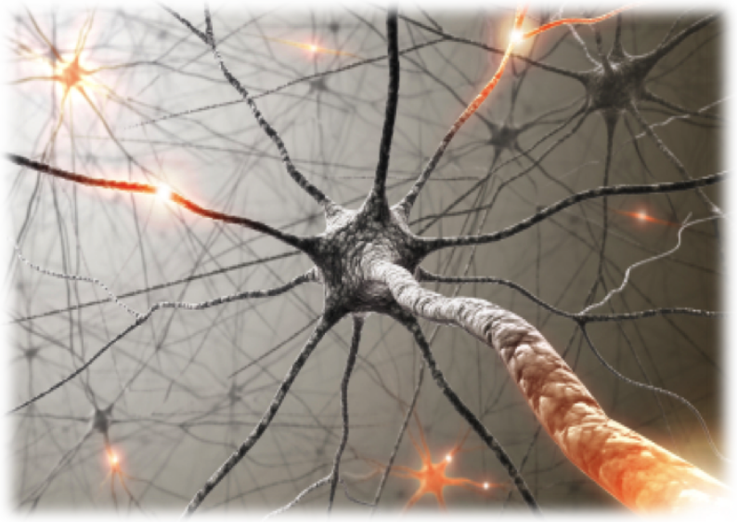
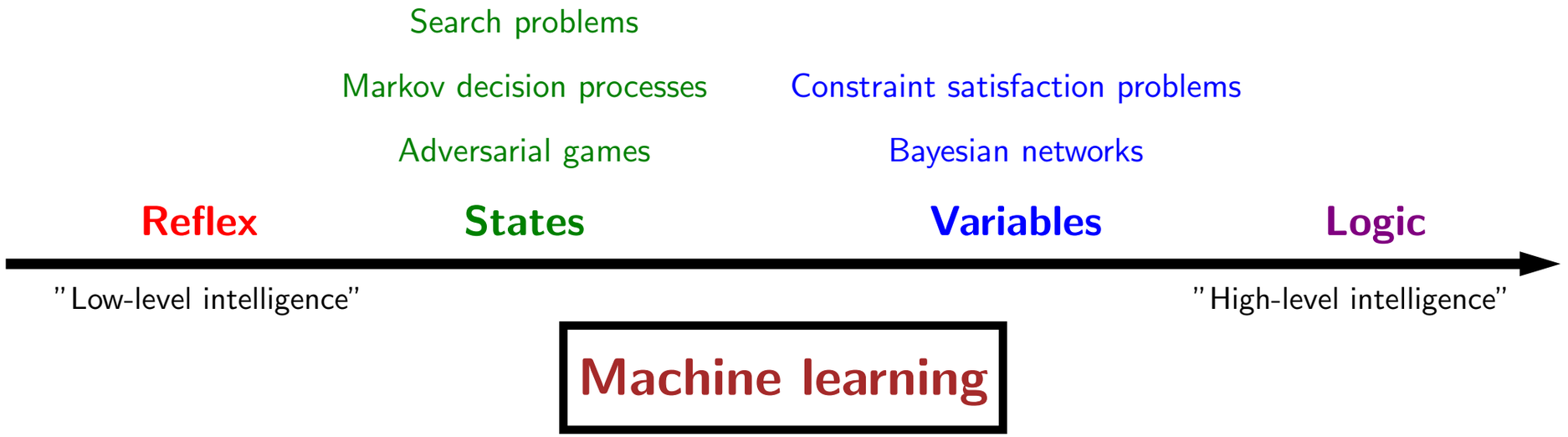
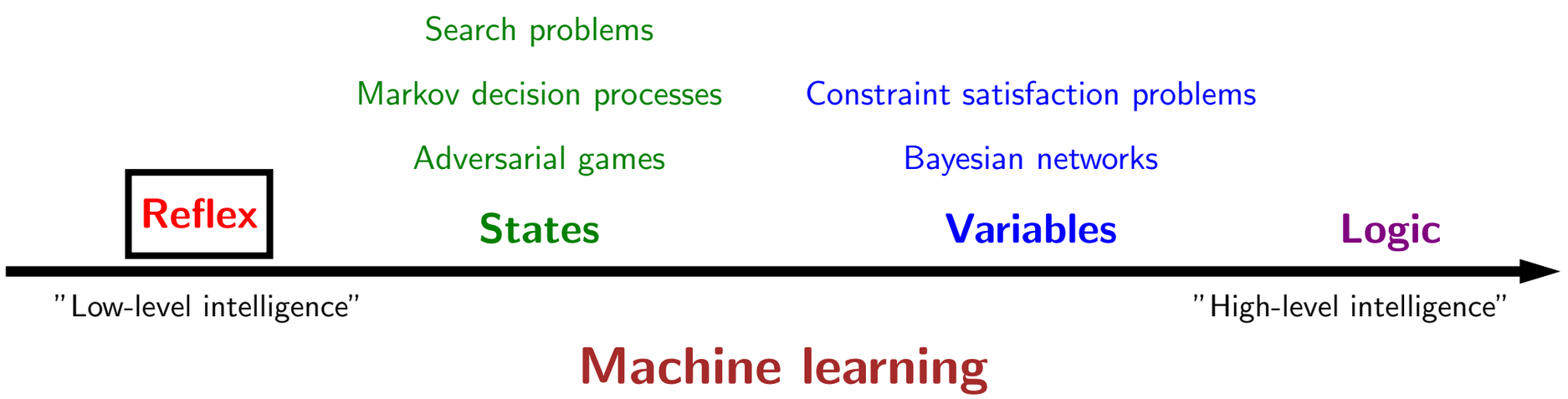
Machine learning: overview



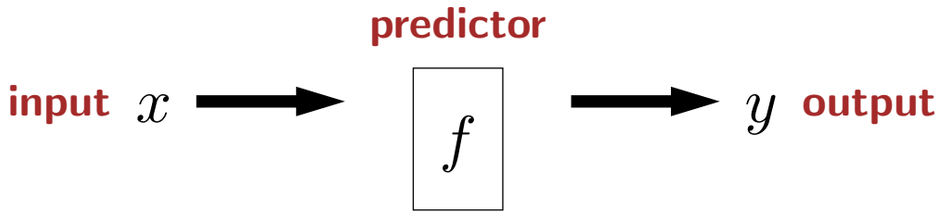
Course plan


Search problems
Markov decision processes
Adversarial games
Constraint satisfaction problems
Bayesian networks
Reflex
States
Variables
Logic
"Low-level intelligence"
"High-level intelligence"
Machine learning

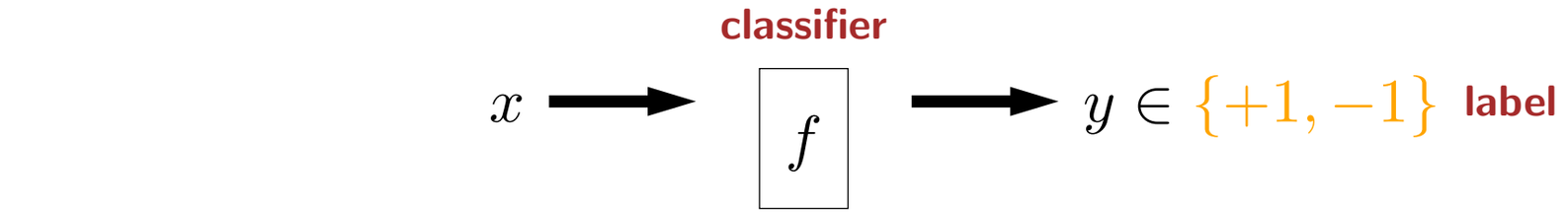
Course plan


Search problems
Markov decision processes
Adversarial games
Constraint satisfaction problems
Bayesian networks
Reflex
States
Variables
Logic
"Low-level intelligence"
"High-level intelligence"
Machine learning

