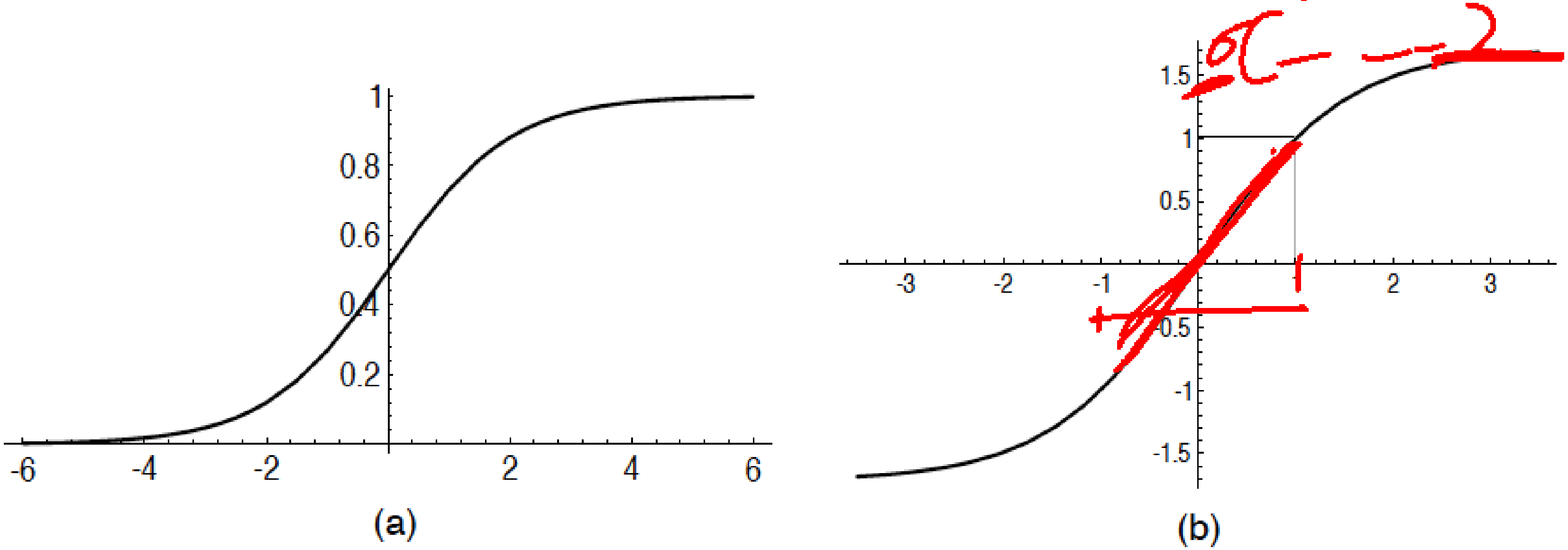
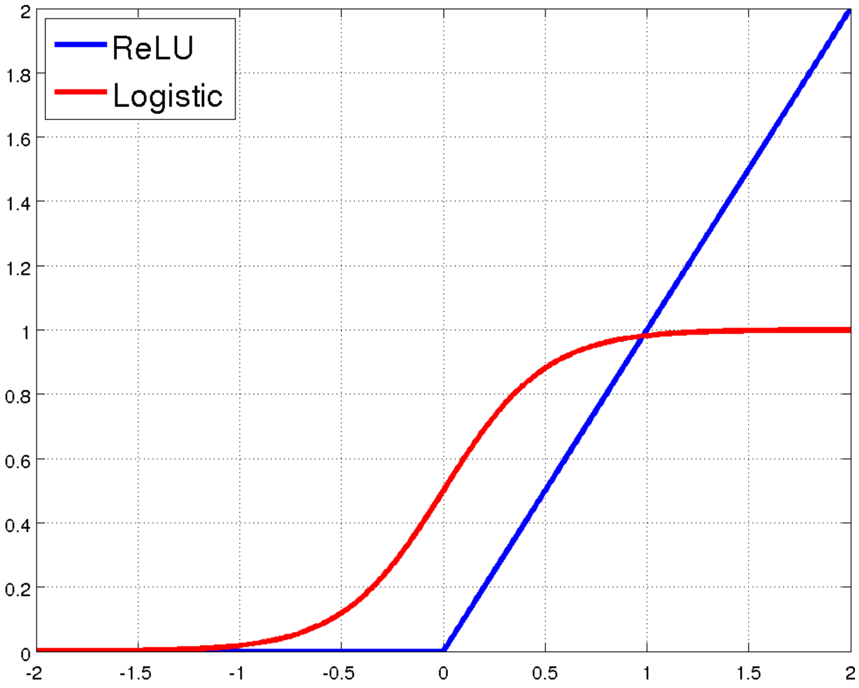
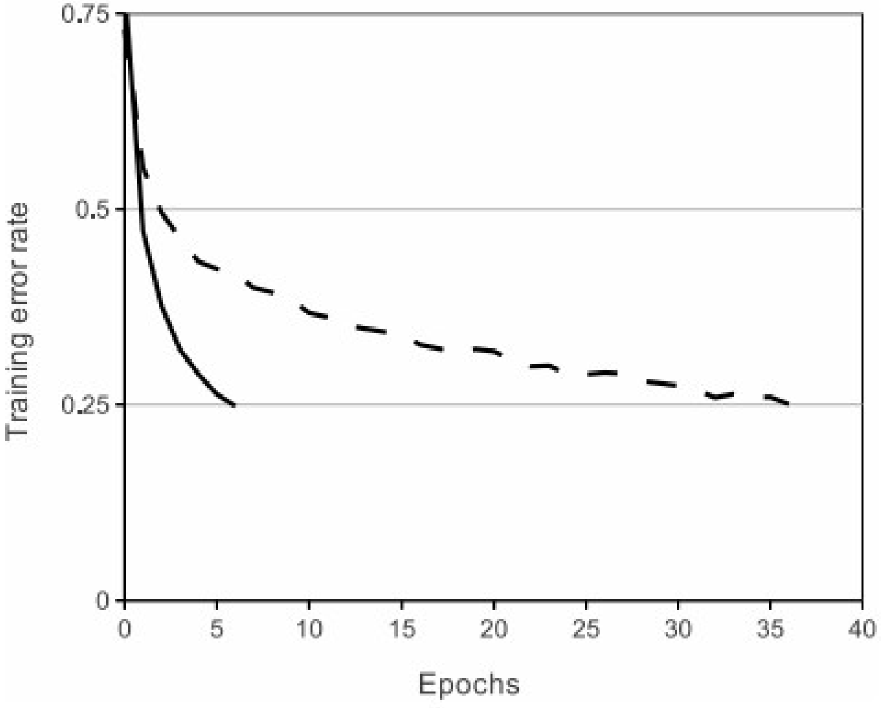


Fig. 4. (a) Not recommended: the standard logistic function, $f(x) = 1/(1 + e^{-x})$. (b) Hyperbolic tangent, $f(x) = 1.7159 \tanh\left(\frac{2}{3}x\right)$.

Rectified Linear Units (ReLU)





[Krizhevsky et al., NIPS12]

Demo Time

- https://playground.tensorflow.org