《多模态机器学习》

第二章 基础概念

黄文炳
中国人民大学高瓴人工智能学院
hwenbing@126.com
2024年秋季

内容提纲

① 单模态基础表示

② 经典机器学习方法

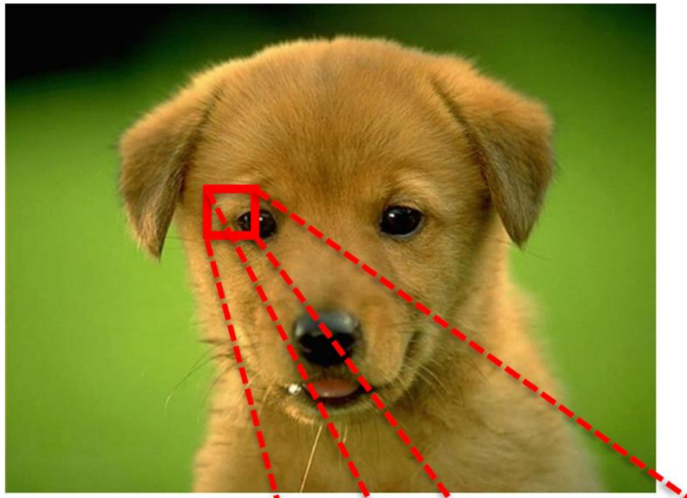
③ 神经网络

④ 优化方法简介

内容提纲

Unimodal Representation: Visual Modality

Color image



Each pixel is represented in $\mathbb{R}^d$, d is the number of colors (d=3 for RGB)

88	82	84	88	85	83	80	93	102
88	80	78	80	80	78	73	94	100
85	79	80	78	77	74	65	91	99
38	35	40	35	39	74	77	70	65
20	25	23	28	37	69	64	60	57
22	26	22	28	40	65	64	59	34
24	28	24	30	37	60	58	56	66
21	22	23	27	38	60	67	65	67
23	22	22	25	38	59	64	67	66

Input observation x_i

88
88
85
38
20
22
24
21
23
82
80
79
35
25
26
28
22
22
84
78
80
⋮

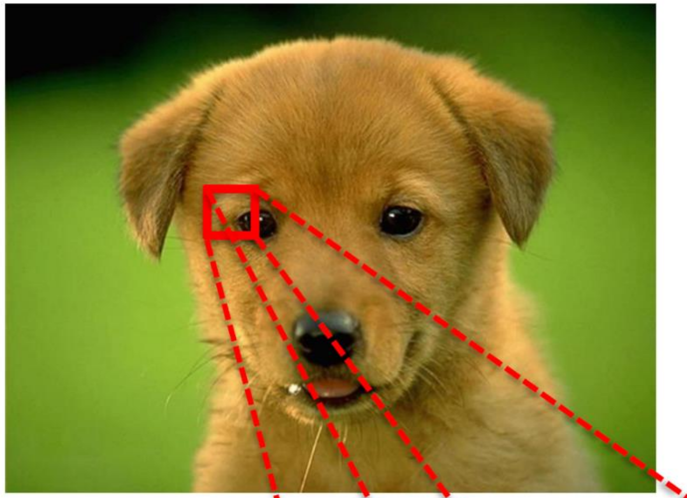
Binary classification problem

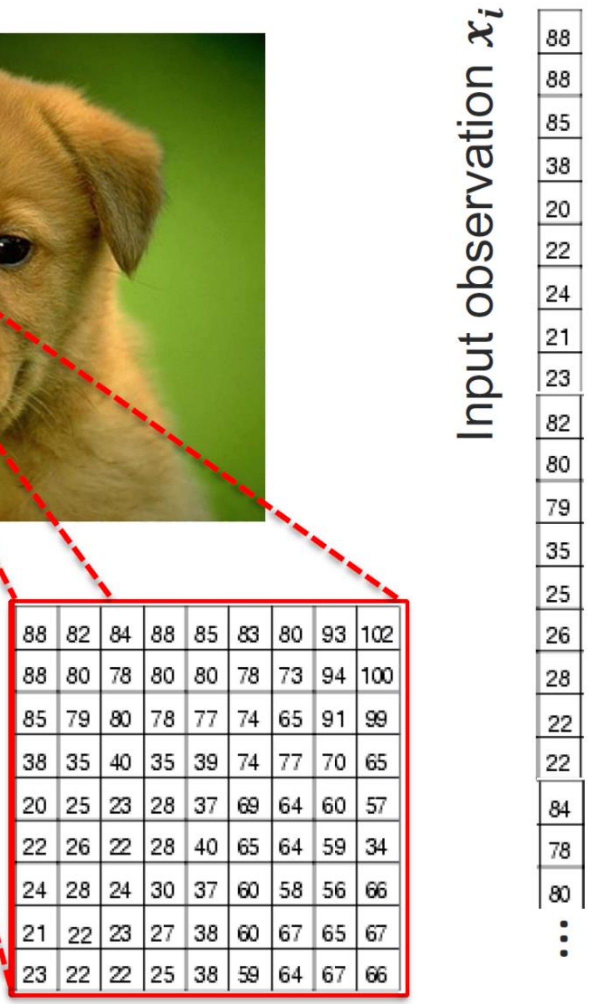
Dog ?

label $y_i \in \mathcal{Y} = \{0,1\}$

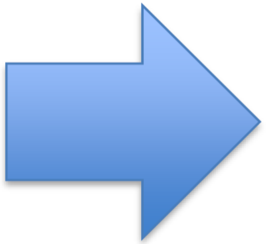
Unimodal Representation: Visual Modality

Color image





Each pixel is represented in $\mathbb{R}^d$, d is the number of colors (d=3 for RGB)



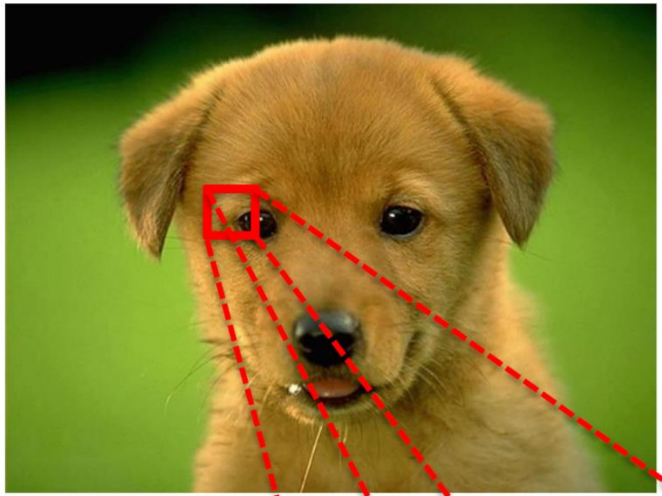
Multi-class classification problem

Duck
-or-
Cat
-or-
Dog
-or-
Pig
-or-
Bird ?

label $y_i \in \mathcal{Y} = \{0,1,2,3, \dots\}$

Unimodal Representation: Visual Modality

Color image



Each pixel is represented in $\mathbb{R}^d$, d is the number of colors (d=3 for RGB)

88	82	84	88	85	83	80	93	102
88	80	78	80	80	78	73	94	100
85	79	80	78	77	74	65	91	99
38	35	40	35	39	74	77	70	65
20	25	23	28	37	69	64	60	57
22	26	22	28	40	65	64	59	34
24	28	24	30	37	60	58	56	66
21	22	23	27	38	60	67	65	67
23	22	22	25	38	59	64	67	66

Input observation x_i

88
88
85
38
20
22
24
21
23
82
80
79
35
25
26
28
22
22
84
78
80
⋮

Multi-label (or multi-task) classification problem

Duck?

Cat ?

Dog ?

Pig ?

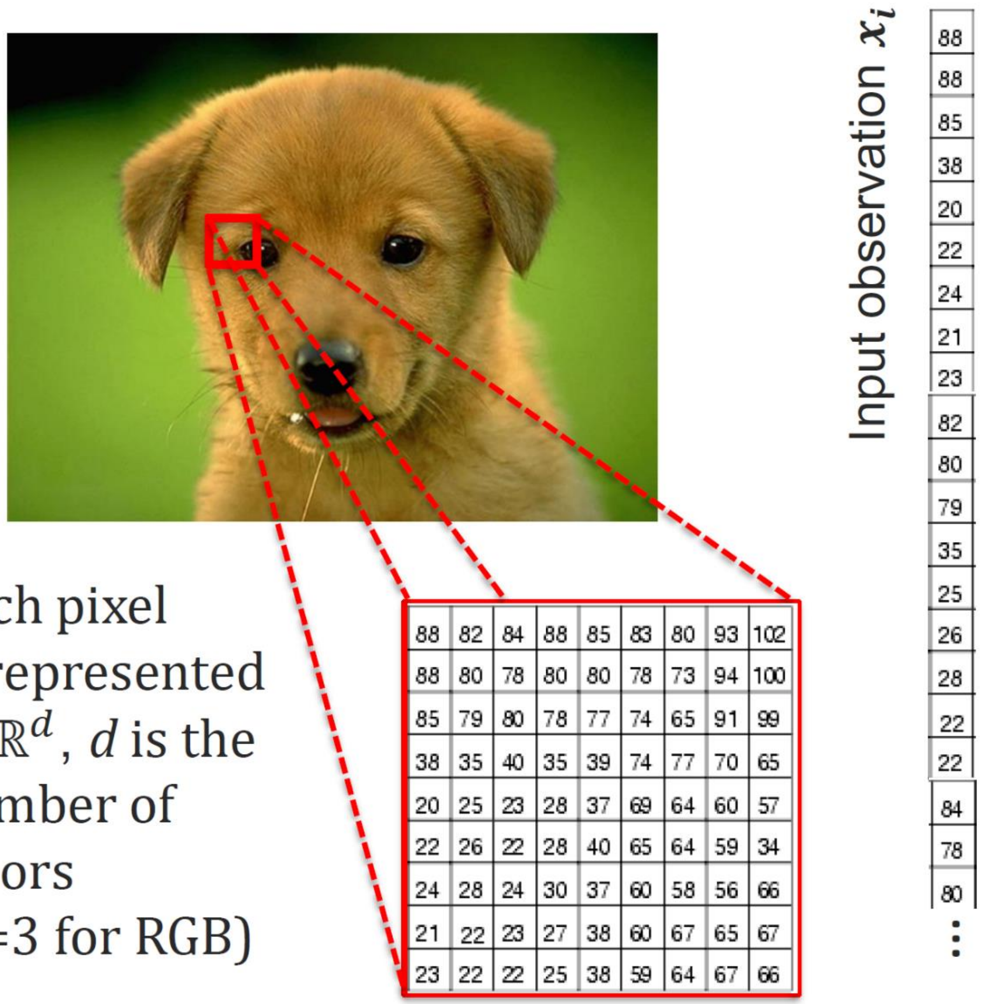
Bird ?

Puppy ?

label vector $\boldsymbol{y}_i \in \mathcal{Y}^m = \{0,1\}^m$

Unimodal Representation: Visual Modality

Color image



Each pixel is represented in $\mathbb{R}^d$, d is the number of colors (d=3 for RGB)

Multi-label (or multi-task) regression problem

Age ?

Height ?

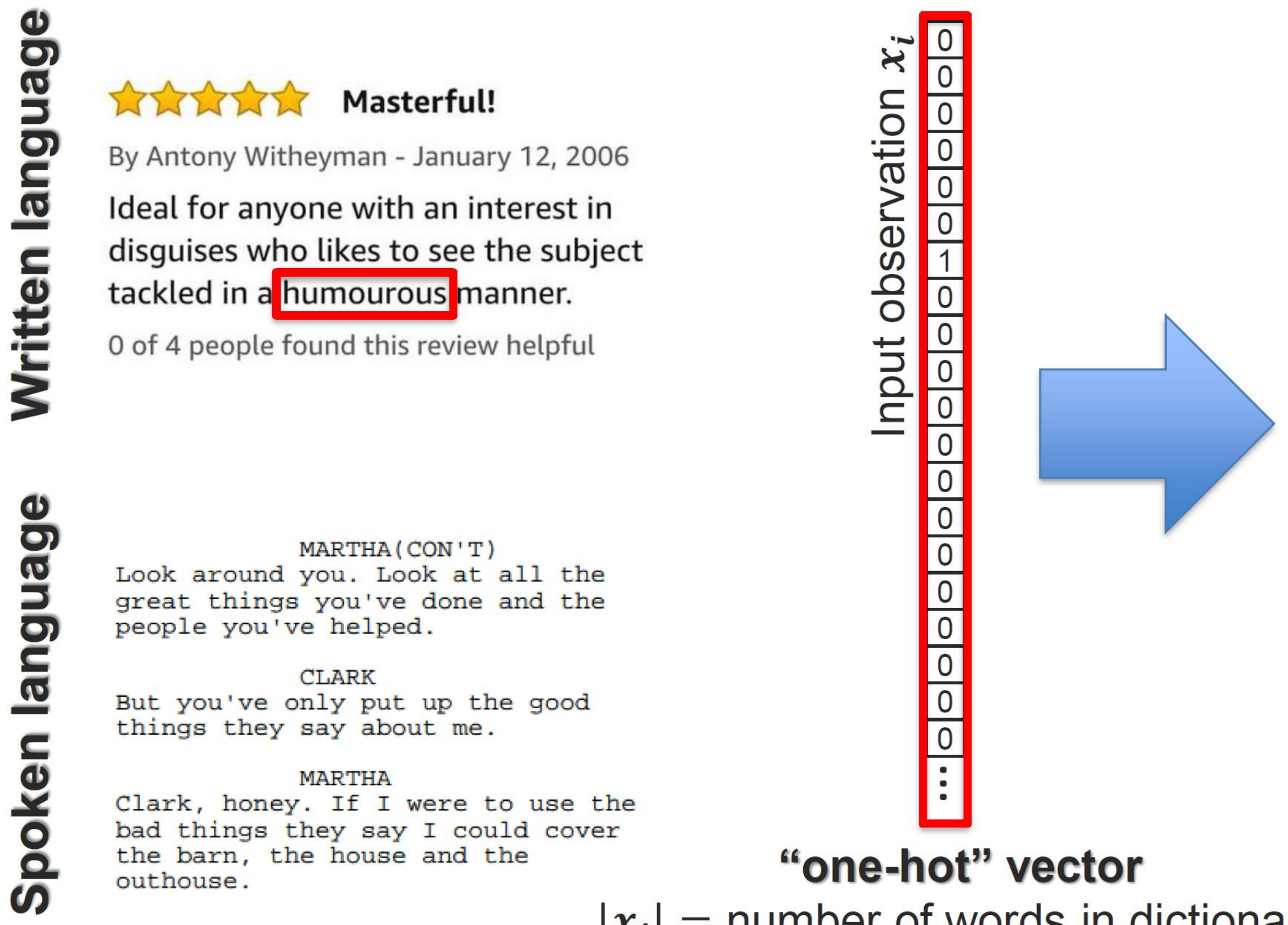
Weight ?

Distance ?

Happy ?

label vector $\boldsymbol{y}_i \in \mathcal{Y}^m = \mathbb{R}^m$

Unimodal Representation: Language Modality



"one-hot" vector

$|x_i|$ = number of words in dictionary

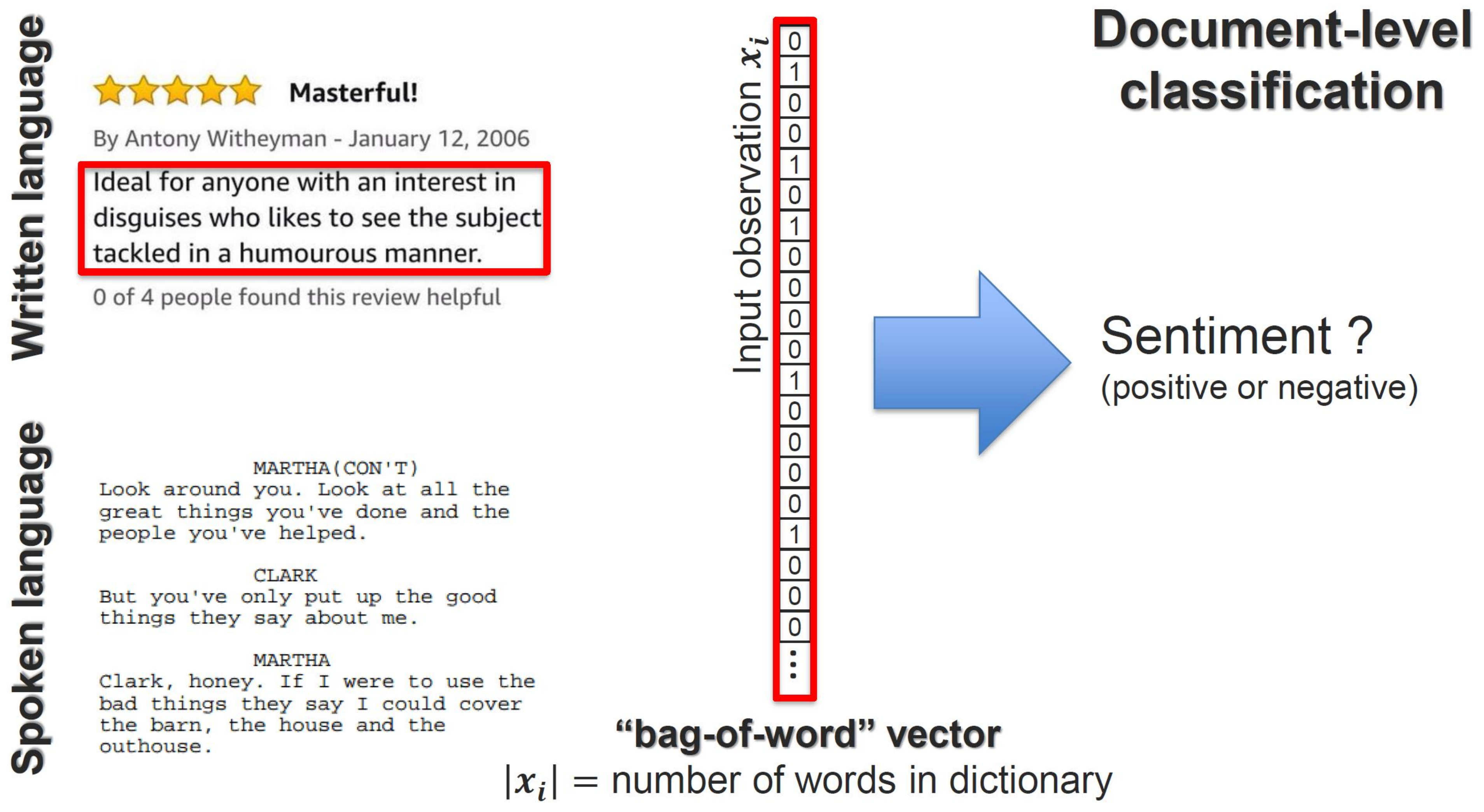
Word-level classification

Part-of-speech ?
(noun, verb,…)

