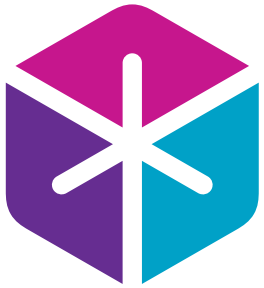
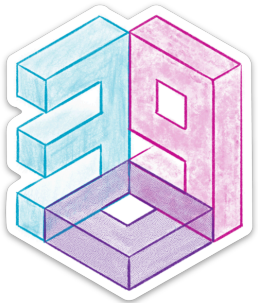




https://introml.mit.edu

6.390 Intro to Machine Learning

Lecture 4: Linear Classification

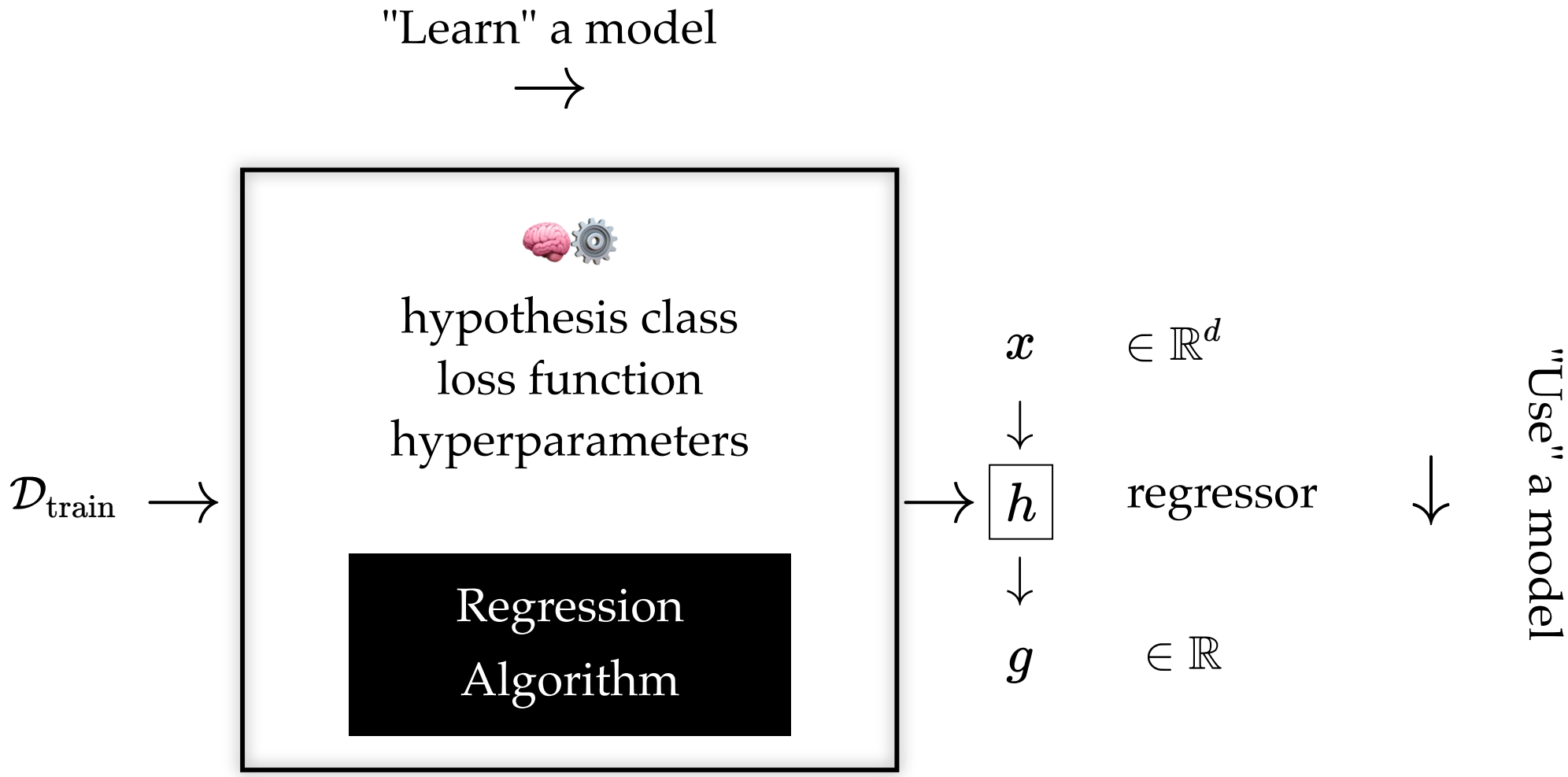
Shen Shen

Sep 28, 2026

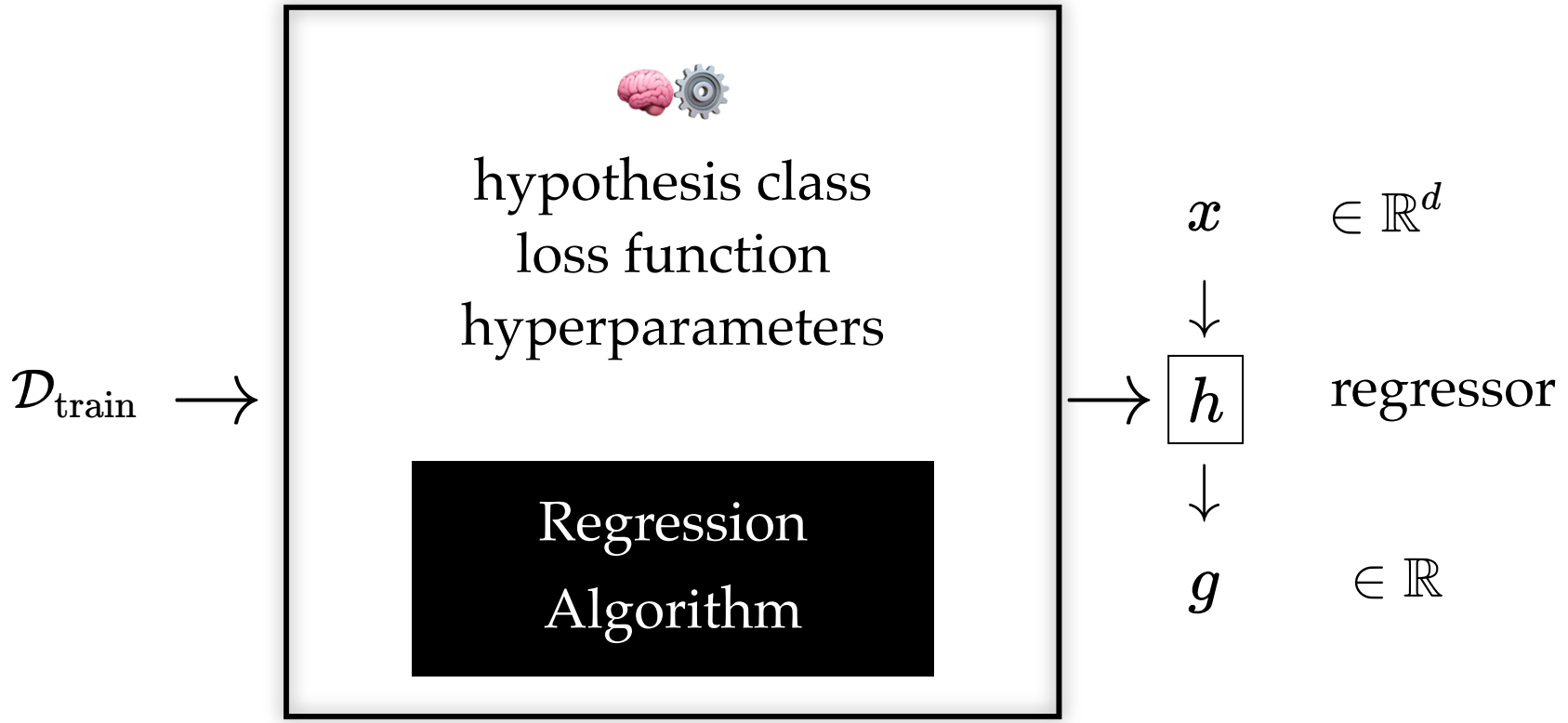
2:30pm, Room 10-250

Slides and Lecture Recording

Recall:


"Learn" a model
hypothesis class
loss function
hyperparameters
Regression Algorithm
$\mathcal{D}_{\text{train}}$
x
$\in \mathbb{R}^d$
h
regressor
g
$\in \mathbb{R}$
"Use" a model

Recall:



"Learn" a model $\longrightarrow$

train, optimize, tune, adapt ...