Reflex-based models

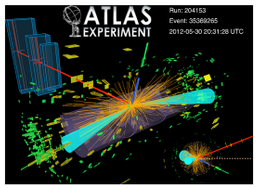


Binary classification



Fraud detection: credit card transaction $\rightarrow$ fraud or no fraud

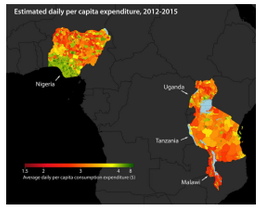
Toxic comments: online comment $\rightarrow$ toxic or not toxic



Higgs boson: measurements of event $\rightarrow$ decay event or background

Regression

$$x \longrightarrow \boxed{f} \longrightarrow y \in \mathbb{R} \quad \textbf{response}$$



Poverty mapping: satellite image $\rightarrow$ asset wealth index

Housing: information about house $\rightarrow$ price

Arrival times: destination, weather, time $\rightarrow$ time of arrival

Structured prediction

$x \longrightarrow \boxed{f} \longrightarrow y$ is a complex object

Machine translation: English sentence $\rightarrow$ Japanese sentence

Dialogue: conversational history $\rightarrow$ next utterance

Image captioning: image $\rightarrow$ sentence describing image

Image segmentation: image $\rightarrow$ segmentation

Roadmap

Tasks

Linear regression

Linear classification

K-means

Models

Non-linear features

Feature templates

Neural networks

Differentiable programming

Algorithms

Stochastic gradient descent

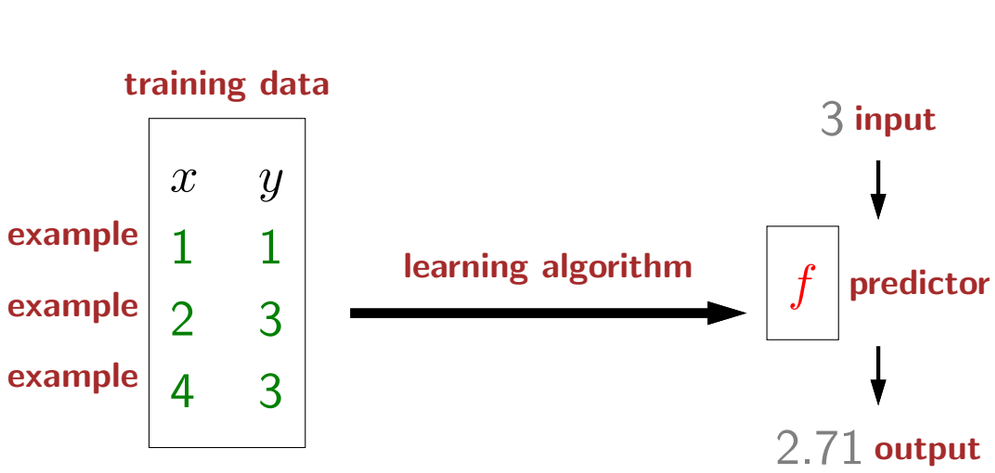
Backpropagation

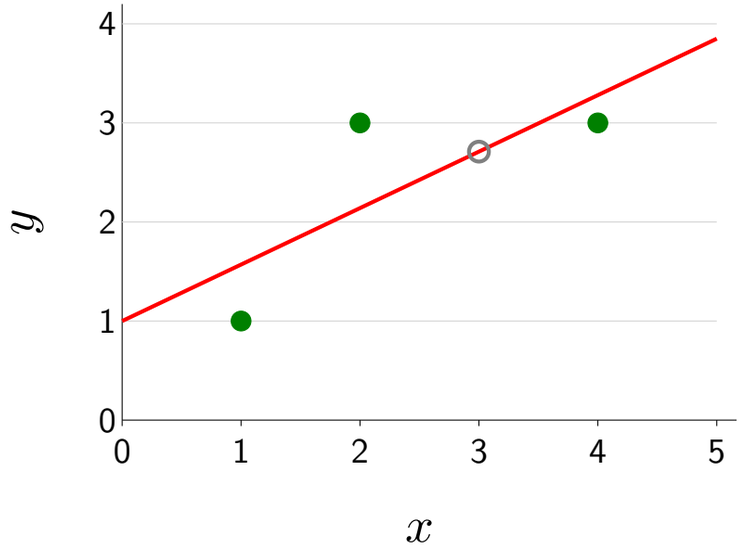
Considerations

Generalization

Best practices

Linear regression framework





Design decisions:

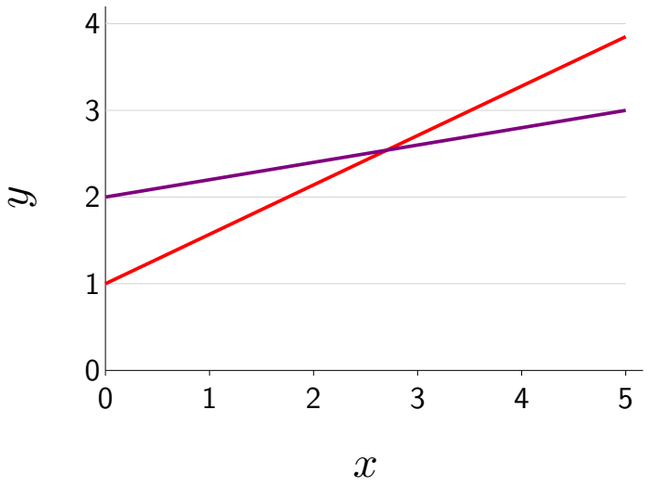
- Which predictors are possible? **hypothesis class**
- How good is a predictor? **loss function**
- How do we compute the best predictor? **optimization algorithm**

Hypothesis class: which predictors?

$$f(x) = 1 + 0.57x$$

$$f(x) = 2 + 0.2x$$

$$f(x) = w_1 + w_2 x$$



Vector notation:

weight vector $\mathbf{w} = [w_1, w_2]$ feature extractor $\phi(x) = [1, x]$ feature vector

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) \text{ score}$$

$$f_{\mathbf{w}}(3) = [1, 0.57] \cdot [1, 3] = 2.71$$

Hypothesis class:

$$\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^2\}$$

Loss function: how good is a predictor?

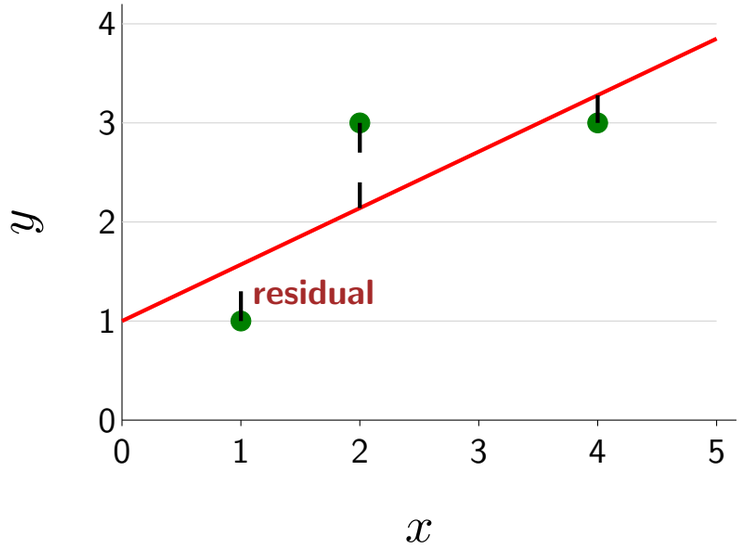
$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