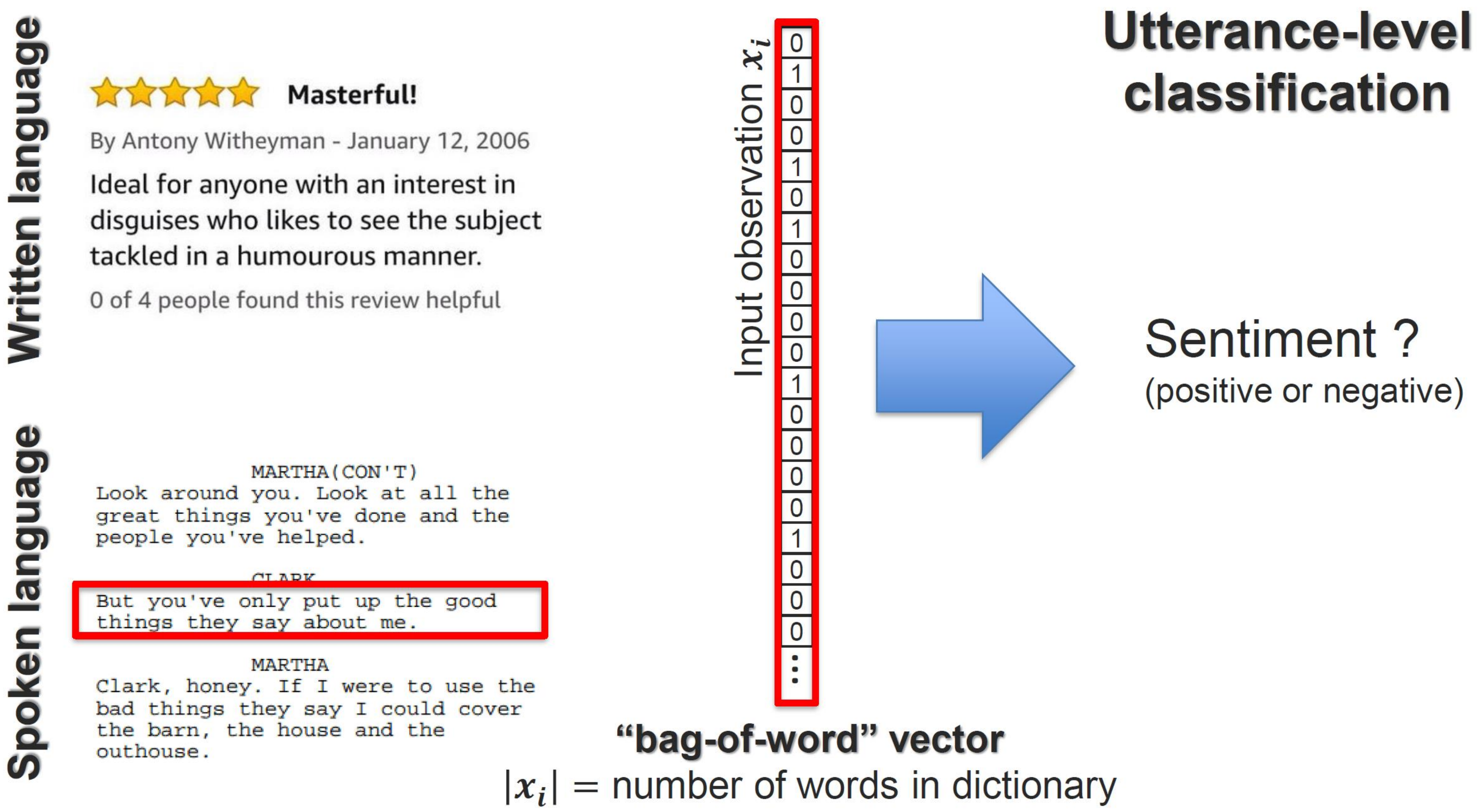
Sentiment ?
(positive or negative)

Named entity ?
(names of person,…)

Unimodal Representation: Language Modality



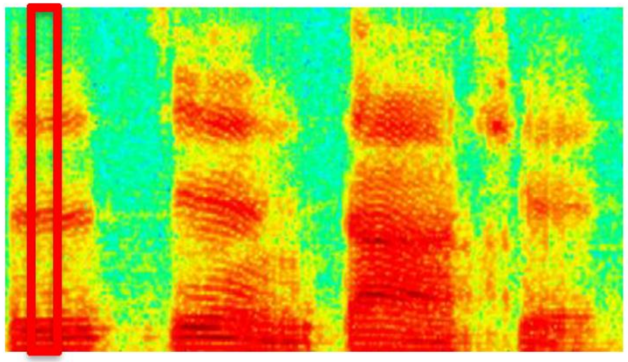
Unimodal Representation: Language Modality



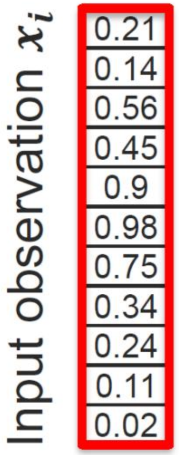
Unimodal Representation: Acoustic Modality

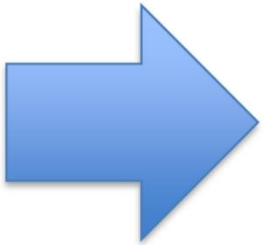
Digitalized acoustic signal

- Sampling rates: 8~96kHz
- Bit depth: 8, 16 or 24 bits
- Time window size: 20ms
  - Offset: 10ms



Spectogram



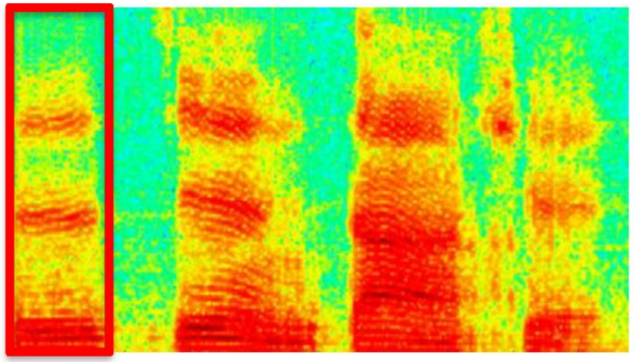


Spoken word ?

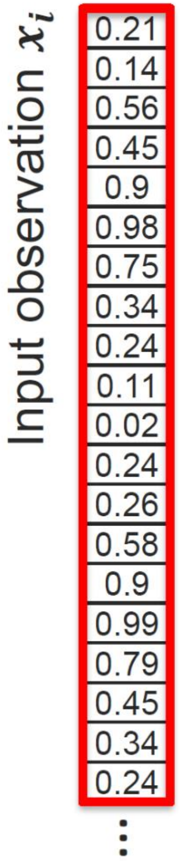
Unimodal Representation: Acoustic Modality

Digitalized acoustic signal

- Sampling rates: 8~96kHz
- Bit depth: 8, 16 or 24 bits
- Time window size: 20ms
  - Offset: 10ms



Spectogram



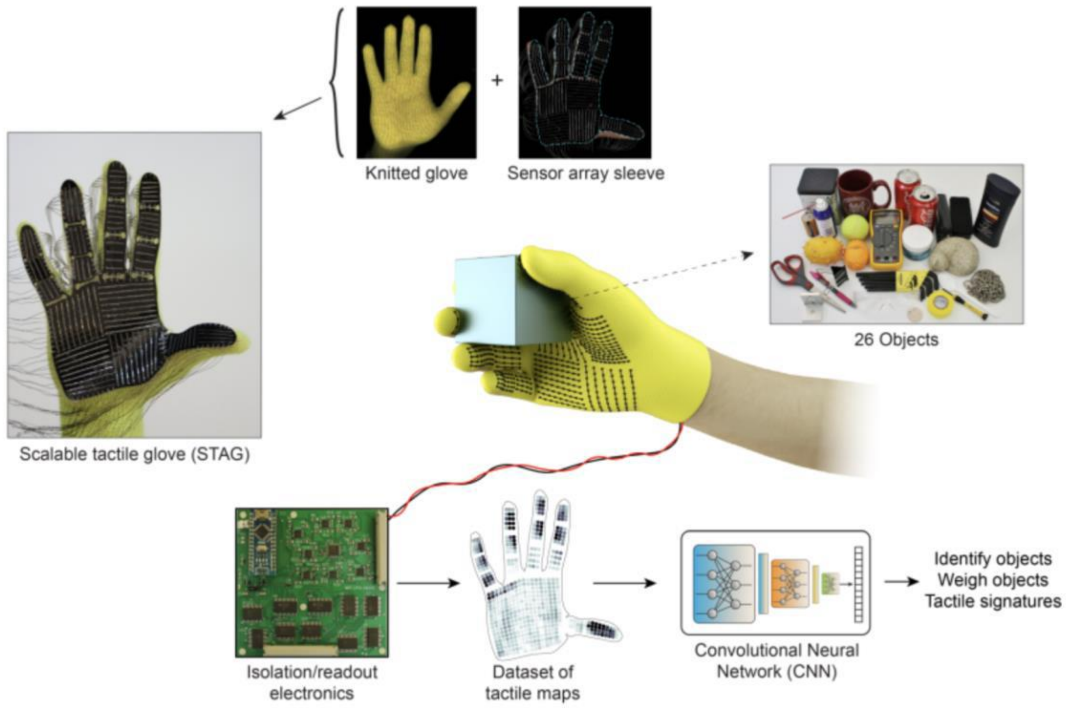
Emotion ?

Spoken word ?

Voice quality ?

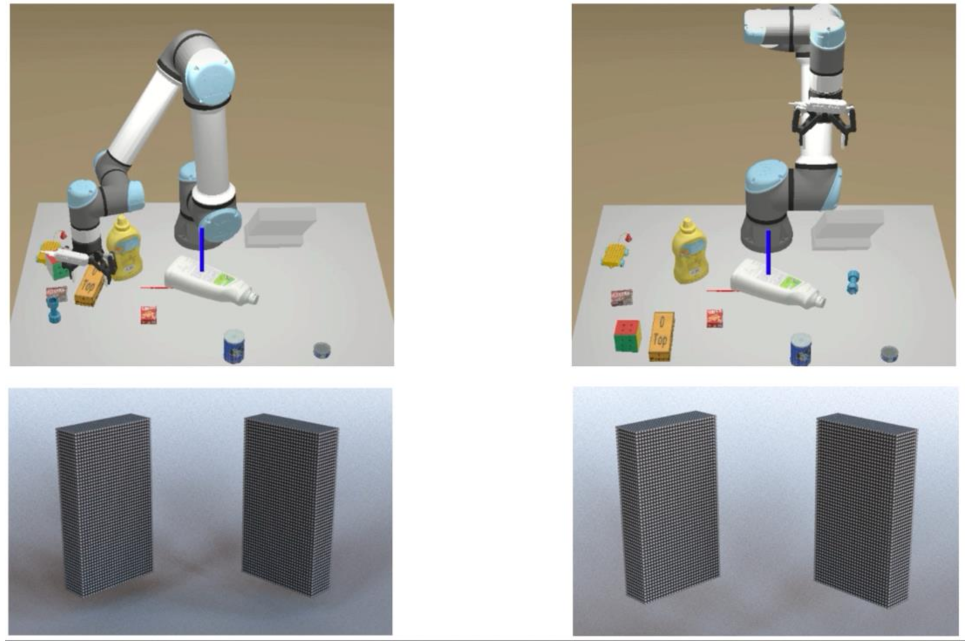
Unimodal Representation: Tactile Modality

The tactile sensor array (548 sensors) is assembled on a knitted glove uniformly distributed over the hand.



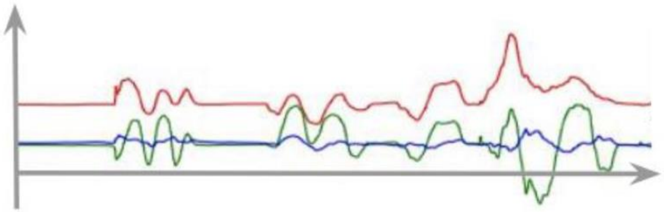
Sundaram et al., Learning the signatures of the human grasp using a scalable tactile glove. Nature 2019

Model the elastic property of the tactile sensor



Elastic Interaction of Particles for Robotic Tactile Simulation, ACMMM 2021

Unimodal Representation: Tactile Modality



Force-Torque Sensor

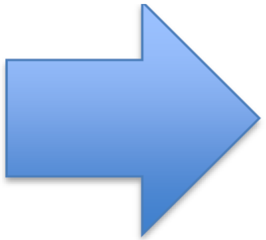
Time series data across six-axis Force-Torque sensor:
T × 6 signal.

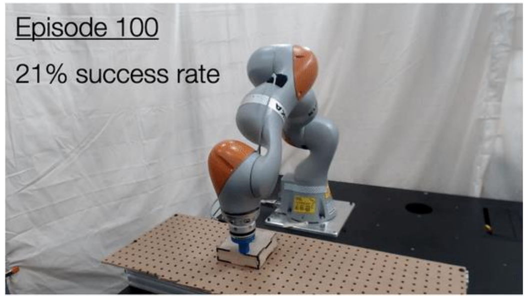
Proprioception

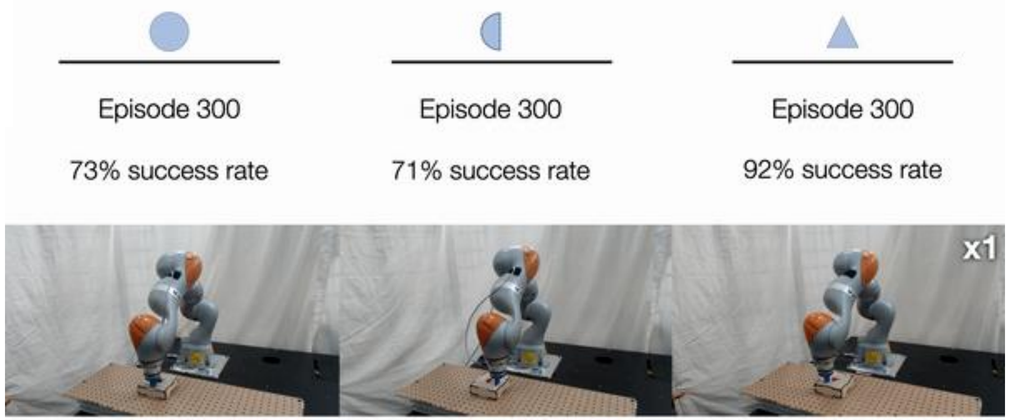
Measure values internal to the system (robot); e.g. motor speed, wheel load, **robot arm joint angles**, battery voltage.

Time series data across current position and velocity of the end-effector:
T × 2d signal.