$\mathbf{w} = [1, 0.57]$

$\phi(x) = [1, x]$

training data $\mathcal{D}_{\text{train}}$

x	y
1	1
2	3
4	3



$$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$$ **squared loss**

$$\text{Loss}(1, 1, [1, 0.57]) = ([1, 0.57] \cdot [1, 1] - 1)^2 = 0.32$$

$$\text{Loss}(2, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 2] - 3)^2 = 0.74$$

$$\text{Loss}(4, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 4] - 3)^2 = 0.08$$

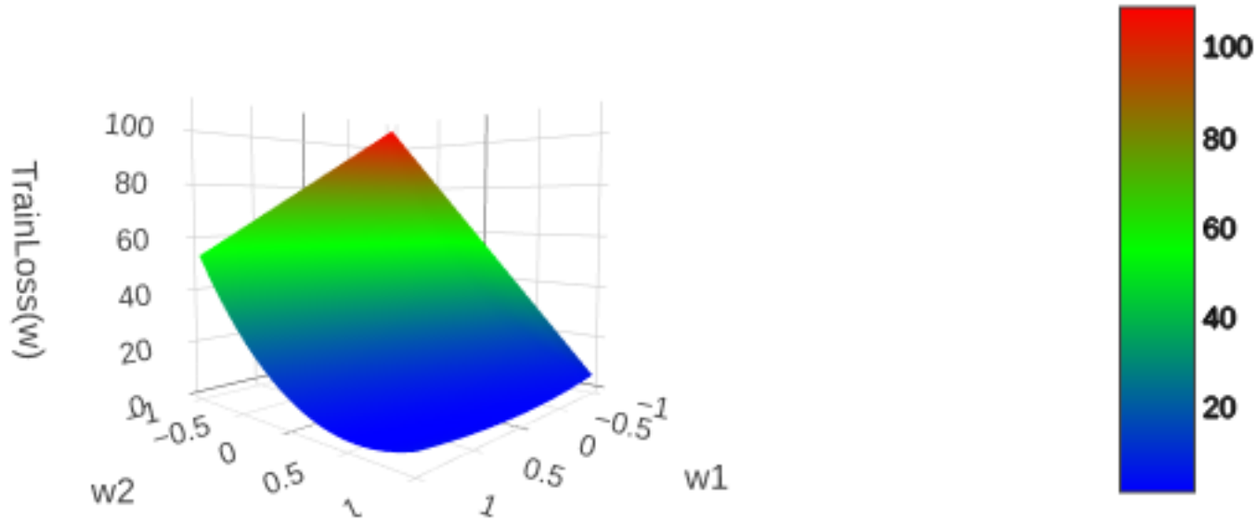
$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{w})$$

$$\text{TrainLoss}([1, 0.57]) = 0.38$$

Loss function: visualization

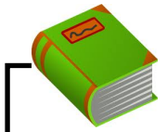
$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} (f_{\mathbf{w}}(x) - y)^2$$

$$\min_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$$



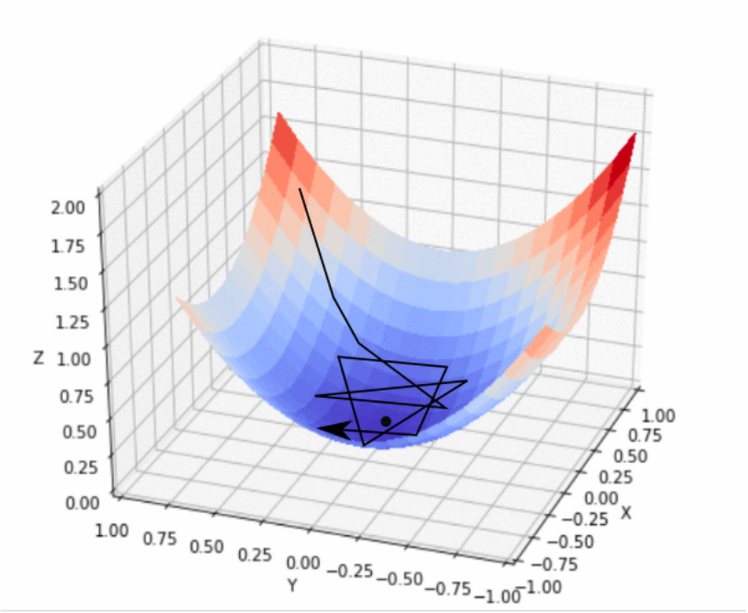
Optimization algorithm: how to compute best?

Goal: $\min_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$



Definition: gradient

The gradient $\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$ is the direction that increases the training loss the most.

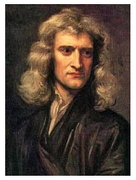


Algorithm: gradient descent

Initialize $\mathbf{w} = [0, \ldots, 0]$

For $t = 1, \ldots, T$: **epochs**

$$\mathbf{w} \leftarrow \mathbf{w} - \underbrace{\eta}_{\text{step size}} \underbrace{\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})}_{\text{gradient}}$$



Computing the gradient

Objective function:

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} (\mathbf{w} \cdot \phi(x) - y)^2$$

Gradient (use chain rule):

$$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} 2(\underbrace{\mathbf{w} \cdot \phi(x) - y}_{\text{prediction} - \text{target}})\phi(x)$$

Gradient descent example

training data $\mathcal{D}_{\text{train}}$

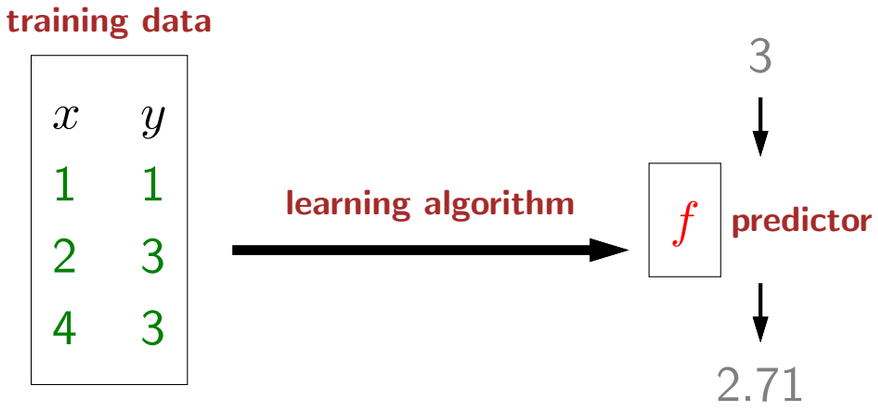
x	y
1	1
2	3
4	3

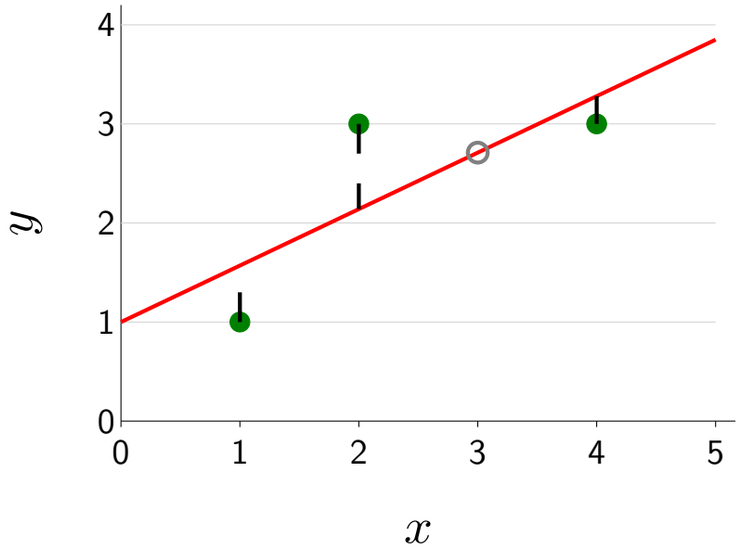
$$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} 2(\mathbf{w} \cdot \phi(x) - y)\phi(x)$$