Next action







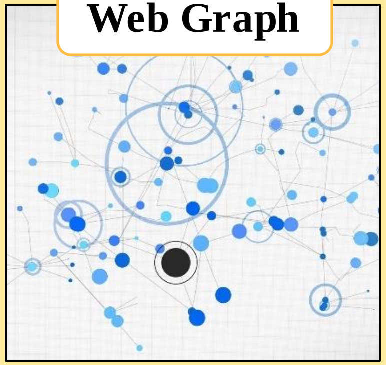
Lee et al., Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact Rich Tasks. ICR A 2019

Unimodal Representation: Tables

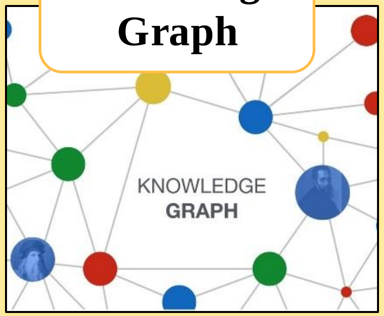
Bao et al., Table to Text: Describing Table Region with Natural Language. AAAI 2018

Unimodal Representation: Graphs

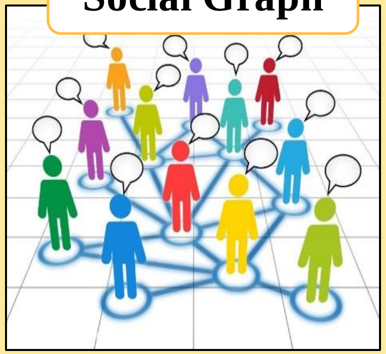
Web Graph



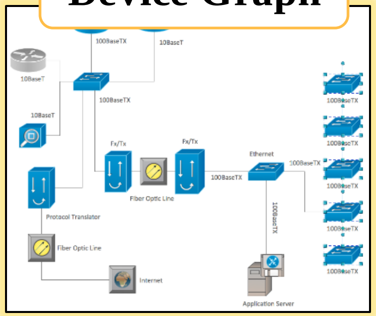
Knowledge Graph



Social Graph

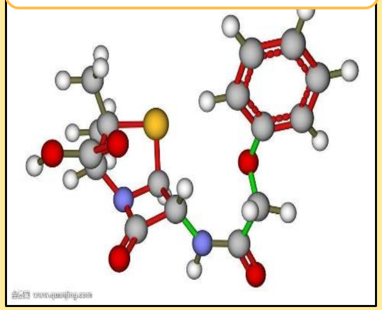


Device Graph

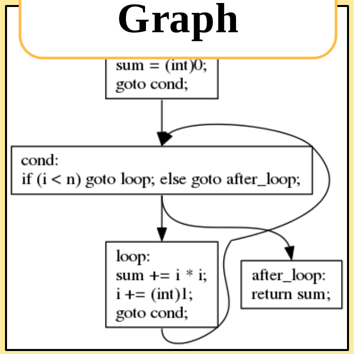


Graphs are Everywhere

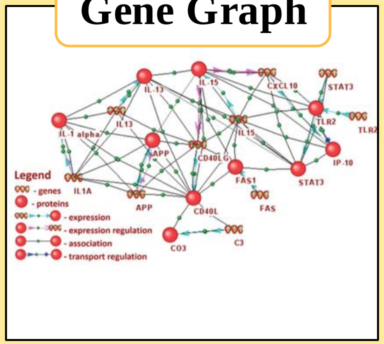
Molecular Graph



Control Flow Graph



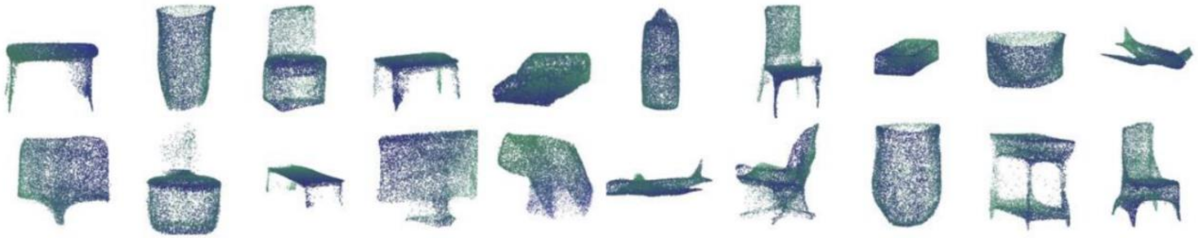
Gene Graph



Brain Graph

Unimodal Representation: Sets

Sets



Point clouds

Set anomaly detection
Set expansion
Set completion
Point cloud classification
Point cloud generation

Zaheer et al., DeepSets . NeurIPS 2017, Li et al., Point Cloud GAN. arxiv 2018

内容提纲

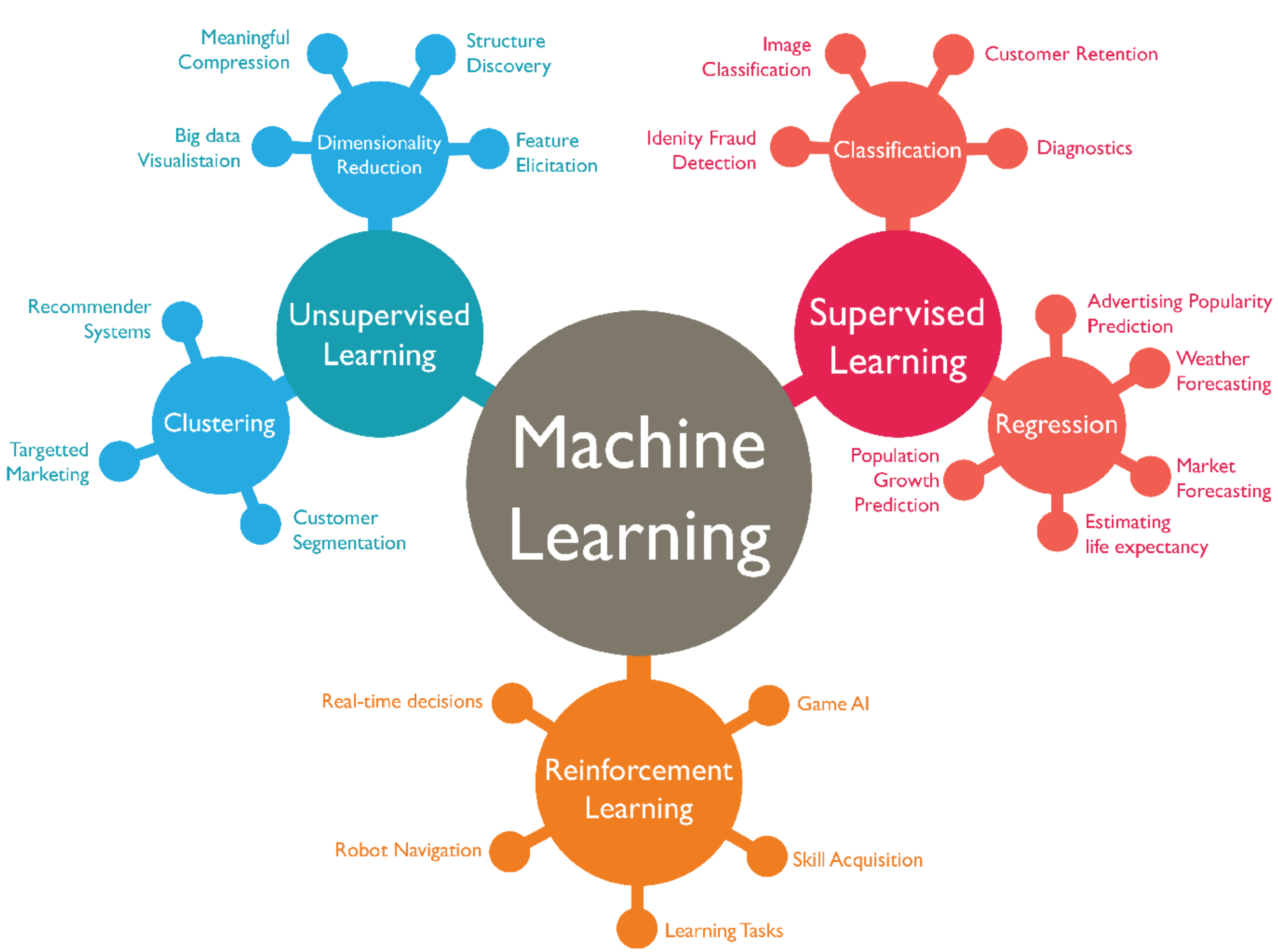
① 单模态基础表示

② 经典机器学习方法

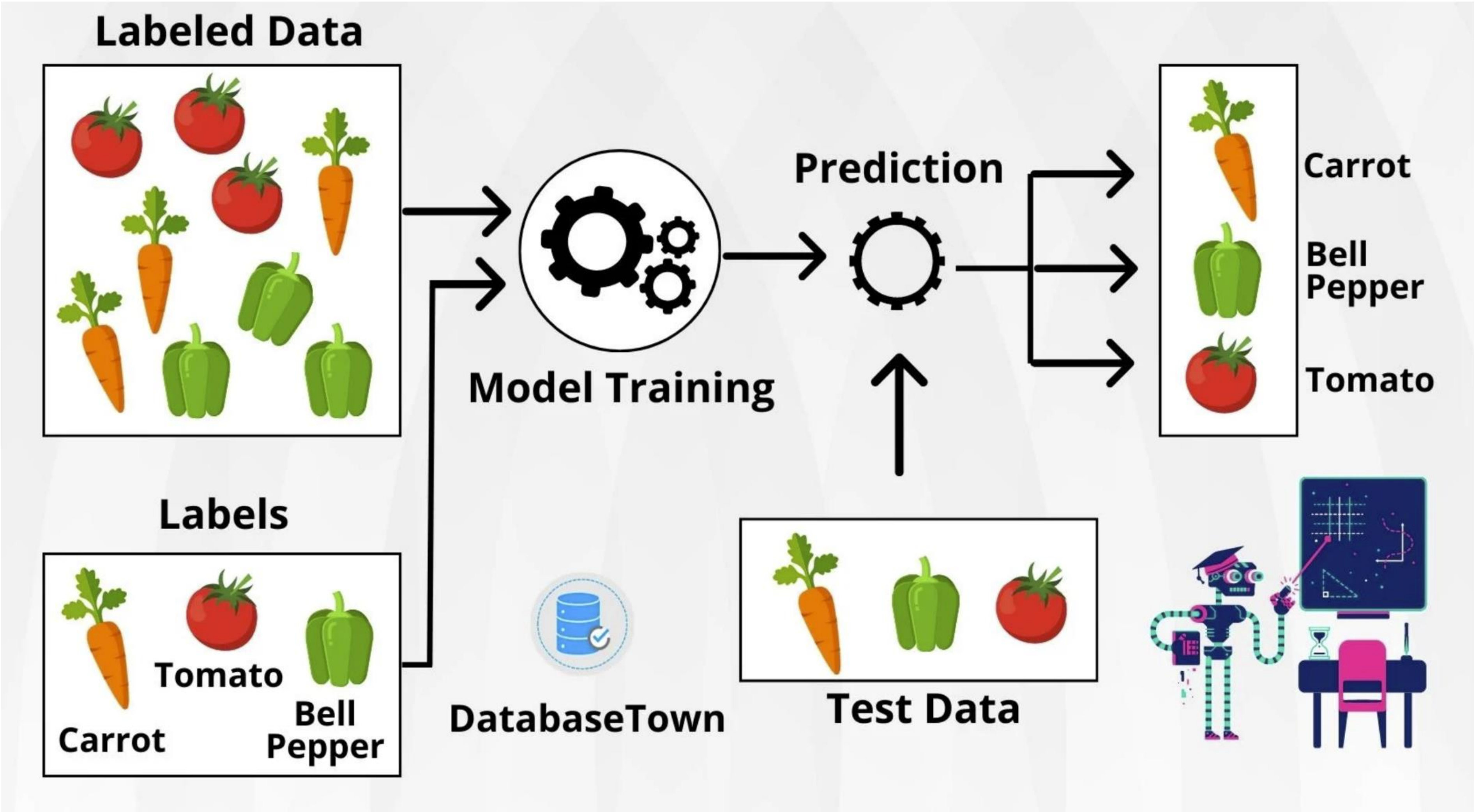
③ 神经网络

④ 优化方法简介

Machine Learning

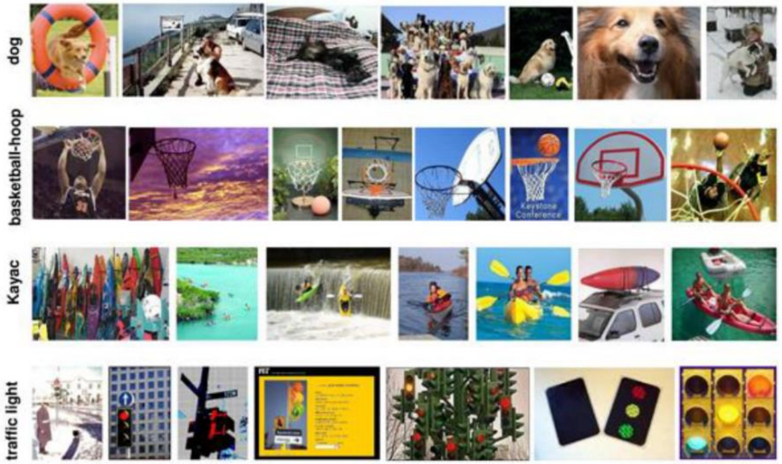


Supervised Learning



Simplest Classifier ?

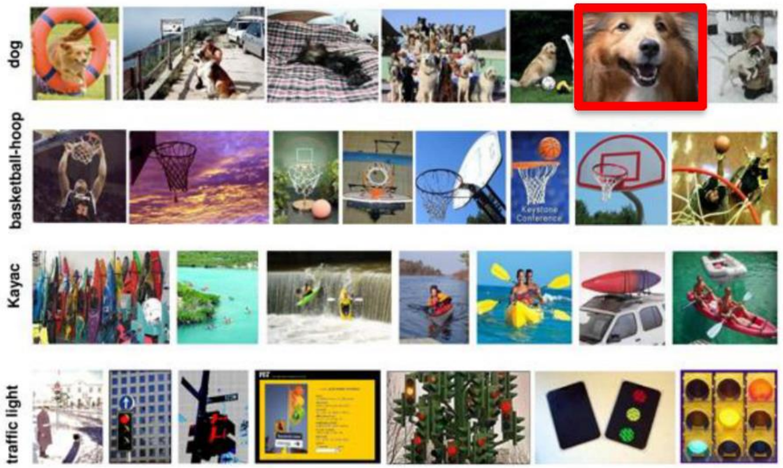
Traffic light
-or-
Dog
-or-
Basket
-or-
Kayak ?



Dataset

Simple Classifier: Nearest Neighbor

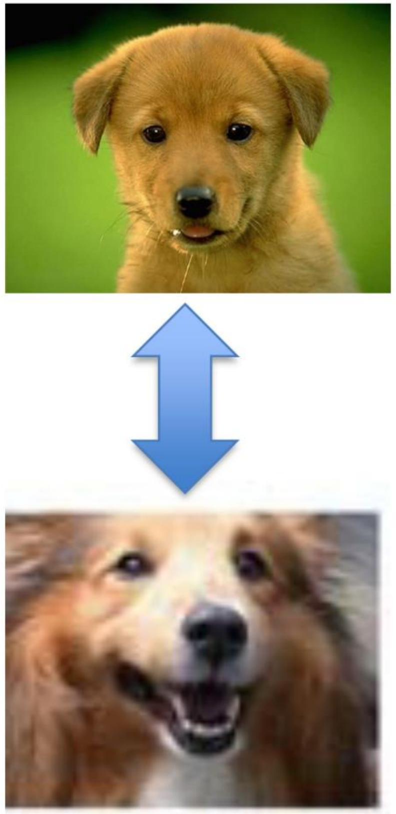
Traffic light
-or-
Dog
-or-
Basket
-or-
Kayak ?



Simple Classifier: Nearest Neighbor

- Non-parametric approaches—key ideas:
  - *“Let the data speak for themselves”*
  - *“Predict new cases based on similar cases”*
  - *“Use multiple local models instead of a single global model”*

Simple Classifier: Nearest Neighbor



Distance metrics

L1 (Manhattan) distance:

$$d_1(x_1, x_2) = \sum_j \left| x_1^j - x_2^j \right|$$

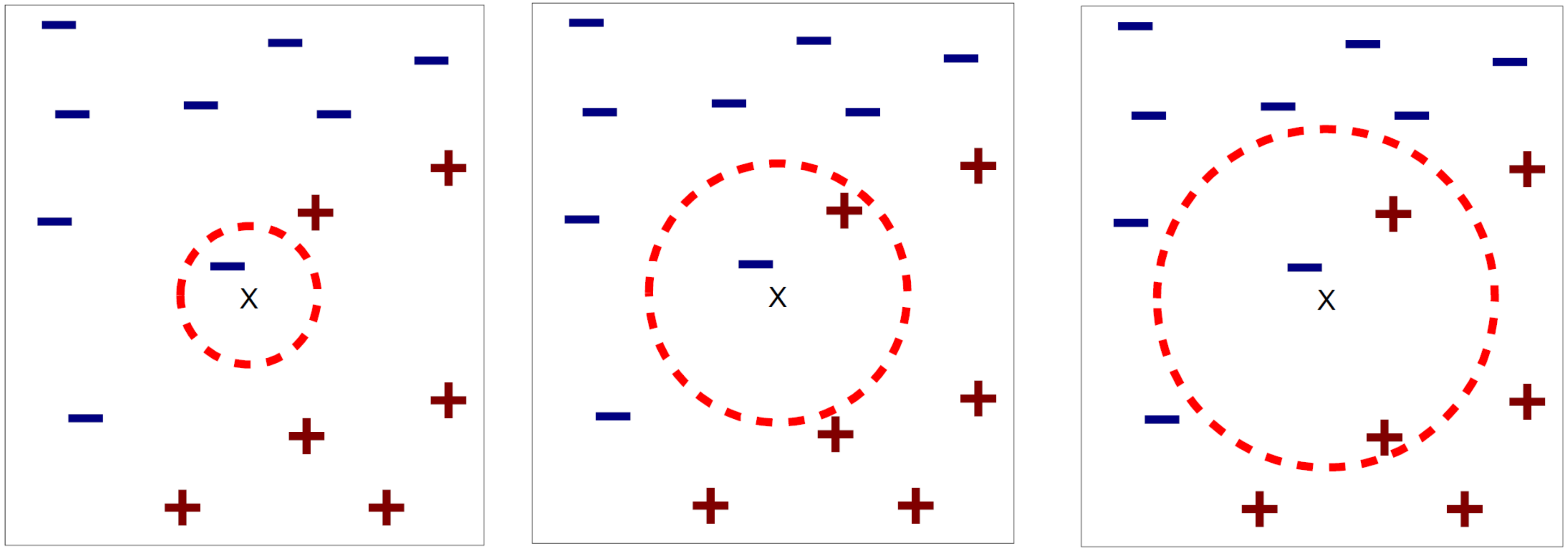
L2 (Eucledian) distance:

$$d_2(x_1, x_2) = \sqrt{\sum_j \left(x_1^j - x_2^j \right)^2}$$

Which distance metric to use?

First hyper-parameter!

Definition of K-Nearest Neighbor

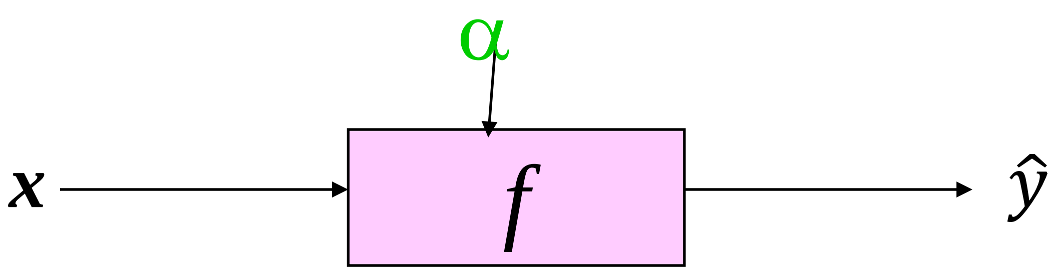


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

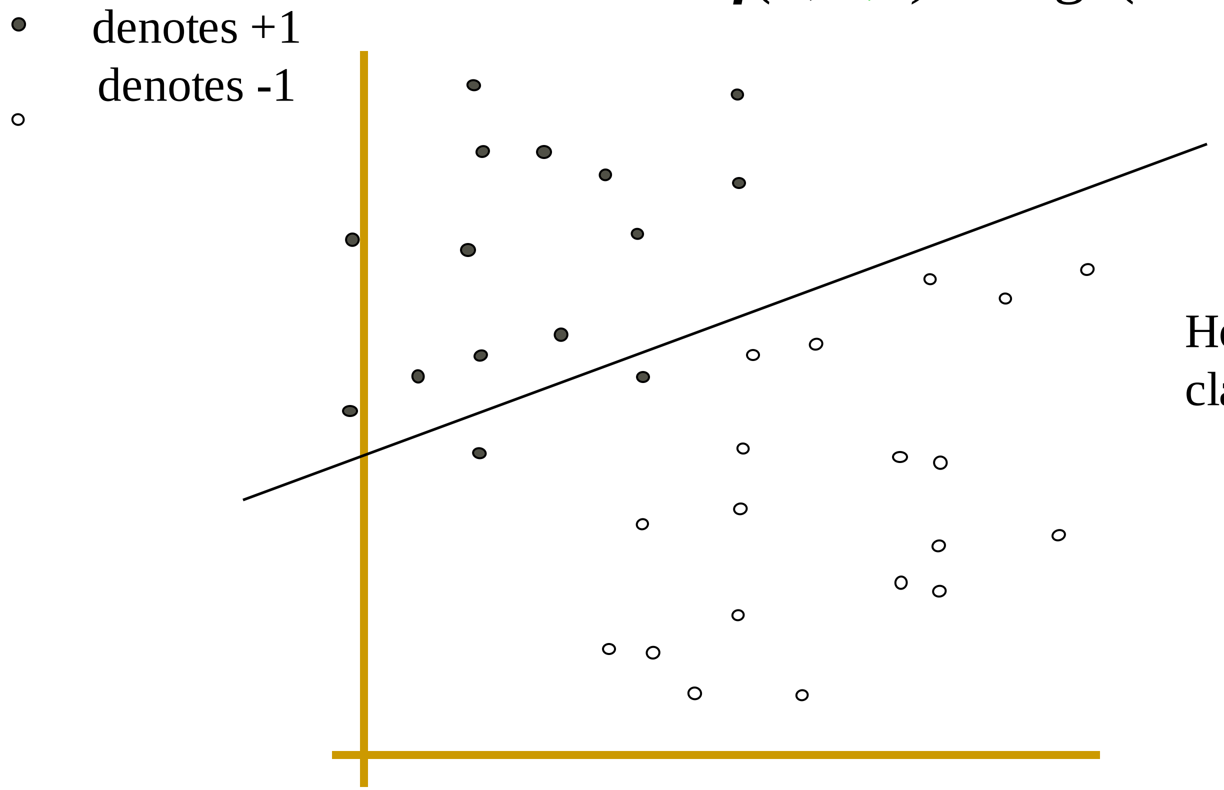
What value should we set K?

Second hyper-parameter!

Linear Classifier

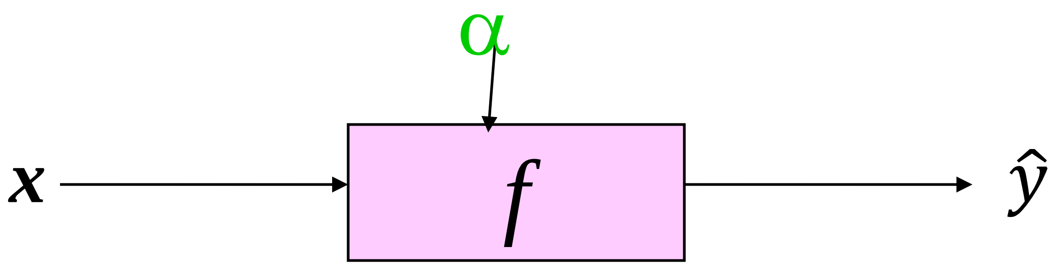


$f(\mathbf{x},\mathbf{w},b) = \text{sign}(\mathbf{w}\mathbf{x} - b)$



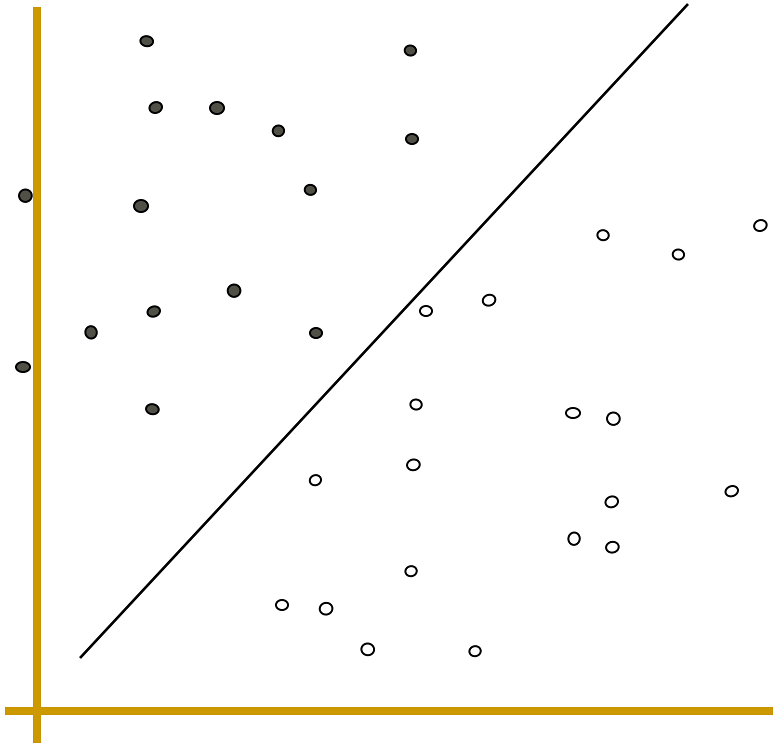
How would you classify this data?

Linear Classifier



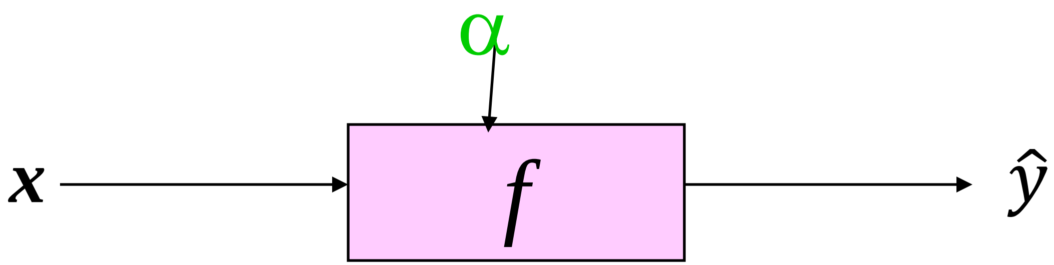
$$f(\mathbf{x},\mathbf{w},b) = \text{sign}(\mathbf{w}\mathbf{x} - b)$$

- ● denotes +1
- ○ denotes -1



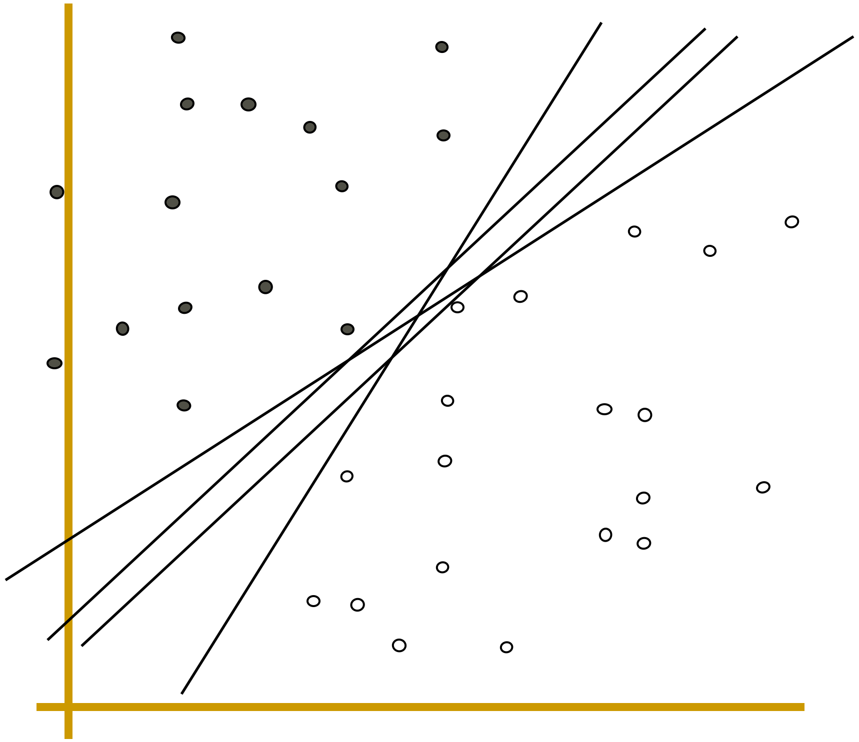
How would you classify this data?

Linear Classifier



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}\mathbf{x} - b)$$

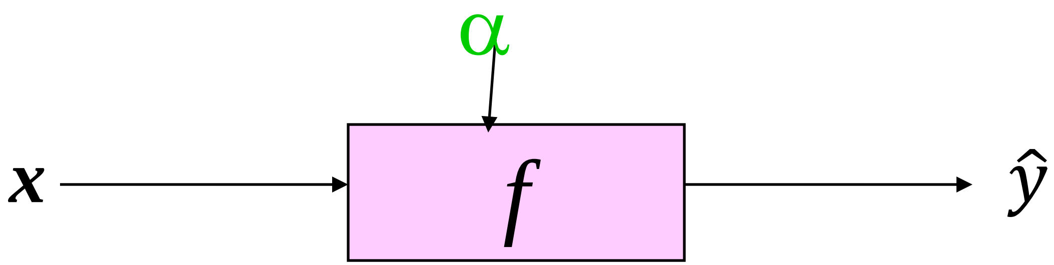
- denotes +1
- denotes -1



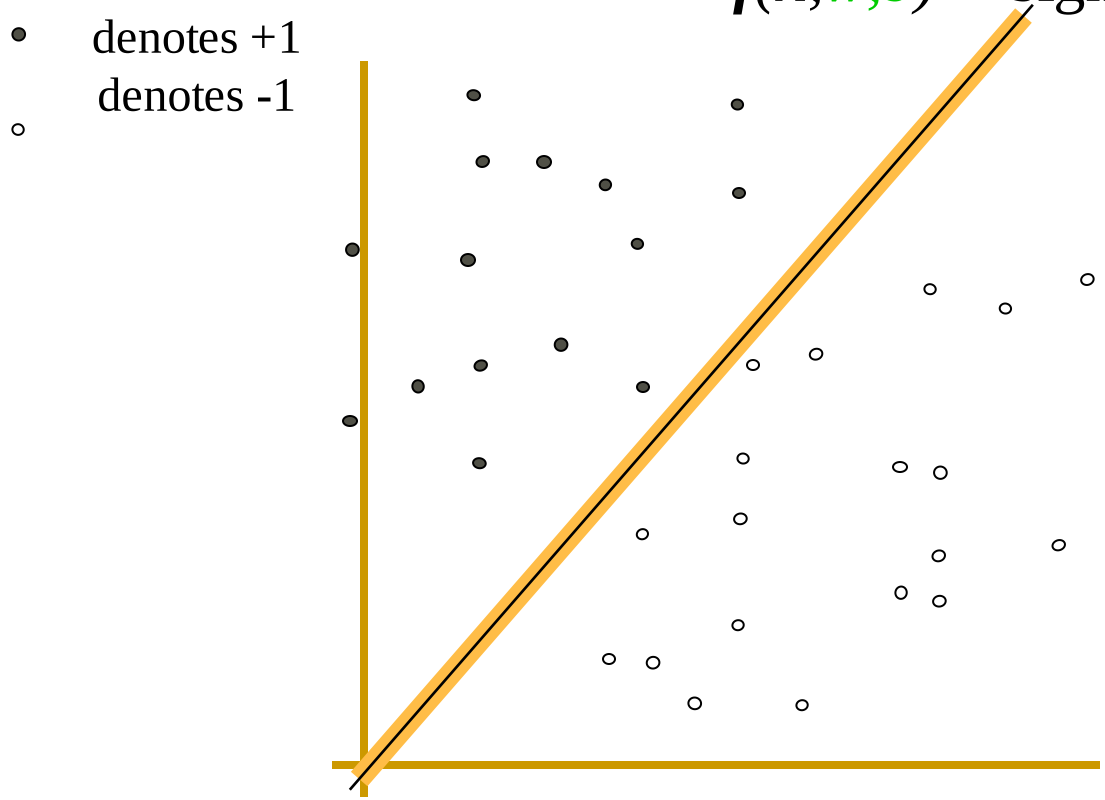
Any of these would be fine..

..but which is best?

Classifier Margin

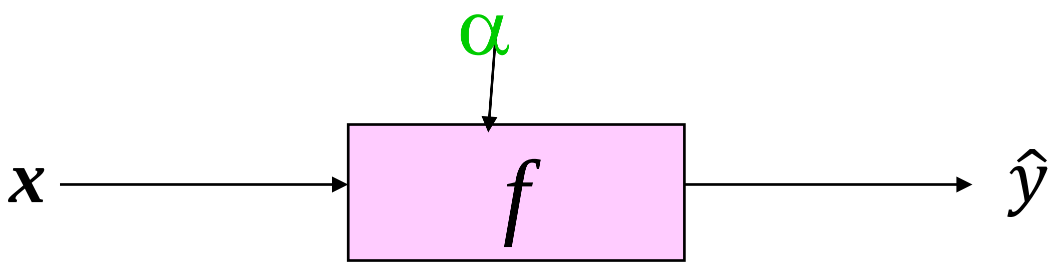


$f(\mathbf{x},\mathbf{w},b) = \text{sign}(\mathbf{w}\mathbf{x} - b)$



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

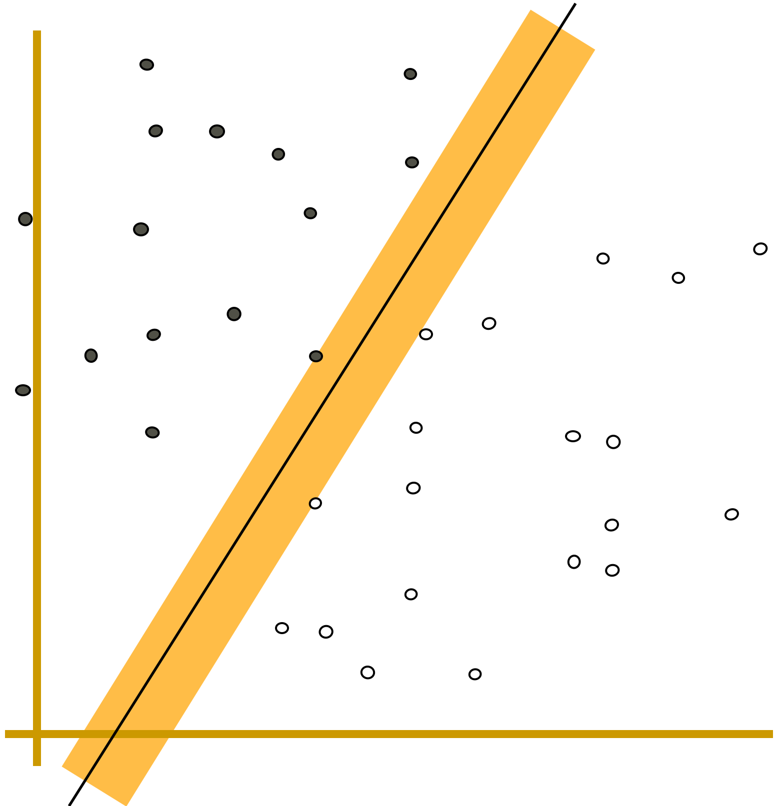
Support Vector Machine: Maximum Margin



$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}\mathbf{x} - b)$$

● denotes +1

○ denotes -1

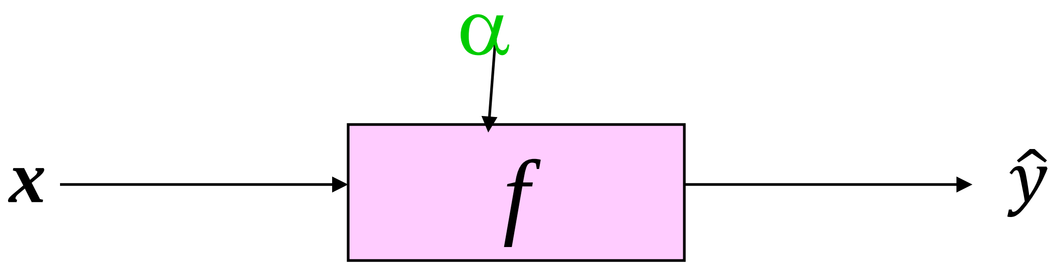


The maximum margin linear classifier is the linear classifier with the maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Linear SVM