Gradient update: $\mathbf{w} \leftarrow \mathbf{w} - 0.1 \nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$

t	$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$	$\mathbf{w}$
		$[0, 0]$
1	$\underbrace{\frac{1}{3}(2([0,0] \cdot [1,1] - 1)[1,1] + 2([0,0] \cdot [1,2] - 3)[1,2] + 2([0,0] \cdot [1,4] - 3)[1,4])}_{=[-4.67, -12.67]}$	$[0.47, 1.27]$
2	$\underbrace{\frac{1}{3}(2([0.47,1.27] \cdot [1,1] - 1)[1,1] + 2([0.47,1.27] \cdot [1,2] - 3)[1,2] + 2([0.47,1.27] \cdot [1,4] - 3)[1,4])}_{=[2.18, 7.24]}$	$[0.25, 0.54]$
...	...	...
200	$\underbrace{\frac{1}{3}(2([1,0.57] \cdot [1,1] - 1)[1,1] + 2([1,0.57] \cdot [1,2] - 3)[1,2] + 2([1,0.57] \cdot [1,4] - 3)[1,4])}_{=[0,0]}$	$[1, 0.57]$

Summary





Which predictors are possible?

Hypothesis class

Linear functions

$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)\}, \phi(x) = [1, x]$

How good is a predictor?

Loss function

Squared loss

$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$

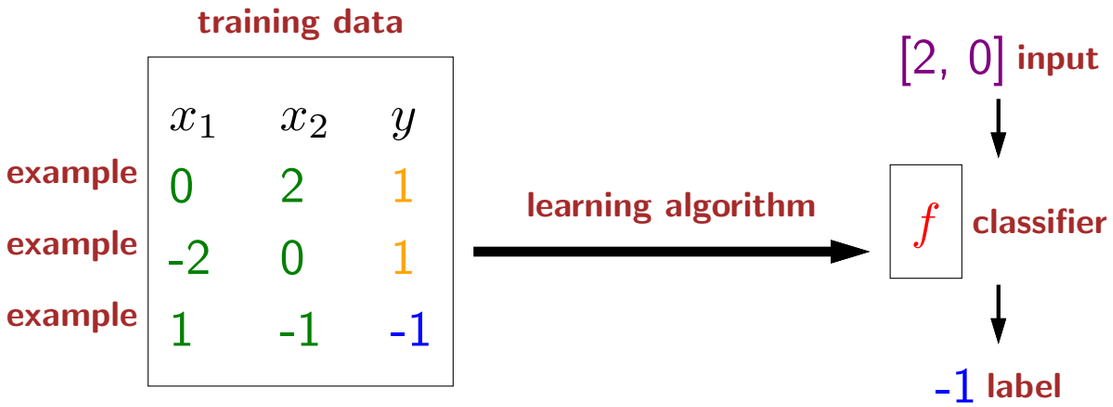
How to compute best predictor?

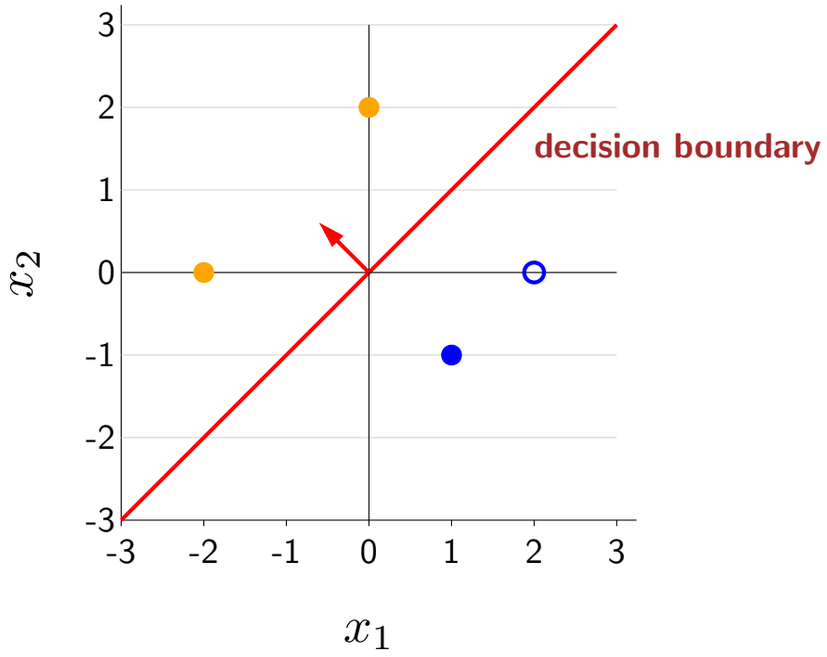
Optimization algorithm

Gradient descent

$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \textbf{TrainLoss}(\mathbf{w})$

Linear classification framework





Design decisions:

Which classifiers are possible? **hypothesis class**

How good is a classifier? **loss function**

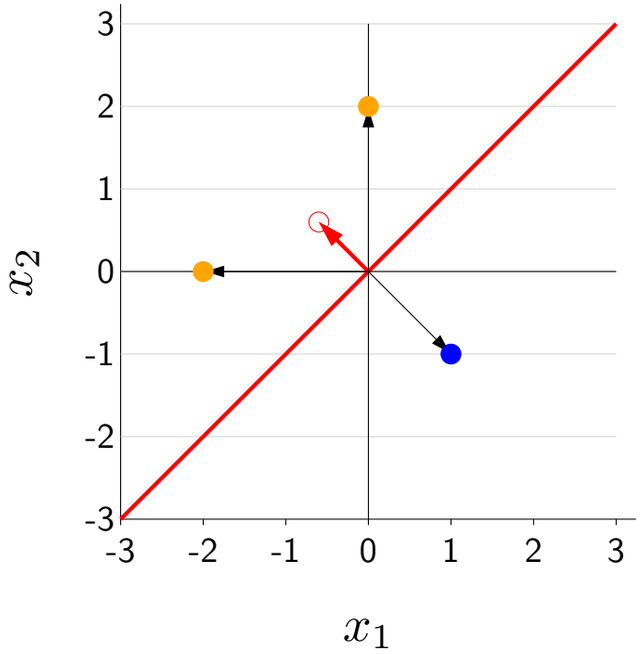
How do we compute the best classifier? **optimization algorithm**

An example linear classifier

$$f(x) = \text{sign}(\overbrace{[-0.6, 0.6]}^{\mathbf{w}} \cdot \overbrace{[x_1, x_2]}^{\phi(x)})$$

$$\text{sign}(z) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \end{cases}$$

x_1	x_2	$f(x)$
0	2	1
-2	0	1
1	-1	-1



$$f([0, 2]) = \text{sign}([-0.6, 0.6] \cdot [0, 2]) = \text{sign}(1.2) = 1$$

$$f([-2, 0]) = \text{sign}([-0.6, 0.6] \cdot [-2, 0]) = \text{sign}(1.2) = 1$$

$$f([1, -1]) = \text{sign}([-0.6, 0.6] \cdot [1, -1]) = \text{sign}(-1.2) = -1$$

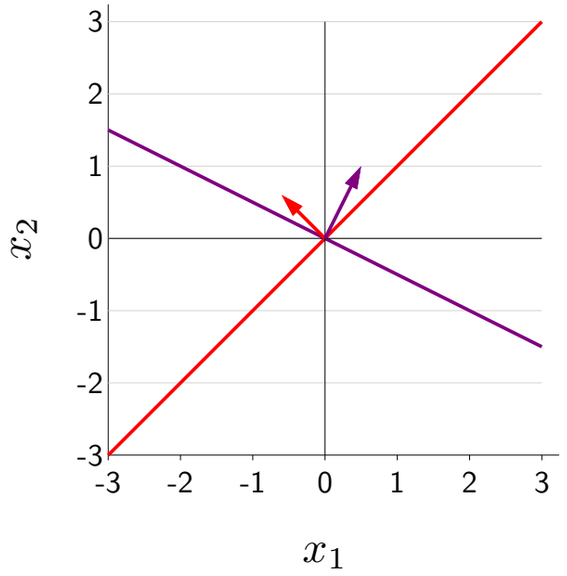
Decision boundary: x such that $\mathbf{w} \cdot \phi(x) = 0$