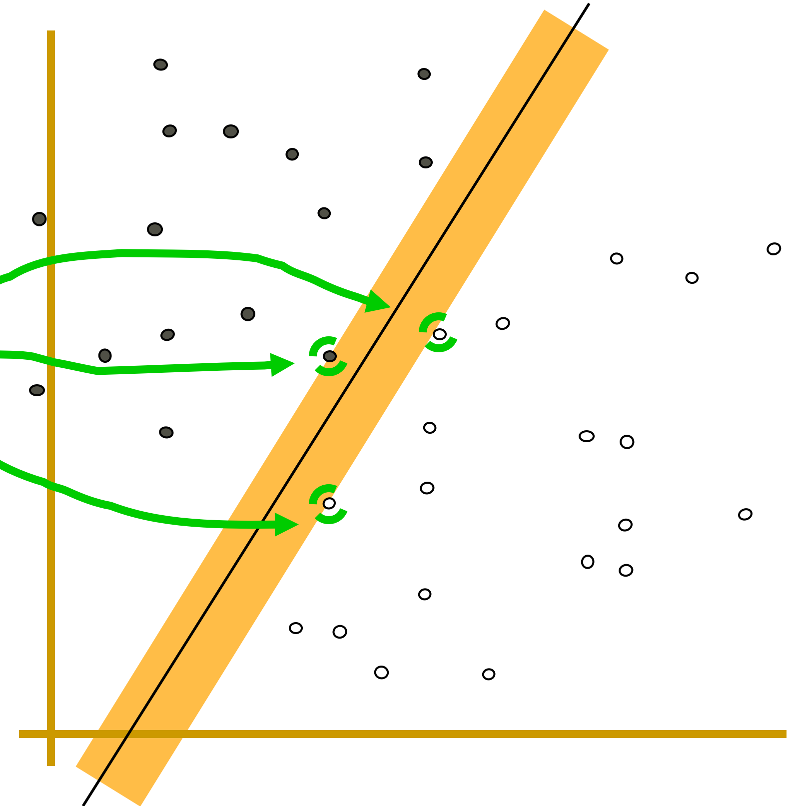
Support Vector Machine: Maximum Margin



$$f(\mathbf{x},\mathbf{w},b) = \text{sign}(\mathbf{w}\mathbf{x} - b)$$

● denotes +1
○ denotes -1

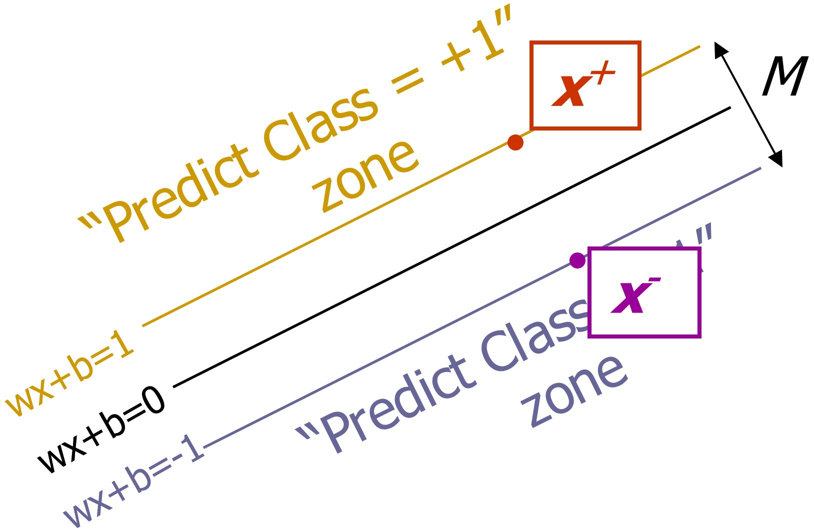
Support Vectors are those datapoints that the margin pushes up against



The maximum margin linear classifier is the linear classifier with the maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Linear SVM

Support Vector Machine: Maximum Margin



M = Margin Width = $\frac{2}{\sqrt{\mathbf{w}.\mathbf{w}}}$

- Plus-plane = $\{ \mathbf{x} : \mathbf{w} . \mathbf{x} + b = +1 \}$
- Minus-plane = $\{ \mathbf{x} : \mathbf{w} . \mathbf{x} + b = -1 \}$

Classify as.. +1 if $\mathbf{w} . \mathbf{x} + b >= 1$

-1 if $\mathbf{w} . \mathbf{x} + b <= -1$

Going beyond Classifiers…

Representation Learning: A Review and New Perspectives

Yoshua Bengio†, Aaron Courville, and Pascal Vincent†
Department of computer science and operations research, U. Montreal
† also, Canadian Institute for Advanced Research (CIFAR)

Data-driven feature representation learning: **Deep neural networks**

内容提纲

① 单模态基础表示

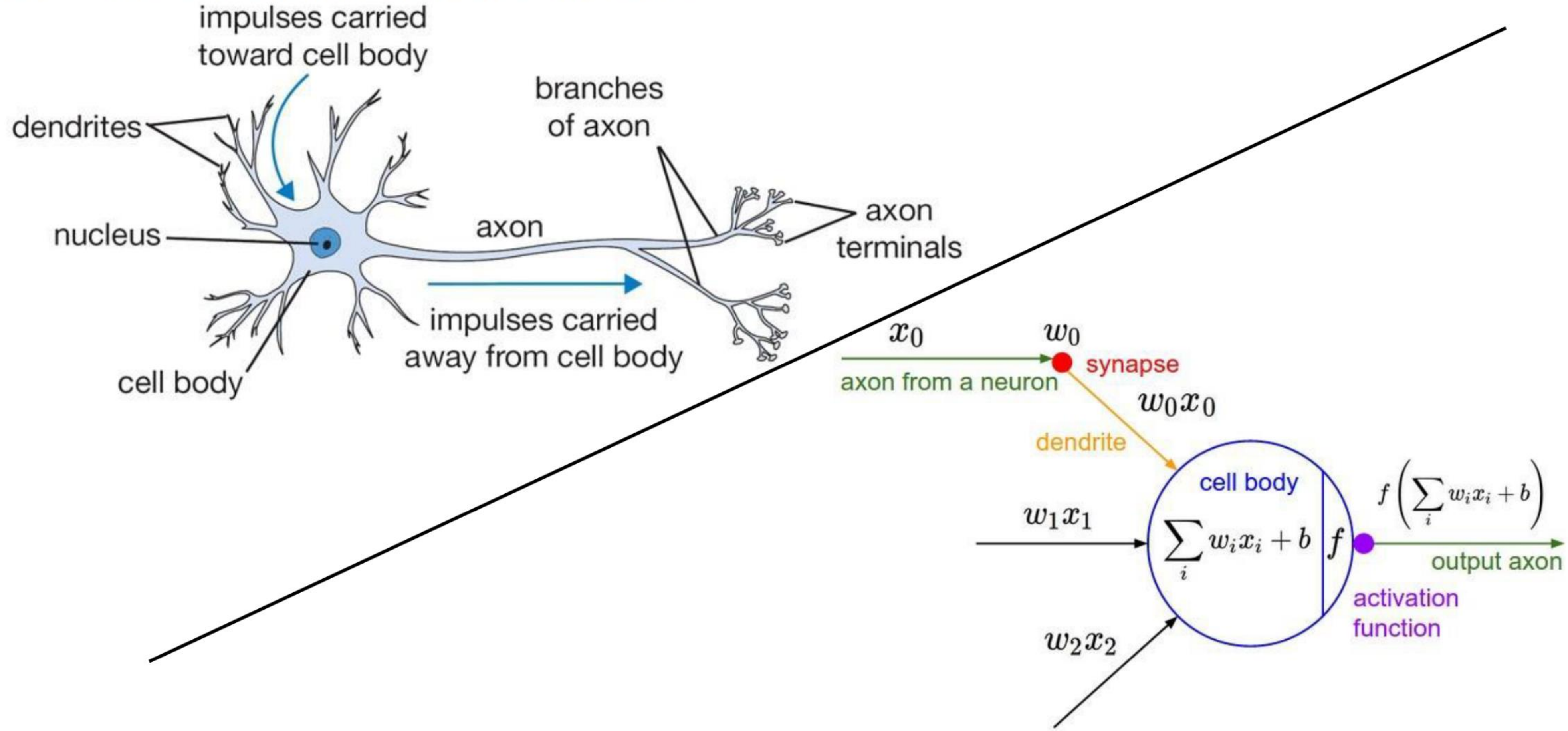
② 经典机器学习方法

③ 神经网络

④ 优化方法简介

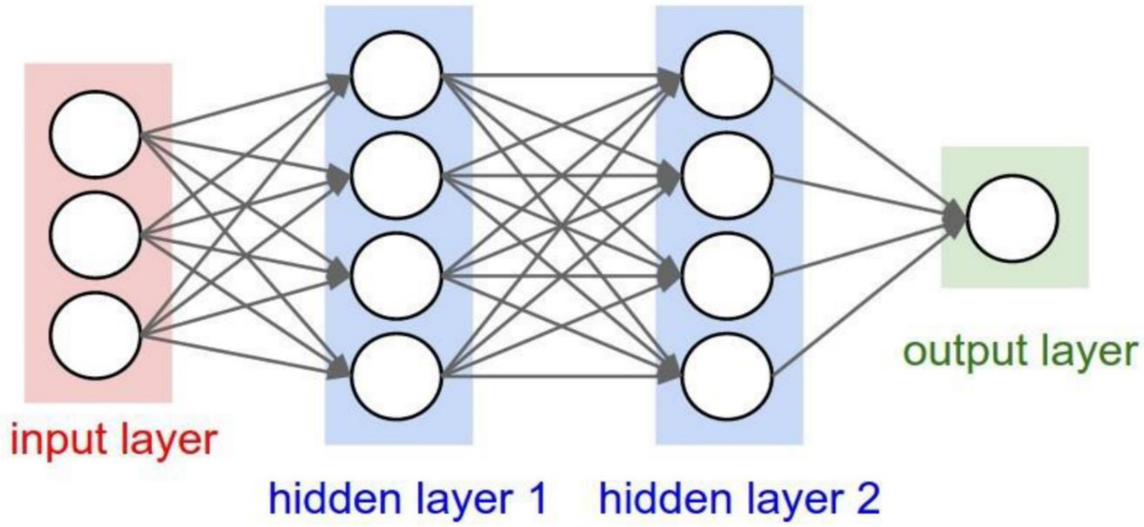
Neural Networks: inspiration

- Made up of artificial neurons

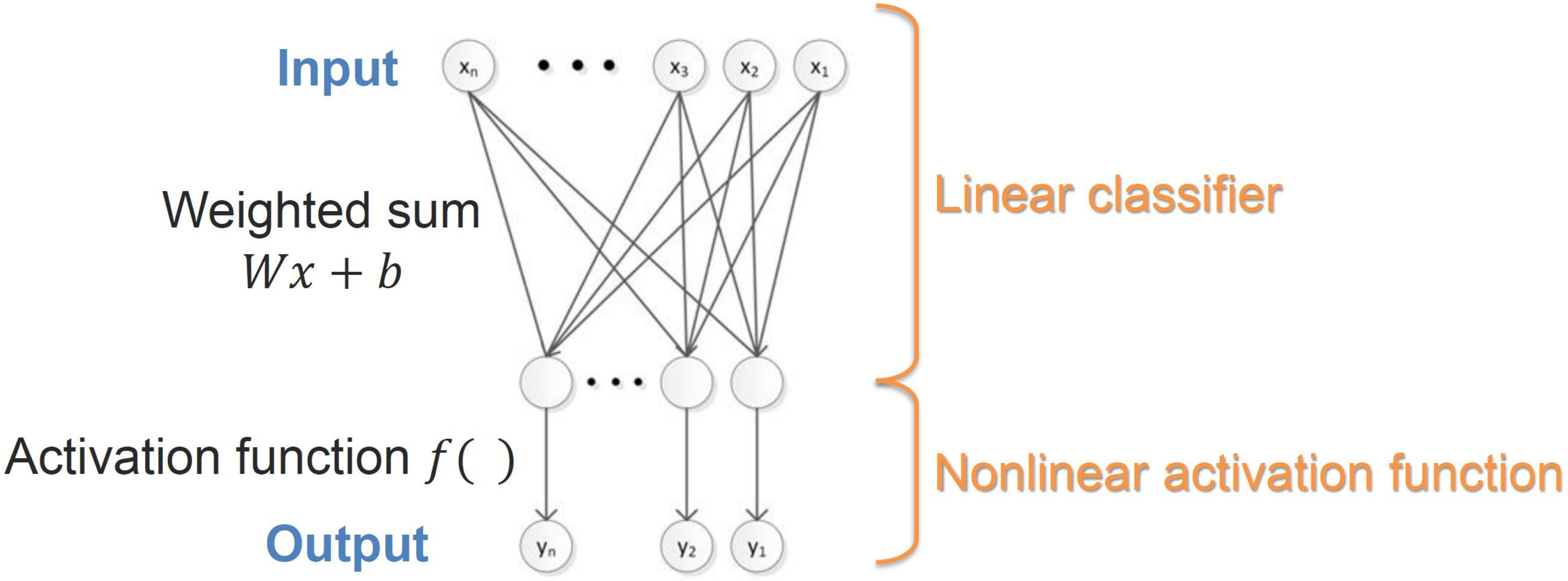


Neural Networks: score function

- Made up of artificial neurons
  - Linear function (dot product) followed by a nonlinear activation function
- Example a Multi Layer Perceptron



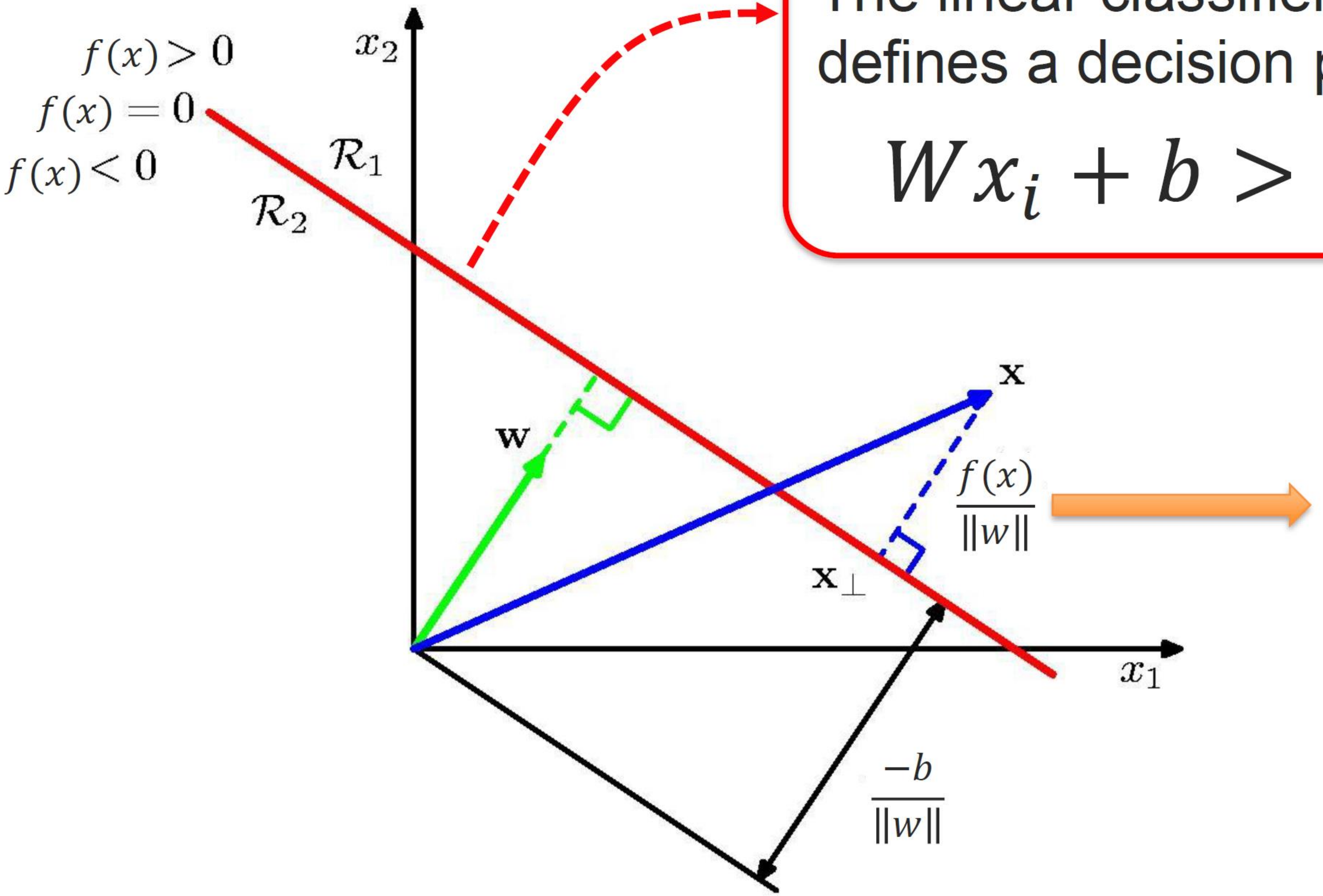
Basic Neural Network Building Block



$$\hat{y} = f(Wx + b)$$

Interpreting a Linear Classifier

$$f(x_i; W, b) = Wx_i + b$$



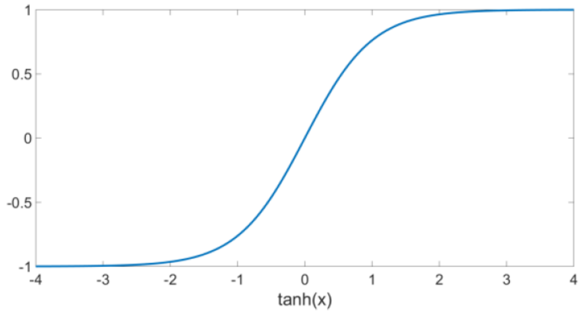
The linear classifier defines a decision plane:

$$Wx_i + b > 0$$

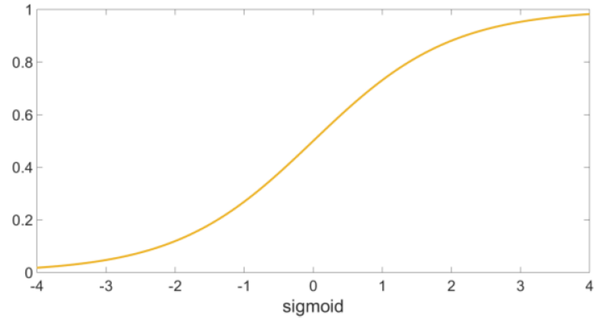
Output can be seen as distance to decision plane