Hypothesis class: which classifiers?

$$\phi(x) = [x_1, x_2]$$

$$f(x) = \text{sign}([-0.6, 0.6] \cdot \phi(x))$$

$$f(x) = \text{sign}([0.5, 1] \cdot \phi(x))$$



General binary classifier:

$$f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$$

Hypothesis class:

$$\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^2\}$$

Loss function: how good is a classifier?

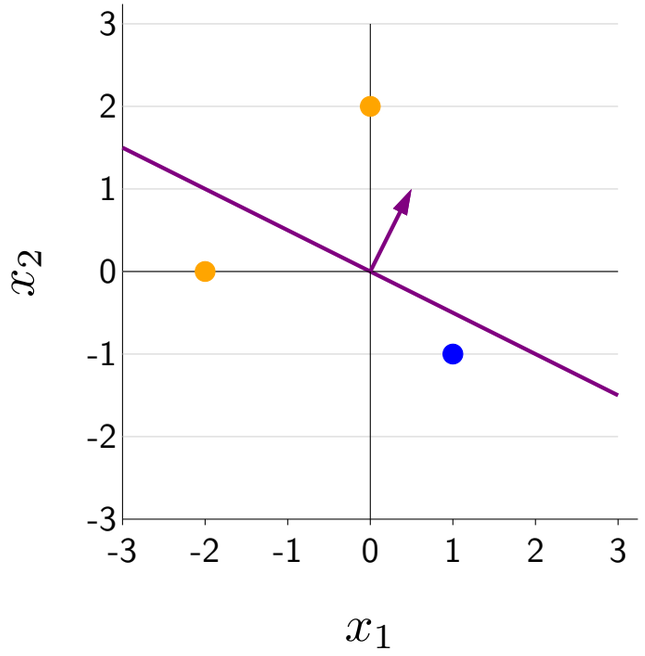
$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

$\mathbf{w} = [0.5, 1]$

$\phi(x) = [x_1, x_2]$

training data $\mathcal{D}_{\text{train}}$

x_1	x_2	y
0	2	1
-2	0	1
1	-1	-1



$$\text{Loss}_{0\text{-}1}(x, y, \mathbf{w}) = \mathbf{1}[f_{\mathbf{w}}(x) \neq y]$$ **zero-one loss**

$$\text{Loss}([0, 2], 1, [0.5, 1]) = \mathbf{1}[\text{sign}([0.5, 1] \cdot [0, 2]) \neq 1] = 0$$

$$\text{Loss}([-2, 0], 1, [0.5, 1]) = \mathbf{1}[\text{sign}([0.5, 1] \cdot [-2, 0]) \neq 1] = 1$$

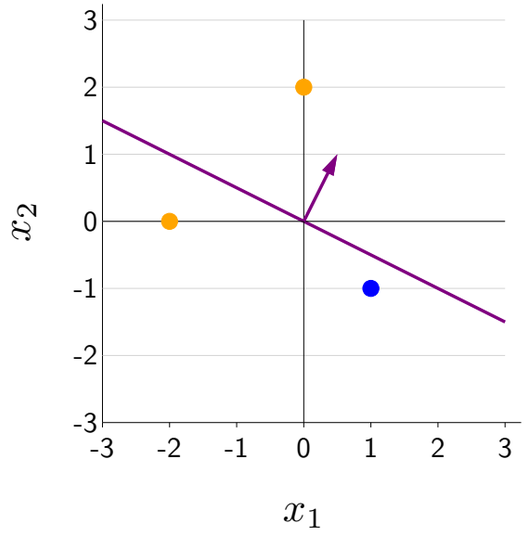
$$\text{Loss}([1, -1], -1, [0.5, 1]) = \mathbf{1}[\text{sign}([0.5, 1] \cdot [1, -1]) \neq -1] = 0$$

$$\text{TrainLoss}([0.5, 1]) = 0.33$$

Score and margin

Predicted label: $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$

Target label: y



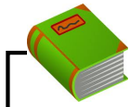
Definition: score

The score on an example (x, y) is $\mathbf{w} \cdot \phi(x)$, how **confident** we are in predicting $+1$.

Definition: margin

The margin on an example (x, y) is $(\mathbf{w} \cdot \phi(x))y$, how **correct** we are.

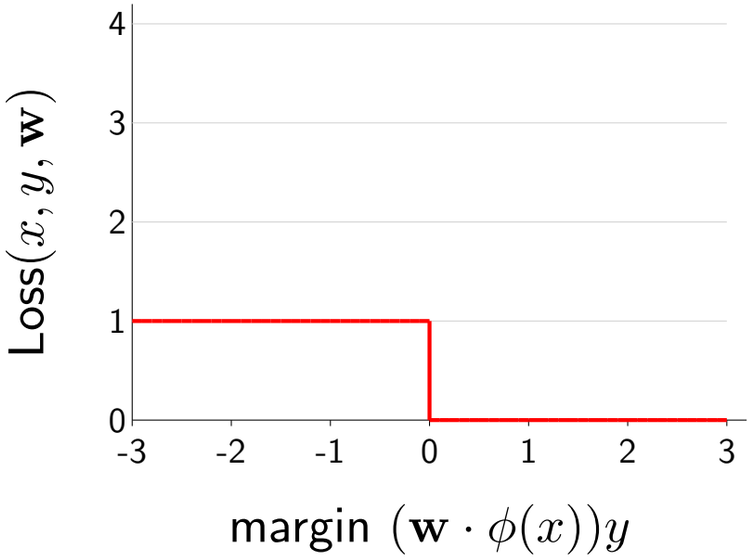
Zero-one loss rewritten



Definition: zero-one loss

$$\text{Loss}_{0\text{-}1}(x, y, \mathbf{w}) = \mathbf{1}[f_{\mathbf{w}}(x) \neq y]$$

$$= \mathbf{1}[\underbrace{(\mathbf{w} \cdot \phi(x))y}_{\text{margin}} \leq 0]$$



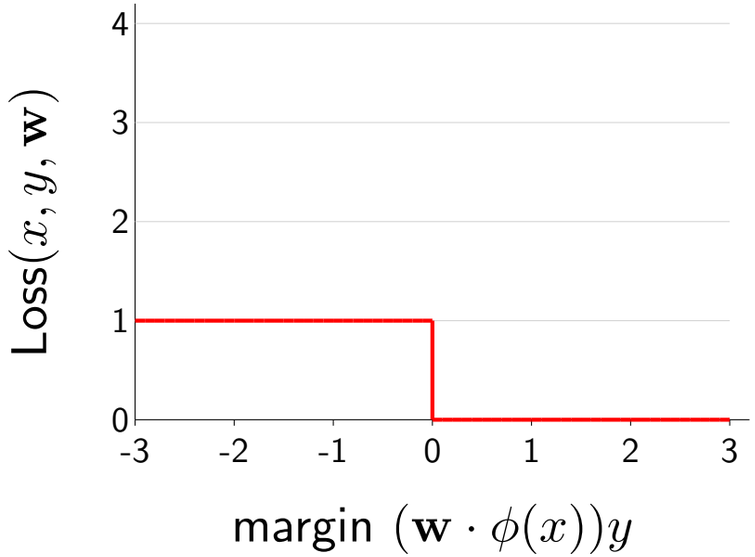
Optimization algorithm: how to compute best?

Goal: $\min_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$

To run gradient descent, compute the gradient:

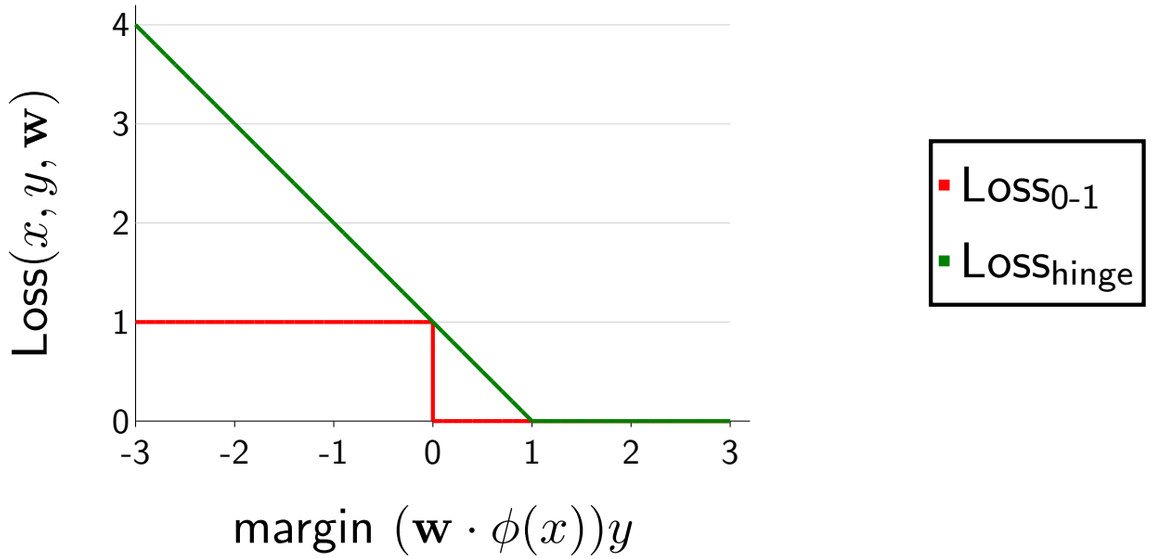
$$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \nabla \text{Loss}_{\text{0-1}}(x, y, \mathbf{w})$$

$$\nabla_{\mathbf{w}} \text{Loss}_{\text{0-1}}(x, y, \mathbf{w}) = \nabla \mathbf{1}[(\mathbf{w} \cdot \phi(x))y \leq 0]$$



Gradient is zero almost everywhere!

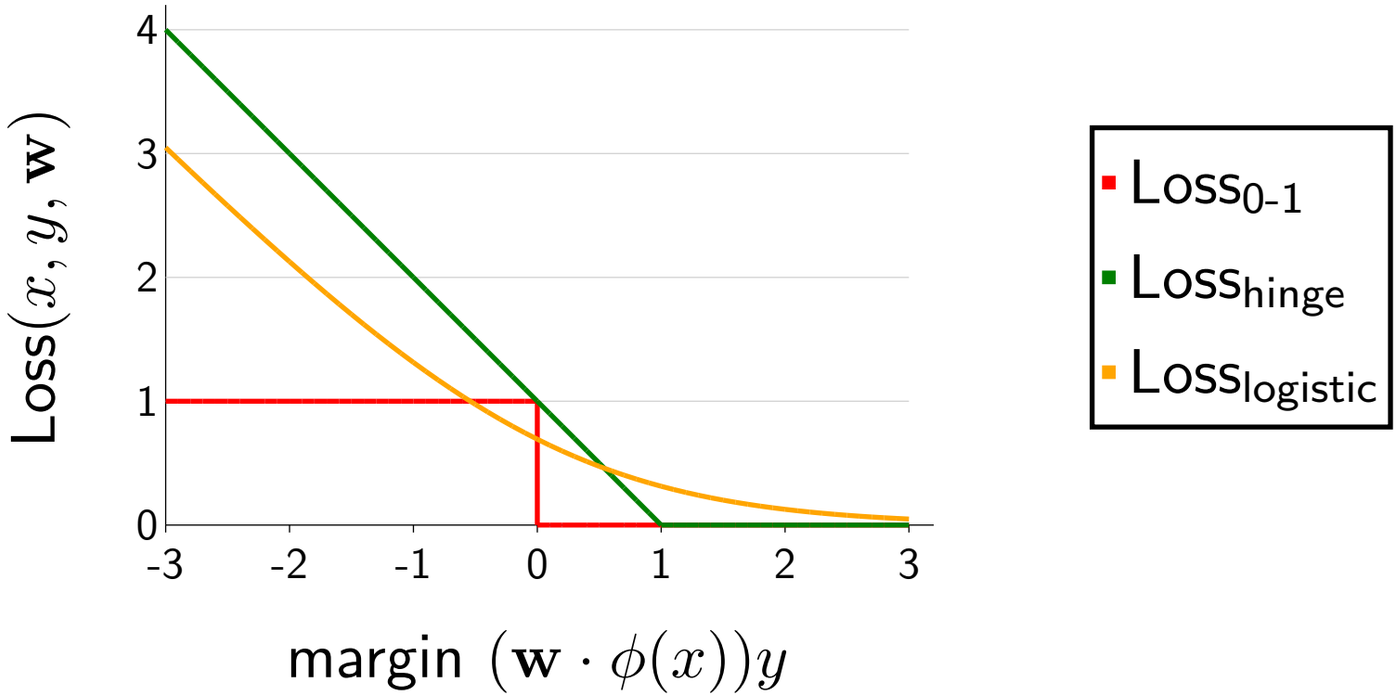
Hinge loss



$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{1 - (\mathbf{w} \cdot \phi(x))y, 0\}$$

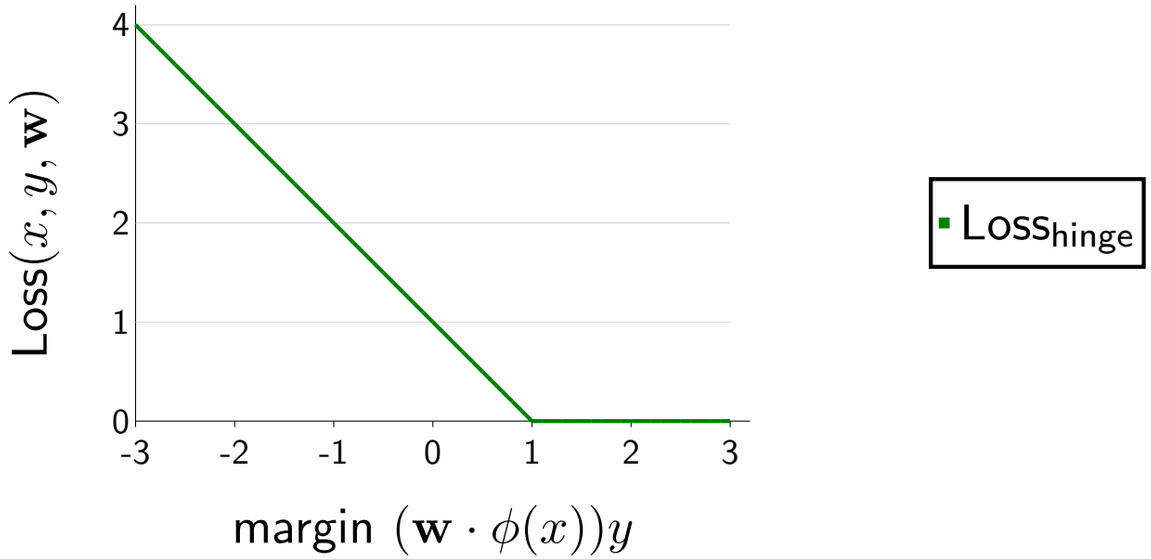
Digression: logistic regression

$$\text{Loss}_{\text{logistic}}(x, y, \mathbf{w}) = \log(1 + e^{-(\mathbf{w} \cdot \phi(x))y})$$