Neural Networks: activation function

- $f(x) = \tanh(x)$

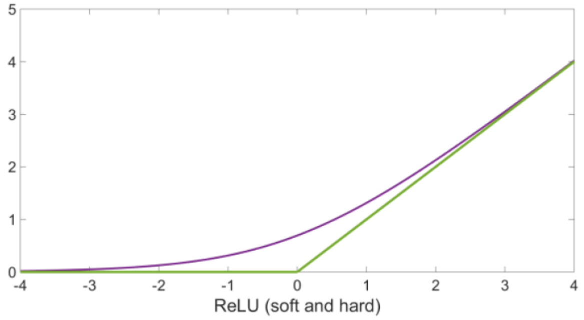


- Sigmoid - $f(x) = (1 + e^{-x})^{-1}$



- Linear – $f(x) = ax + b$

- ReLU $f(x) = \max(0, x) \sim \log(1 + \exp(x)\ )$
    - Rectifier Linear Units
    - Faster training - no gradient vanishing
    - Induces sparsity



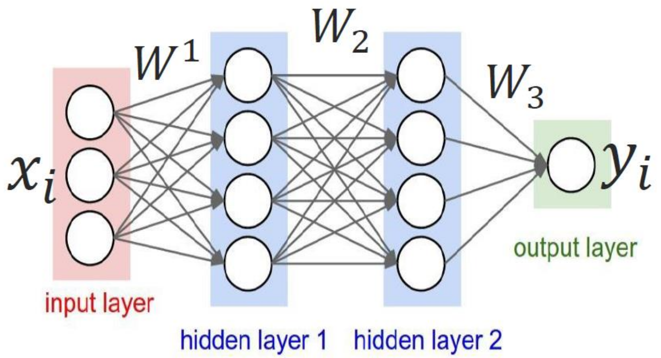
Multi-Layer Perceptron (MLP)

Activation functions (individual layers)

$$f_{1;W_1}(x) = \sigma(W_1 x + b_1)$$

$$f_{2;W_2}(x) = \sigma(W_2 x + b_2)$$

$$f_{3;W_3}(x) = \sigma(W_3 x + b_3)$$



Score function

$$y_i = f(x_i) = f_{3;W_3}(f_{2;W_2}(f_{1;W_1}(x_i)))$$

How to integrate all the output scores?

Neural Network: loss function

(or cost function or objective)

Scores $f(x_i; W)$ — **Label** $y_i = 2\ (dog)$ → **Loss** $L_i = ?$

Image x_i

(Size: 32*32*3)

0 (duck) ?	-12.3
1 (cat) ?	45.6
2 (dog) ?	98.7
3 (pig) ?	12.2
4 (bird) ?	-45.3

Multi-class problem

How to assign only one number representing how "unhappy" we are about these scores?

The loss function quantifies the amount by which the prediction scores deviate from the actual values.

First Loss Function: Cross Entropy Loss

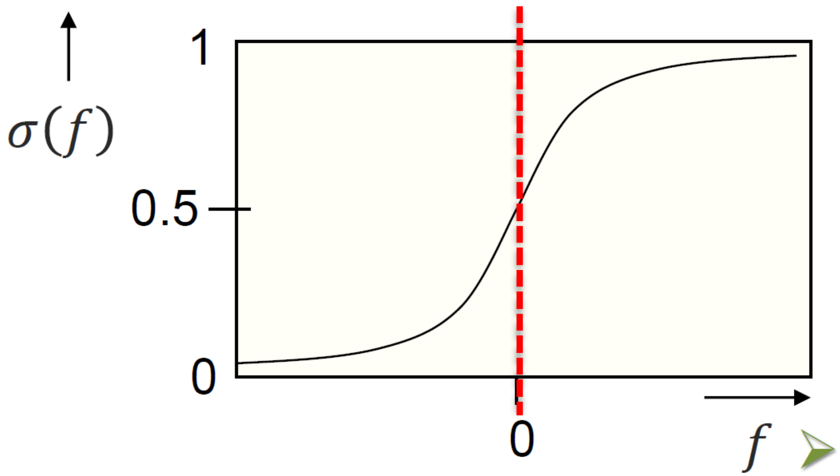
(or logistic loss)

Logistic function: $$\sigma(f) = \frac{1}{1 + e^{-f}}$$

Logistic regression: (two classes)

$$p(y_i = \text{"}dog\text{"} | x_i; w) \;\; = \sigma(w^T x_i)$$

= true
for two-class problem



➤ Score function

First Loss Function: Cross Entropy Loss

(or logistic loss)

Logistic function: $$\sigma(f) = \frac{1}{1+e^{-f}}$$

Logistic regression: (two classes) $$p(y_i = \text{"dog"} | x_i; w) \;= \sigma(w^T x_i)$$

= true
for two-class problem

Softmax function: (multiple classes) $$p(y_i | x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

Cross-entropy loss: $$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Softmax function

Minimizing the negative log likelihood.

Second Loss Function: Hinge Loss

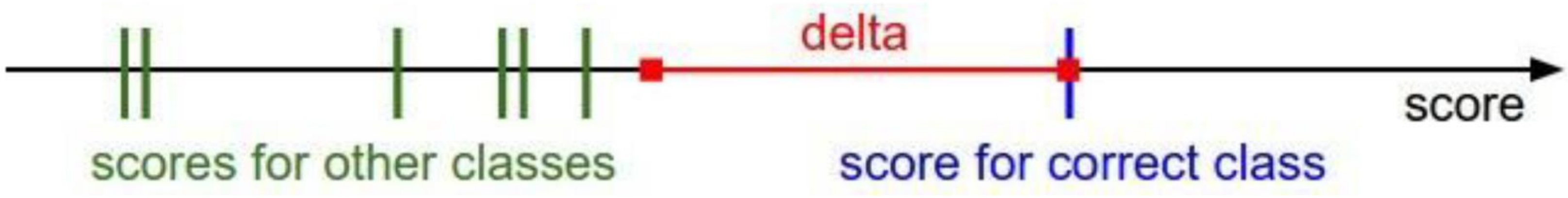
(or max-margin loss or Multi-class SVM loss)

$$L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i} + \Delta)$$

loss due to example i

sum over all incorrect labels

difference between the correct class score and incorrect class score



Different concepts

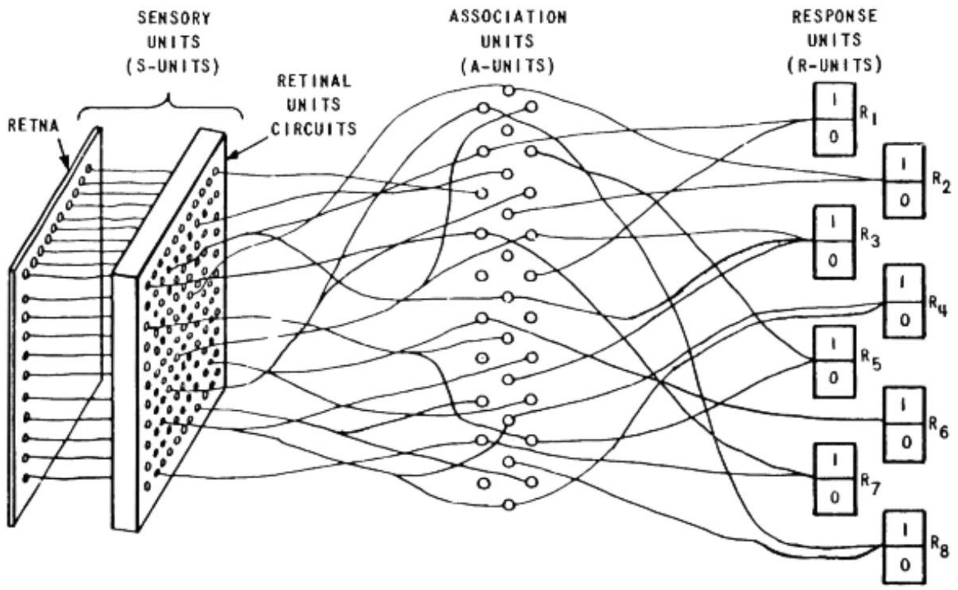
Sigmoid	Relu	Logistic Function	Softmax Function
$\sigma(x) \coloneqq \frac{1}{(1+e^{-x})}$	$\sigma(x) \coloneqq \max(0, x)$	$\sigma(\boldsymbol{x}) \coloneqq \frac{1}{\left(1+e^{-(\boldsymbol{w}^\top \boldsymbol{x}+b)}\right)}$ (Or Linear Function + Sigmoid)	$\sigma(\boldsymbol{x})_y \coloneqq \frac{e^{x_y}}{\sum_j e^{x_j}}$

Cross-Entropy Loss	Hinge Loss
$L(\boldsymbol{p}, y) \coloneqq -\log \hat{p}_y$	$L(\boldsymbol{x}, y) \coloneqq \sum_{j \neq y} \max(0, x_j - x_y + \Delta)$

Linear Classifier (Perceptron)	Logistic Regression	MLP
$\text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$	Logistic Function + Cross-Entropy Loss	(Linear Function + Activation) $\times L$ + Softmax + Cross-Entropy Loss

Brief History of Deep Learning

Early neural networks



Perceptron, one of the first neural network architectures

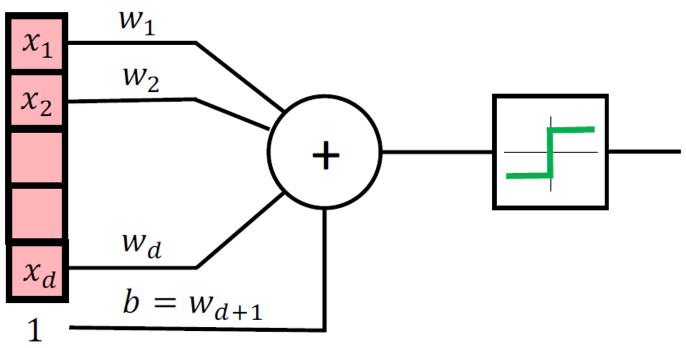
F. Rosenblatt

1957

Rosenblatt 1957

Brief History of Deep Learning

"Simple perceptron"

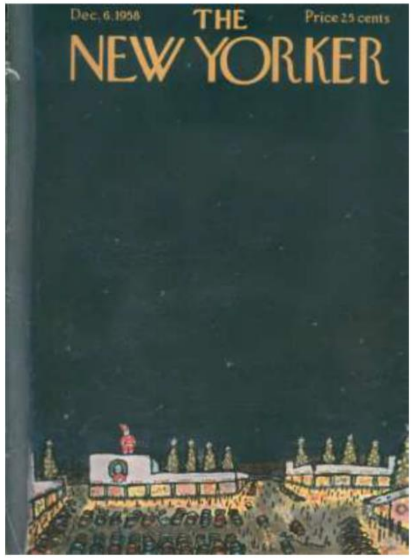


$$y = \operatorname{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

Brief History of Deep Learning

Early hype

> "First serious rival to the human brain even devised."
>
> "Remarkable machine capable of what amounts to thought"
>
> — The New Yorker

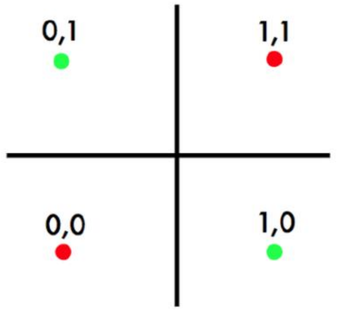


1958

Manson, Stewart, Gill 1958

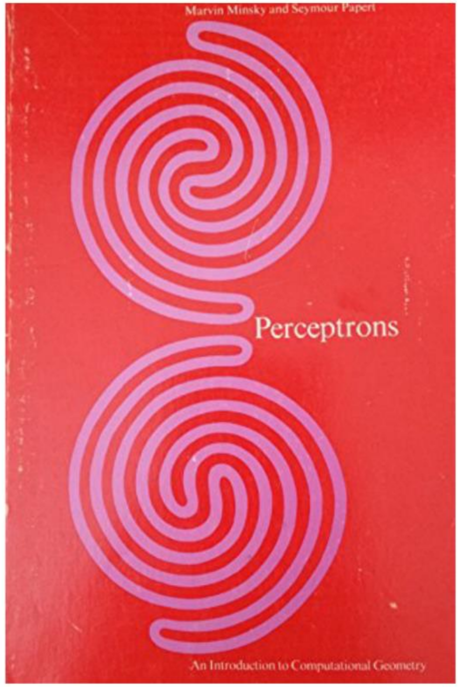
Brief History of Deep Learning

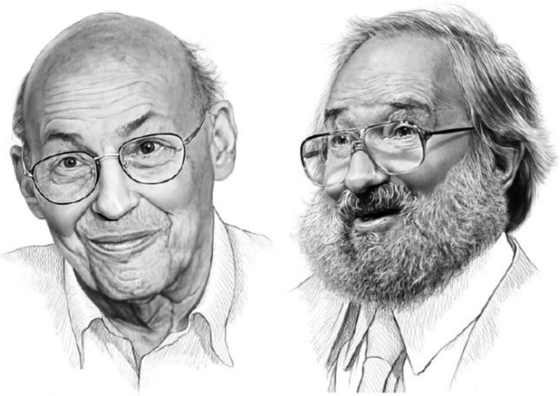
The "XOR Affair"



XOR

"[simple] perceptron cannot represent even the XOR function"





M. Minsky **S. Papert**

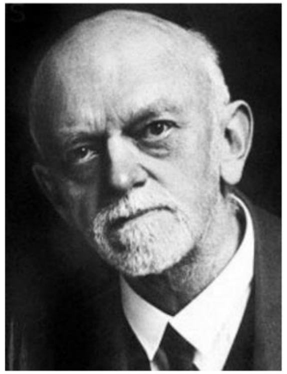
1969

Minsky, Papert 1969

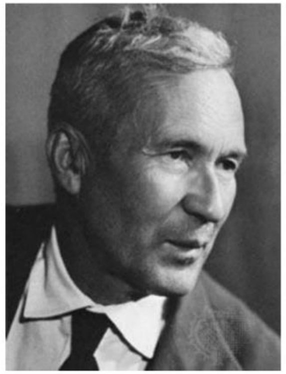
Brief History of Deep Learning

Brief History of Deep Learning

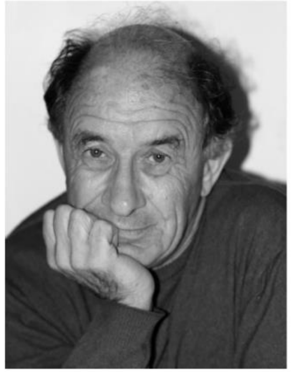
Universal approximation



D. Hilbert



A. Kolmogorov



V. Arnold

G. Cybenko

K. Hornik

13th Problem

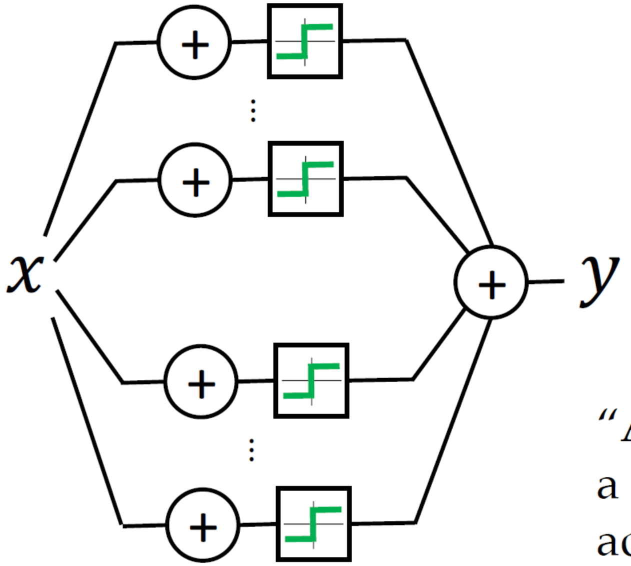
$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

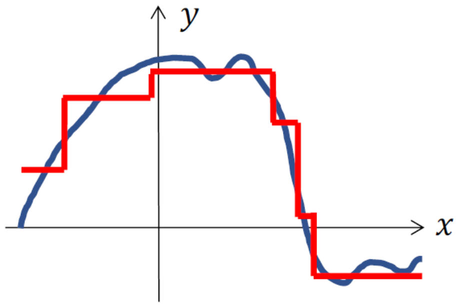
Results specific to multilayer neural networks