Intuition: Try to increase margin even when it already exceeds 1

Gradient of the hinge loss



$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{1 - (\mathbf{w} \cdot \phi(x))y, 0\}$$

$$\nabla \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \begin{cases} -\phi(x)y & \text{if } \{1 - (\mathbf{w} \cdot \phi(x))y\} > \{0\} \\ 0 & \text{otherwise} \end{cases}$$

Hinge loss on training data

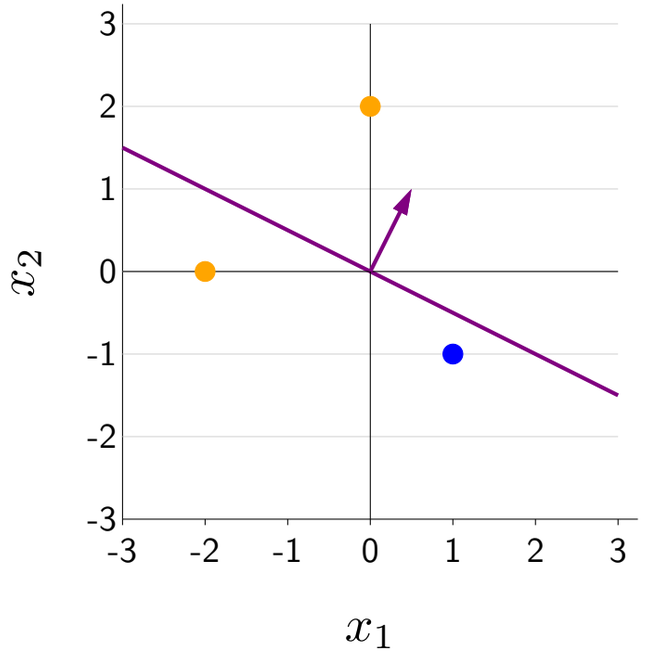
$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

$\mathbf{w} = [0.5, 1]$

$\phi(x) = [x_1, x_2]$

training data $\mathcal{D}_{\text{train}}$

x_1	x_2	y
0	2	1
-2	0	1
1	-1	-1



$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{1 - (\mathbf{w} \cdot \phi(x))y, 0\}$$

$\text{Loss}([0, 2], 1, [0.5, 1]) = \max\{1 - [0.5, 1] \cdot [0, 2](1), 0\} = 0$

$\text{Loss}([-2, 0], 1, [0.5, 1]) = \max\{1 - [0.5, 1] \cdot [-2, 0](1), 0\} = 2$

$\text{Loss}([1, -1], -1, [0.5, 1]) = \max\{1 - [0.5, 1] \cdot [1, -1](-1), 0\} = 0.5$

$\text{TrainLoss}([0.5, 1]) = 0.83$

$\nabla\text{Loss}([0, 2], 1, [0.5, 1]) = [0, 0]$

$\nabla\text{Loss}([-2, 0], 1, [0.5, 1]) = [2, 0]$

$\nabla\text{Loss}([1, -1], -1, [0.5, 1]) = [1, -1]$

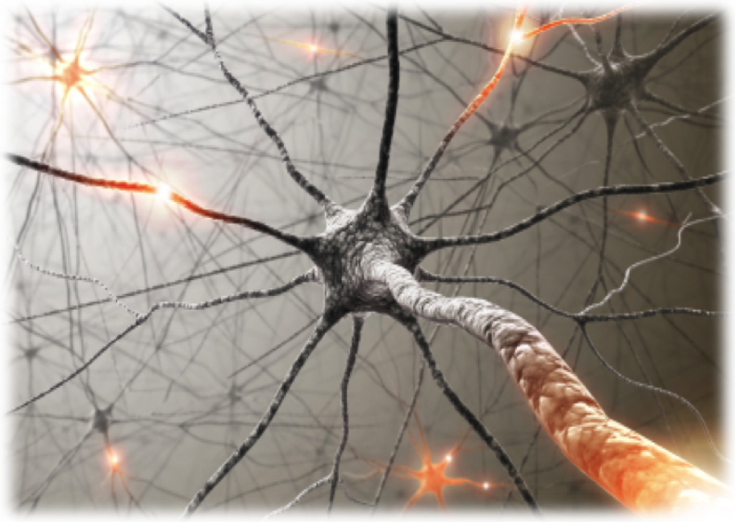
$\nabla\text{TrainLoss}([0.5, 1]) = [1, -0.33]$

Summary so far

$$\underbrace{\mathbf{w} \cdot \phi(x)}_{\text{score}}$$

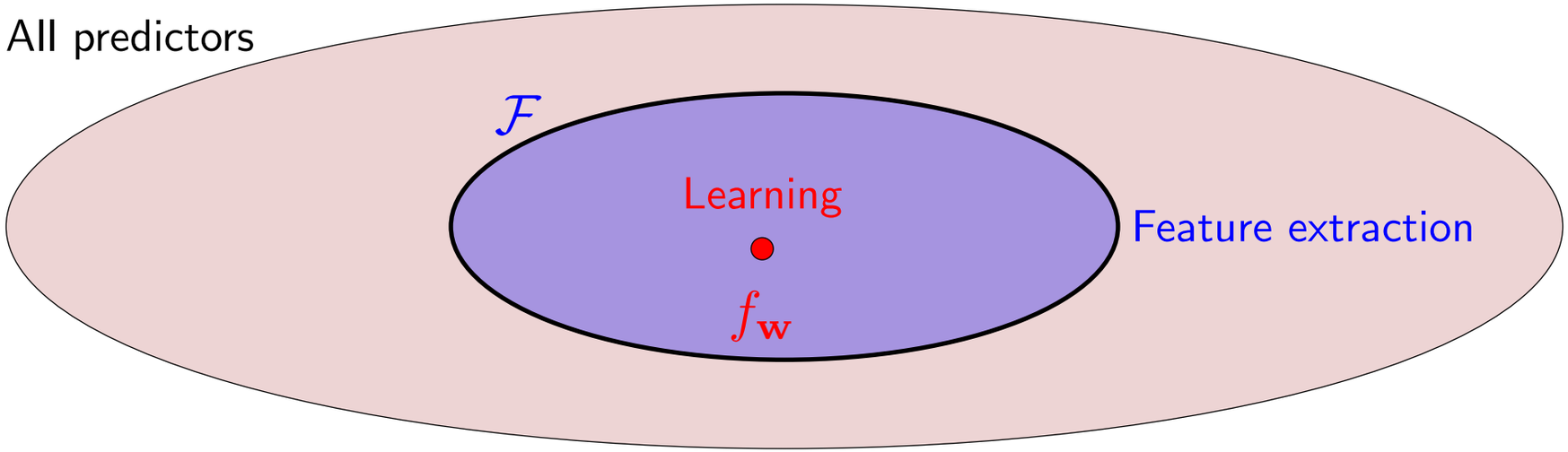
	Regression	Classification
Prediction $f_{\mathbf{w}}(x)$	score	sign(score)
Relate to target y	residual (score $- y$)	margin (score y)
Loss functions	squared absolute deviation	zero-one hinge logistic
Algorithm	gradient descent	gradient descent

Machine learning: feature templates



Feature extraction + learning

$$\mathcal{F} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$



- Feature extraction: choose $\mathcal{F}$ based on domain knowledge
- Learning: choose $f_{\mathbf{w}} \in \mathcal{F}$ based on data

Want $\mathcal{F}$ to contain good predictors but not be too big

Feature extraction with feature names

Example task:

classifier

string (x) → $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$ → valid email address? (y)

Question: what properties of x **might be** relevant for predicting y?

Feature extractor: Given x, produce set of (feature name, feature value) pairs

abc@gmail.com

x

feature extractor ϕ

arbitrary!

length>10	: 1
fracOfAlpha	: 0.85
contains_@	: 1
endsWith_com	: 1
endsWith_org	: 0

$\phi(x)$

Prediction with feature names

Weight vector $\mathbf{w} \in \mathbb{R}^d$

length>10	:-1.2
fracOfAlpha	:0.6
contains_@	:3
endsWith_com	:2.2
endsWith_org	:1.4

Feature vector $\phi(x) \in \mathbb{R}^d$

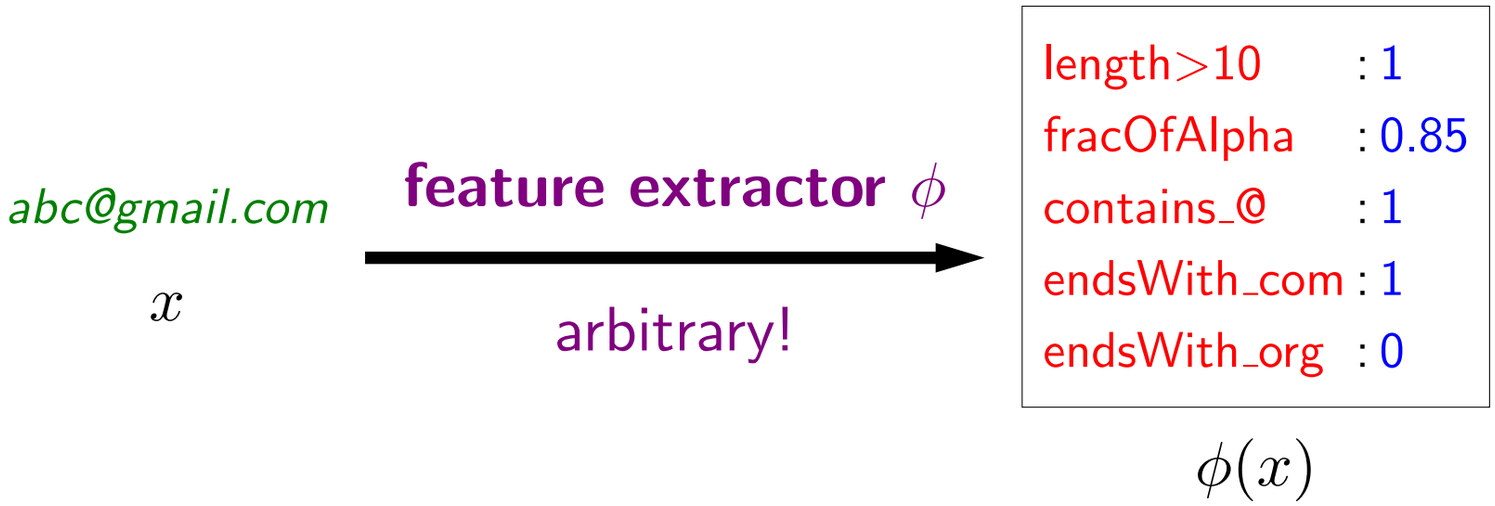
length>10	:1
fracOfAlpha	:0.85
contains_@	:1
endsWith_com	:1
endsWith_org	:0

Score: weighted combination of features

$$\mathbf{w} \cdot \phi(x) = \sum_{j=1}^{d} w_j \phi(x)_j$$

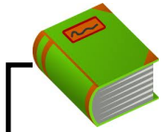
Example: $-1.2(1) + 0.6(0.85) + 3(1) + 2.2(1) + 1.4(0) = 4.51$

Organization of features?



Which features to include? Need an organizational principle...

Feature templates



Definition: feature template

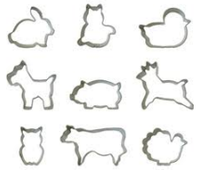
A **feature template** is a group of features all computed in a similar way.

abc@gmail.com

last three characters equals ___

endsWith_aaa : 0
endsWith_aab : 0
endsWith_aac : 0
...
endsWith_com : 1
...
endsWith_zzz : 0

Define types of pattern to look for, not particular patterns

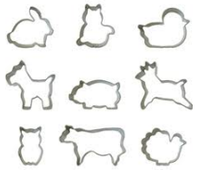


Feature templates example 1

Input:

abc@gmail.com

Feature template	Example feature	
Last three characters equals ___	Last three characters equals *com*	: 1
Length greater than ___	Length greater than *10*	: 1
Fraction of alphanumeric characters	Fraction of alphanumeric characters	: 0.85



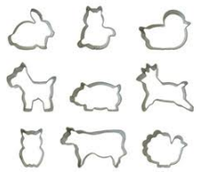
Feature templates example 2

Input:

Latitude: 37.4068176

Longitude: -122.1715122

Feature template	Example feature name
Pixel intensity of image at row ___ and column ___ (___ channel)	Pixel intensity of image at row *10* and column *93* (*red* channel) : 0.8
Latitude is in [___, ___] and longitude is in [___, ___]	Latitude is in [*37.4*, *37.5*] and longitude is in [*-122.2*, *-122.1*] : 1



Sparsity in feature vectors

abc@gmail.com

last character equals ___

endsWith_a	0
endsWith_b	0
endsWith_c	0
endsWith_d	0
endsWith_e	0
endsWith_f	0
endsWith_g	0
endsWith_h	0
endsWith_i	0
endsWith_j	0
endsWith_k	0
endsWith_l	0
endsWith_m	**1**
endsWith_n	0
endsWith_o	0
endsWith_p	0
endsWith_q	0
endsWith_r	0
endsWith_s	0
endsWith_t	0
endsWith_u	0
endsWith_v	0
endsWith_w	0
endsWith_x	0
endsWith_y	0
endsWith_z	0

Compact representation:

```
{"endsWith_m":  1}
```

Two feature vector implementations

Arrays (good for dense features):

pixelIntensity(0,0)	0.8
pixelIntensity(0,1)	0.6
pixelIntensity(0,2)	0.5
pixelIntensity(1,0)	0.5
pixelIntensity(1,1)	0.8
pixelIntensity(1,2)	0.7
pixelIntensity(2,0)	0.2
pixelIntensity(2,1)	0
pixelIntensity(2,2)	0.1

```
[0.8, 0.6, 0.5, 0.5, 0.8, 0.7, 0.2, 0, 0.1]
```

Dictionaries (good for sparse features):

fracOfAlpha	0.85
contains_a	0
contains_b	0
contains_c	0
contains_d	0
contains_e	0
...	
contains_@	1
...	

```
{"fracOfAlpha": 0.85, "contains_@": 1}
```

Summary

$$\mathcal{F} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$

Feature template:

abc@gmail.com

last three characters equals ___

endsWith_aaa : 0
endsWith_aab : 0
endsWith_aac : 0
...
endsWith_com : 1
...
endsWith_zzz : 0

Dictionary implementation:

```
{"endsWith_com":  1}
```