Hilbert 1900; Arnold 1956; Kolmogorov 1957; Cybenko 1989; Hornik 1991

Brief History of Deep Learning

Universal approximation





"A 2-layer perceptron can approximate a continuous function to any desired accuracy"

Cybenko 1989; Hornik 1991; Barron 1993; Leshno et al 1993; Maiorov 1999; Pinkus 1999

内容提纲

Learning Model Parameters

We have our training data

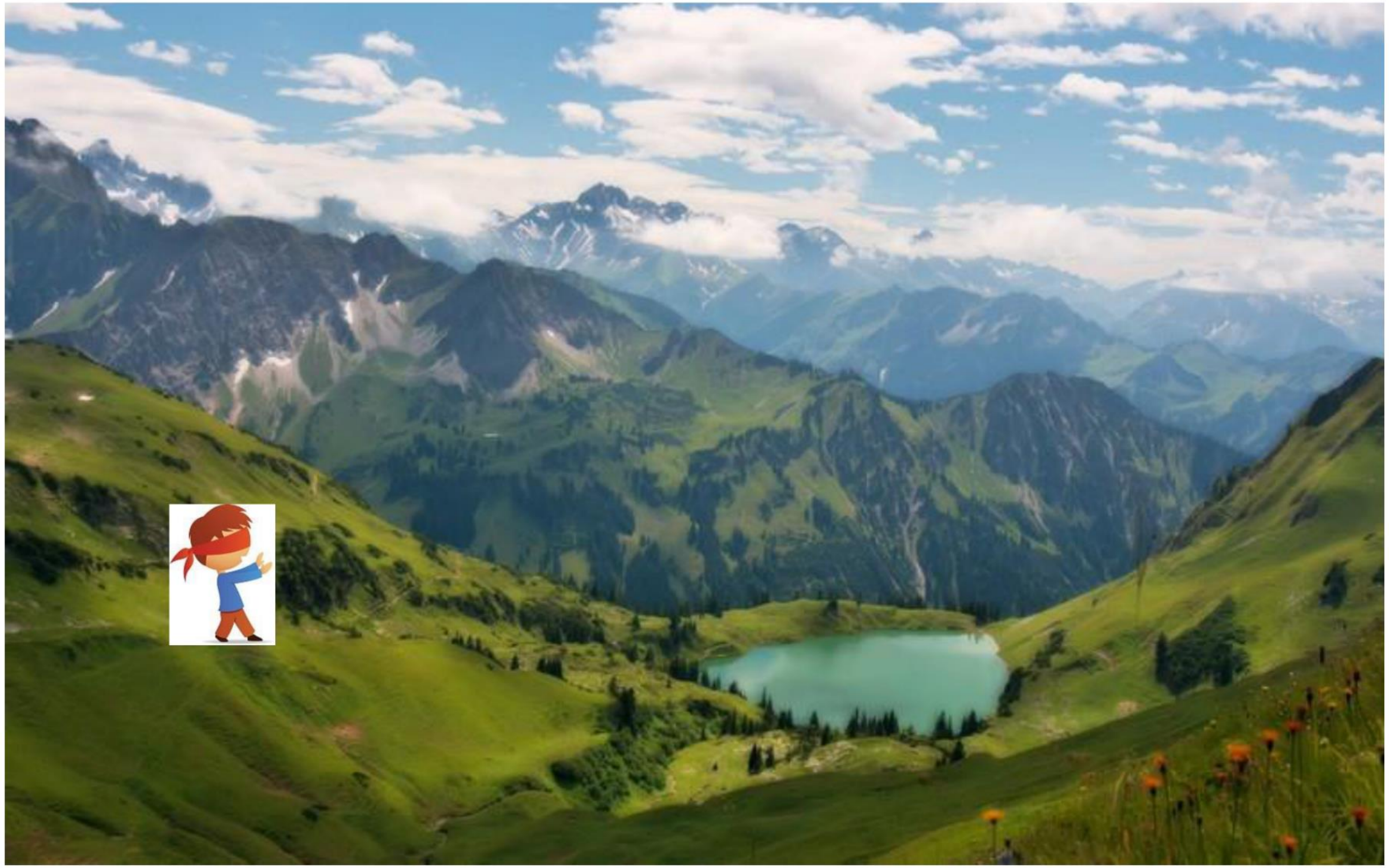
- $\mathbb{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$ (e.g. images, videos, text etc.)
- $\mathbb{Y} = \{y_1, y_2, \dots, y_n\}$ (labels)

We want to learn the W (weights and biases) that leads to best loss

$$\underset{W}{\operatorname{argmin}}[L(\mathbb{X}, \mathbb{Y}, W)]$$

The notation means find W for which $L(\mathbb{X}, \mathbb{Y}, W)$ has the lowest value

Optimization



Analytical Gradient

If we know the function and it is **differentiable**

- Derivative/gradient is defined at every point in *f*
- Sometimes use differentiable approximations
- Some are locally differentiable

Examples:

$$f(x) = \frac{1}{1+e^{-x}}; \frac{df}{dx} = (1-f(x))f(x)$$

$$f(x) = (x-y)^2; \frac{df}{dx} = 2(x-y)$$

How to follow the gradient

Many methods for optimization

- **Gradient Descent (actually the “simplest” one)**
- Newton methods (use Hessian – second derivative)
- Quasi-Newton (use approximate Hessian)
  - BFGS
  - LBFGS
  - Don’t require learning rates (fewer hyperparameters)
  - But, do not work with stochastic and batch methods so rarely used to train modern Neural Networks

All of them look at the gradient

- Very few non gradient based optimization methods

Parameter Update Strategies

Gradient descent:

$$\theta^{(t+1)} = \theta^{t} - \epsilon_k \nabla_\theta L$$

New model parameters

Previous parameters

Learning rate at iteration *k*

Gradient of our loss function

$$\epsilon_k = (1-\alpha)\epsilon_0 + \alpha\epsilon_\tau$$

Learning rate at iteration *k*

Decay

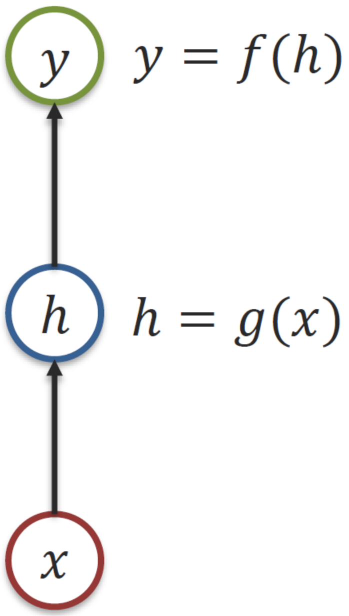
Initial learning rate

Decay learning rate linearly until iteration τ

Gradient Computation

Chain rule:

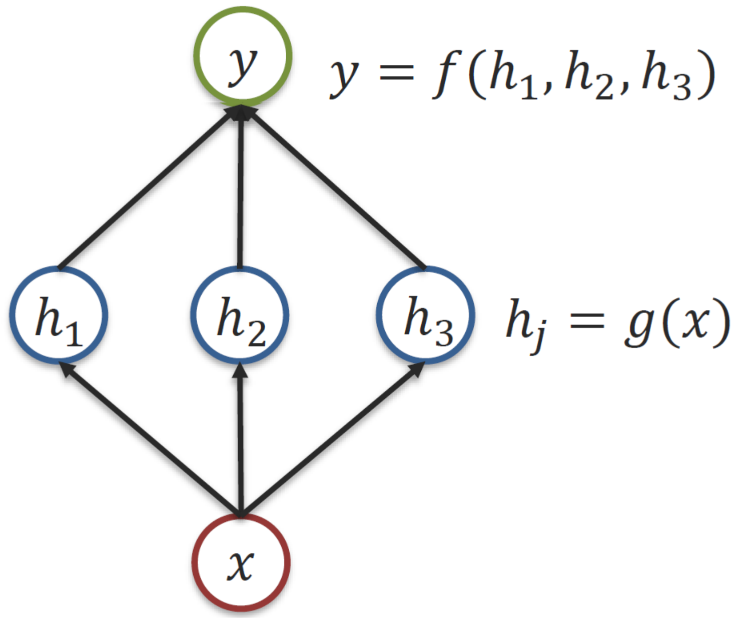
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial h}\frac{\partial h}{\partial x}$$



Gradient Computation

Multiple-path chain rule:

$$\frac{\partial y}{\partial x} = \sum_j \frac{\partial y}{\partial h_j} \frac{\partial h_j}{\partial x}$$



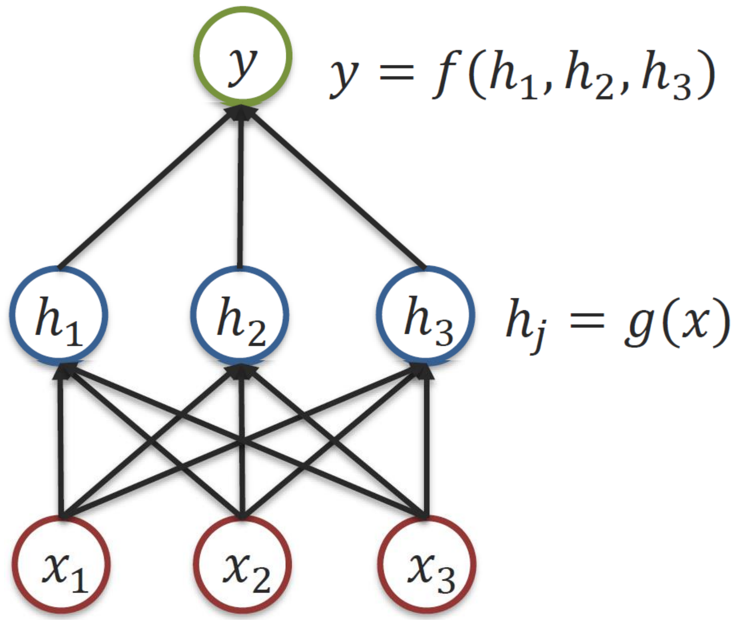
Gradient Computation

Multiple-path chain rule:

$$\frac{\partial y}{\partial x_1} = \sum_j \frac{\partial y}{\partial h_j} \frac{\partial h_j}{\partial x_1}$$

$$\frac{\partial y}{\partial x_2} = \sum_j \frac{\partial y}{\partial h_j} \frac{\partial h_j}{\partial x_2}$$

$$\frac{\partial y}{\partial x_3} = \sum_j \frac{\partial y}{\partial h_j} \frac{\partial h_j}{\partial x_3}$$



Gradient Computation

Vector representation:

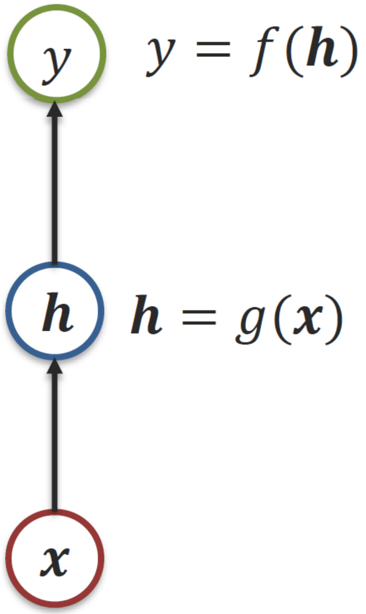
$$\nabla_{\boldsymbol{x}}\, y = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \frac{\partial y}{\partial x_3}\right]$$

Gradient

$$\nabla_{\boldsymbol{x}}\, y = \left(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right)^T \nabla_{\boldsymbol{h}}\, y$$

"local" Jacobian
(matrix of size $|h| \times |x|$ computed using partial derivatives)

"backprop" Gradient



Back-Propagation Algorithm (efficient gradient)

Forward pass

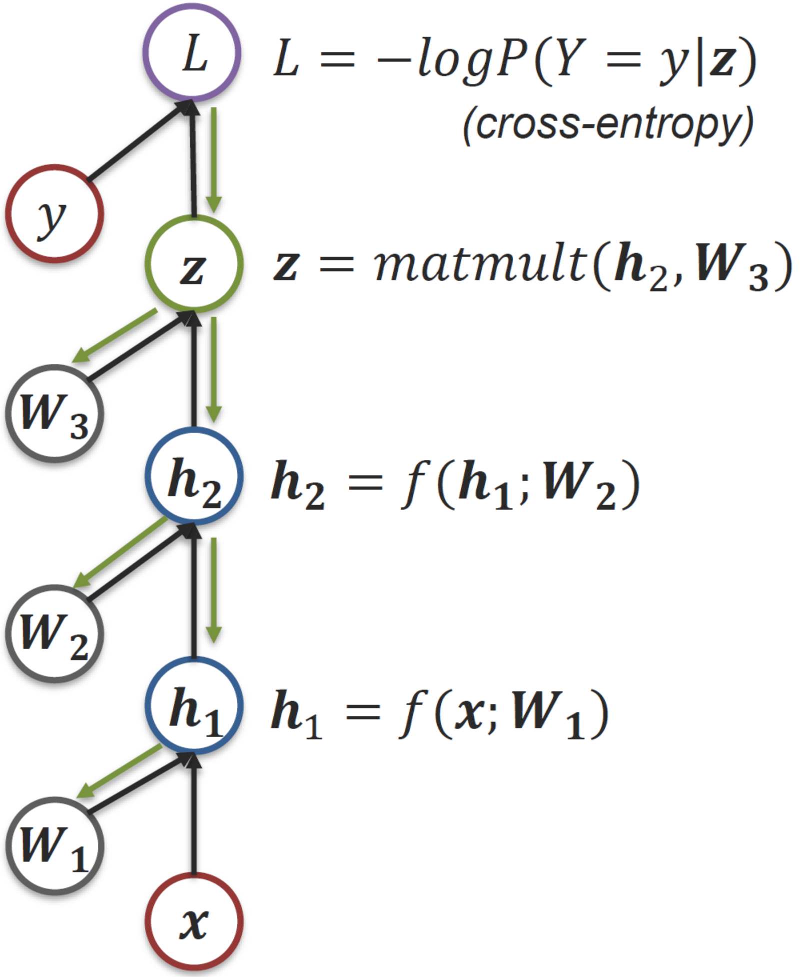
- Following the graph topology, compute value of each unit

Backpropagation pass

- Initialize output gradient = 1
- Compute "local" Jacobian matrix using values from forward pass
- Use the chain rule:

Gradient = "local" Jacobian x "backprop" gradient

- Why is this rule important?



Computational Graph: Multilayer Feedforward Network

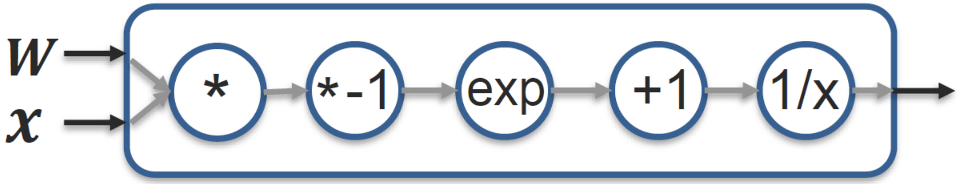
Computational unit:

$\boldsymbol{h} = f(\boldsymbol{x}; \boldsymbol{W})$

- Multiple input
- One output
- Vector/tensor

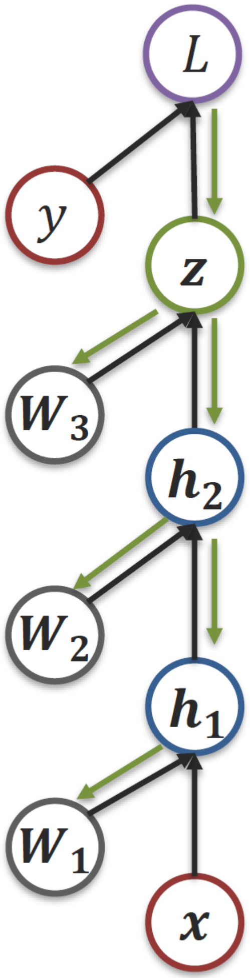
- Sigmoid unit:

$$h_j = (1 + e^{-W_j x})^{-1}$$



Differentiable "unit" function!

(or close approximation to compute "local Jacobian)



$L = -logP(Y = y|\boldsymbol{z})$

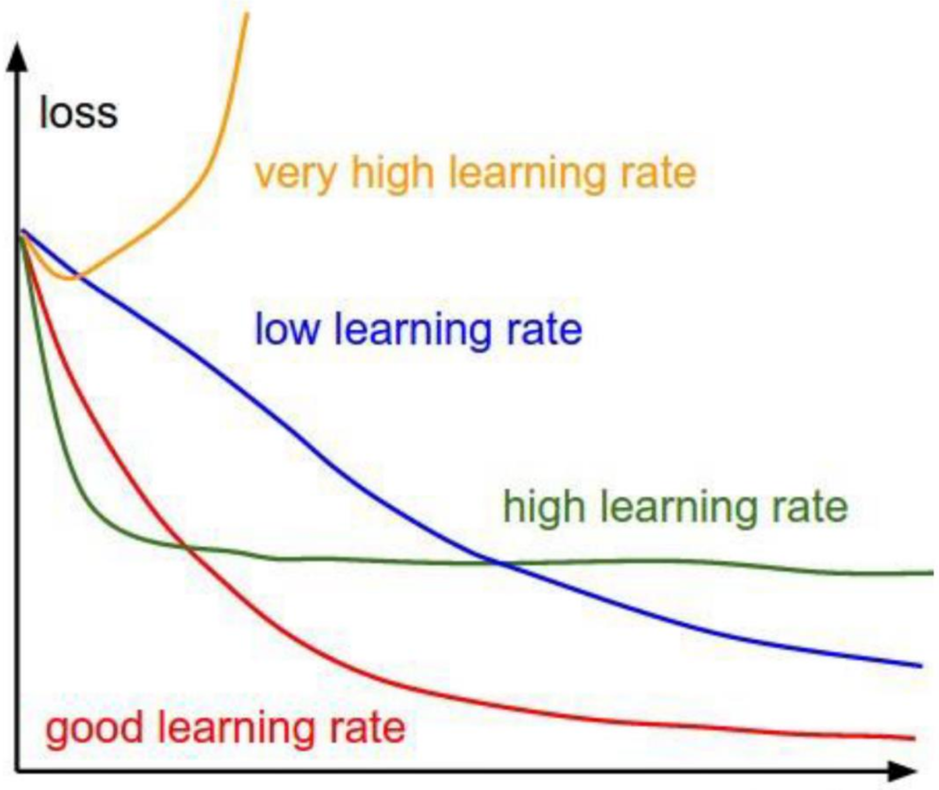
(cross-entropy)

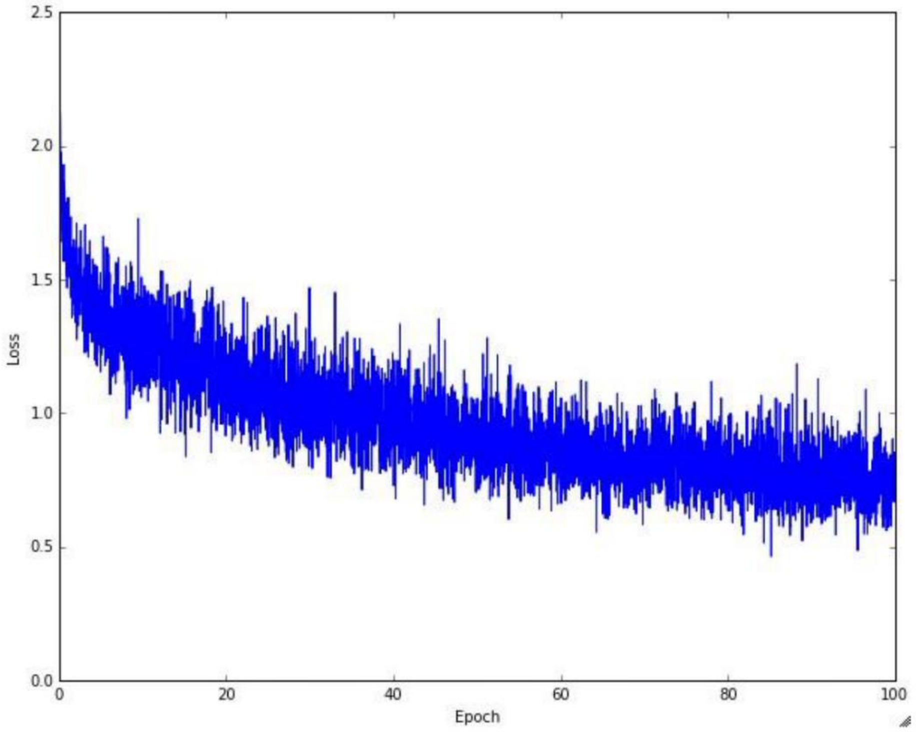
$\boldsymbol{z} = matmult(\boldsymbol{h}_2, \boldsymbol{W}_3)$

$\boldsymbol{h}_2 = f(\boldsymbol{h}_1; \boldsymbol{W}_2)$

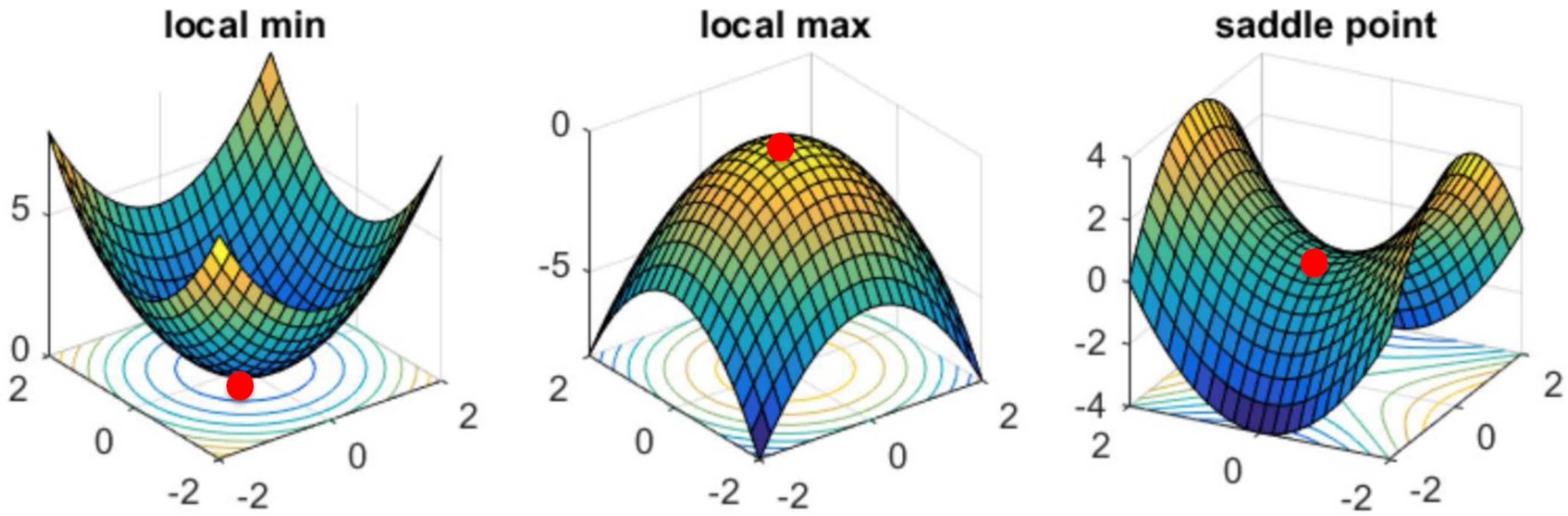
$\boldsymbol{h}_1 = f(\boldsymbol{x}; \boldsymbol{W}_1)$

Interpreting learning rates





Critical Points



Detecting Saddles

One way to detect saddles:

- Calculate Hessian at point x
- If Hessian is indefinite you have a saddle for sure.
- If Hessian is not indefinite you really can't tell.

"My loss isn't changing"

- You are definitely close to a critical point
  - You may be in a saddle point
  - You may be in the local minima/maxima
- One trick: quickly check the surrounding
  - Best practical trick if Hessian is not indefinite.

Adaptive Learning Rate

Key Idea: Let neurons who just started learning have huge learning rate.

Adaptive Learning Rate is an active area of research:

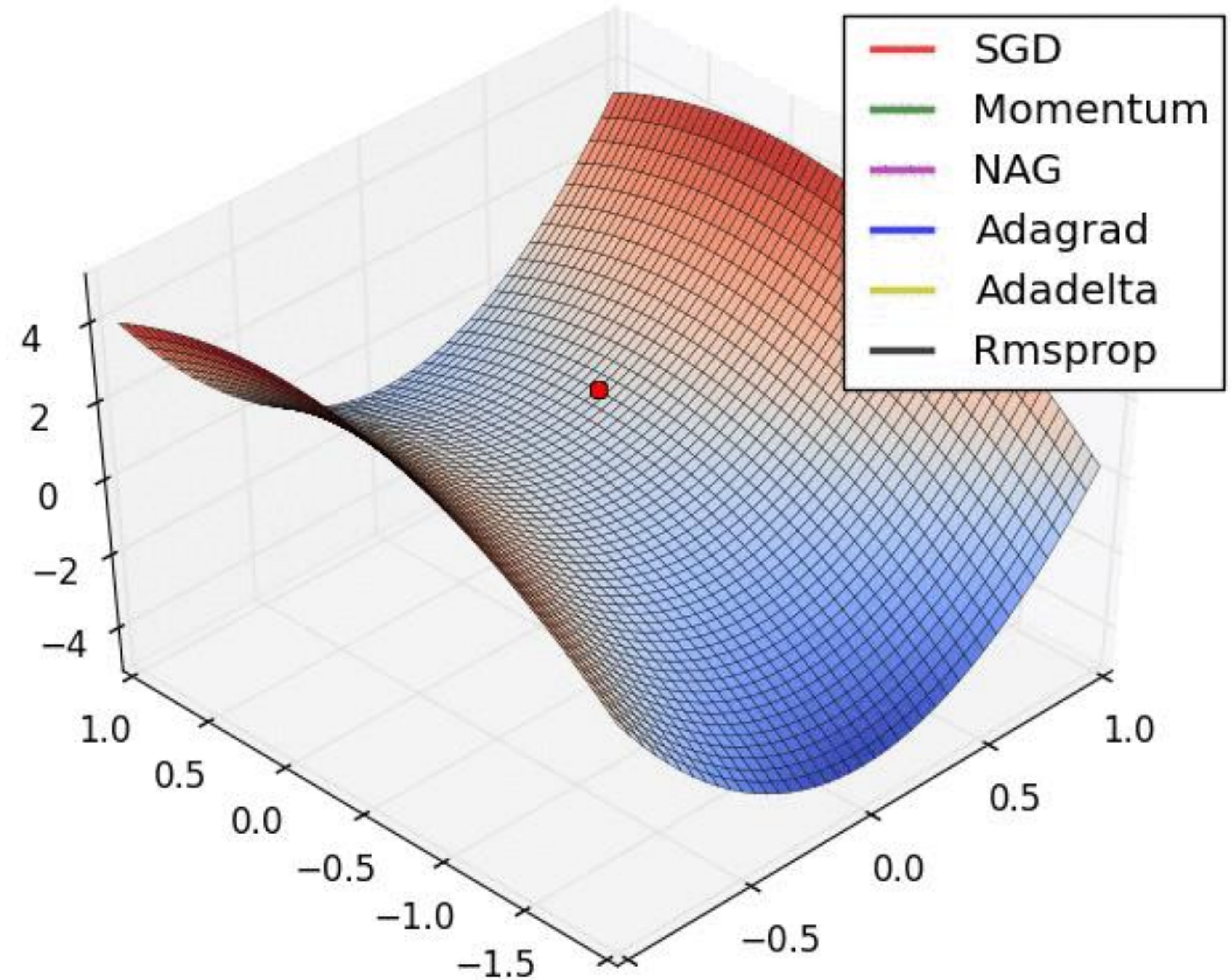
- Adadelta
- RMSProp

```
cache = decay_rate * cache + (1 - decay_rate) * dx**2
x += - learning_rate * dx / (np.sqrt(cache) + eps)
```

- Adam

```
m = beta1*m + (1-beta1)*dx
v = beta2*v + (1-beta2)*(dx**2)
x += - learning_rate * m / (np.sqrt(v) + eps)
```

Adaptive Learning Rate

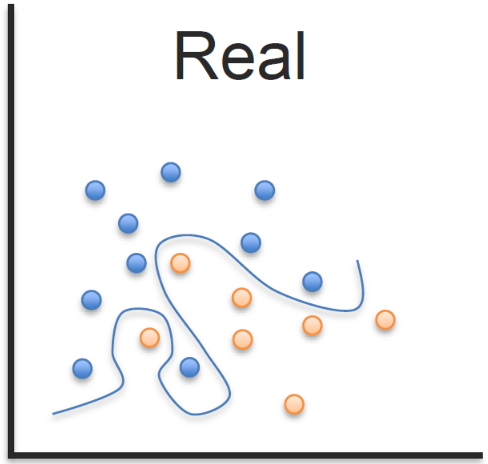


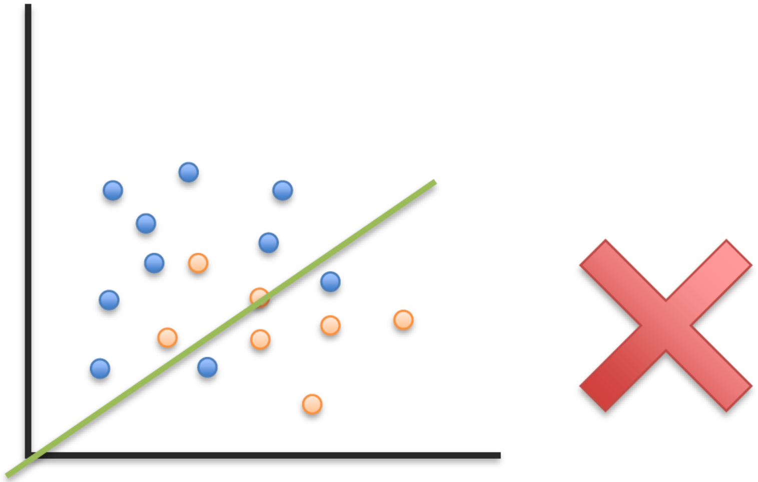
Bias Variance

Problem of bias and variance

- Simple models are unlikely to find the solution to a hard problem, thus probability of finding the right model is low.

Not an issue these days!

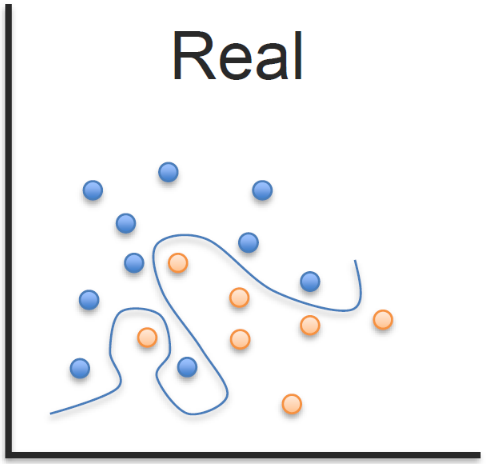


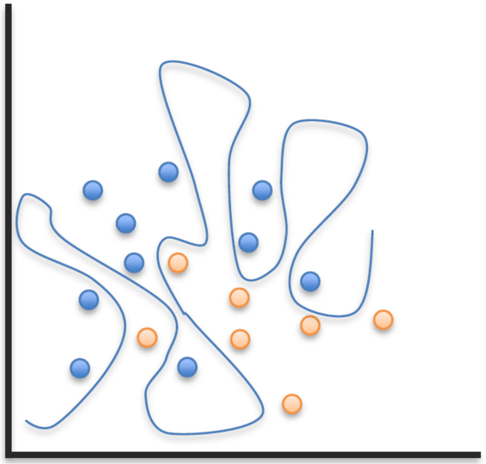


Problem of bias and variance

- Simple models are unlikely to find the solution to a hard problem, thus probability of finding the right model is low.
- Complex models find many solutions to a problem, thus probability of finding the right model is again low.

A big issue with deep learning!

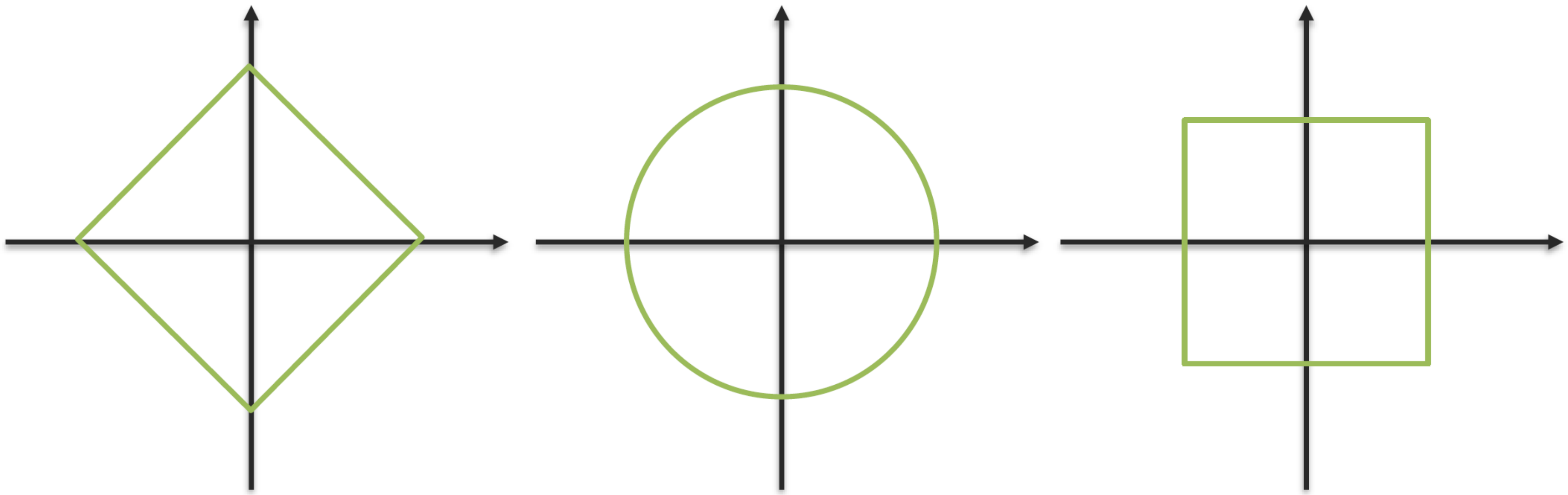




Parameter Regularization

Adding prior to the network parameters

- L^p Norms



L^1 L^2 L^∞

Minimize: $Loss(x;\theta) + \propto ||\theta||$

Structural Regularization

Lots of models can learn everything.

- Go for simpler ones. ← Occam's razor

Take advantage of the structure and "invariances" present in each modality:

- CNNs: translation invariance
- LSTMs: sequential structure
- GRUs: sequential structure

总结

- 了解不同模态的基础表示
- 了解经典机器学习算法
- 掌握神经网络基本原理
- 掌握神经网络的优化实践