adjust/update/find θ

gradient-based

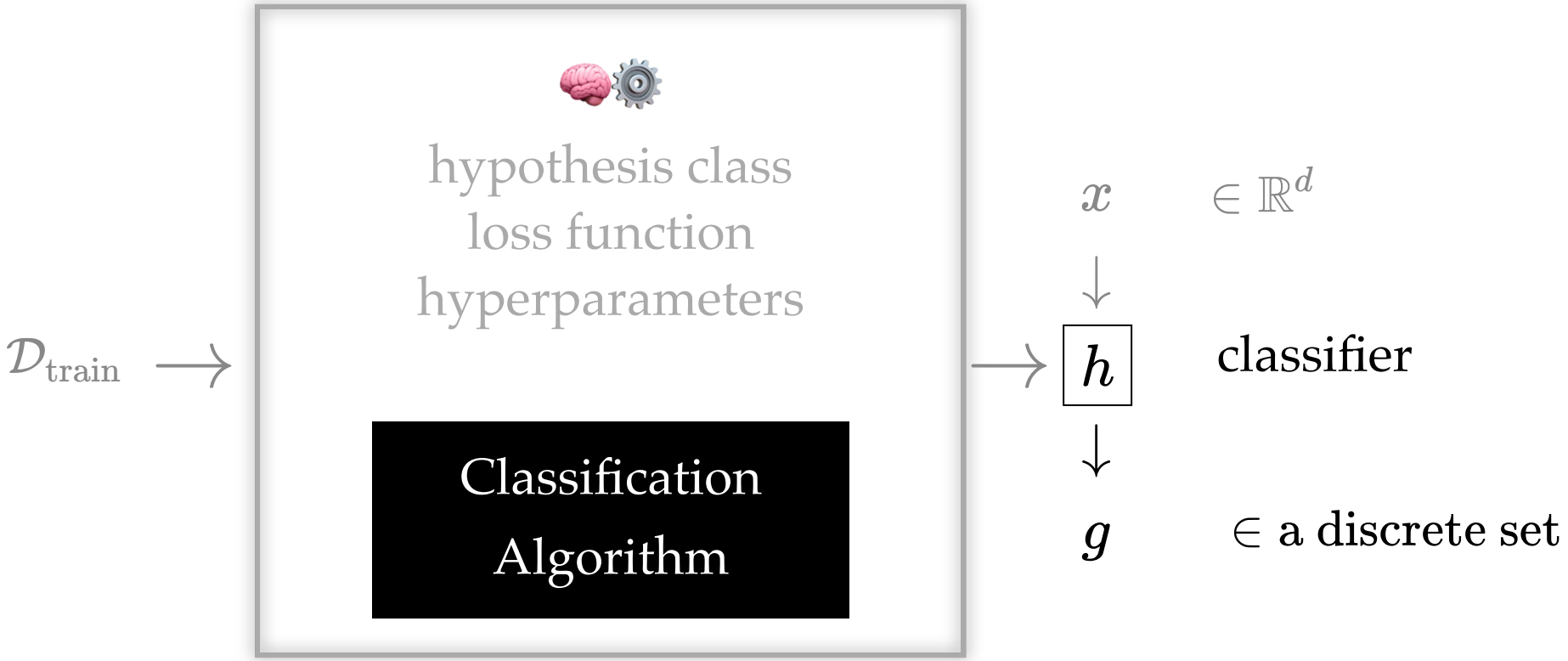
$\downarrow$ "Use" a model

predict, test, evaluate, infer ...

plug in the θ found

no gradients involved

Today:

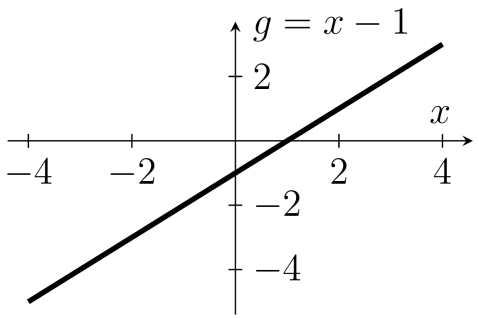

hypothesis class
loss function
hyperparameters
Classification Algorithm
$\mathcal{D}_{\text{train}}$
x
$\in \mathbb{R}^d$
h
classifier
g
$\in$ a discrete set
$\{0, 1\}$
{😍, 🥺}
{"good", "better", "best", ...}
{"fish", "grizzly", "chameleon", ...}

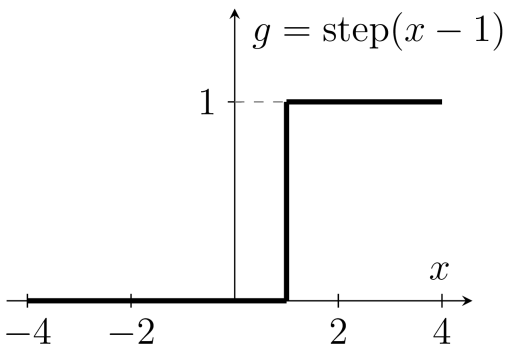
Outline

1. Linear (binary) classifiers
    - to use: **separator, normal vector**
    - to learn: no gradient to follow
2. Linear logistic (binary) classifiers
3. Linear multi-class classifiers

	linear regressor	linear binary classifier
features	$x \in \mathbb{R}^d$	
label	$y \in \mathbb{R}$	$y \in \{0, 1\}$
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$	
linear combo	$\theta^T x + \theta_0 \quad = z$	
predict	$g = z$	$g = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$

e.g., $\theta = 1,\ \theta_0 = -1$





Today, z stands for $\theta^T x + \theta_0$.

https://shenshen.mit.edu/demos/classification/separator.html

Outline

1. Linear (binary) classifiers
    - to use: separator, normal vector
    - to learn: no gradient to follow
2. Linear logistic (binary) classifiers
3. Linear multi-class classifiers

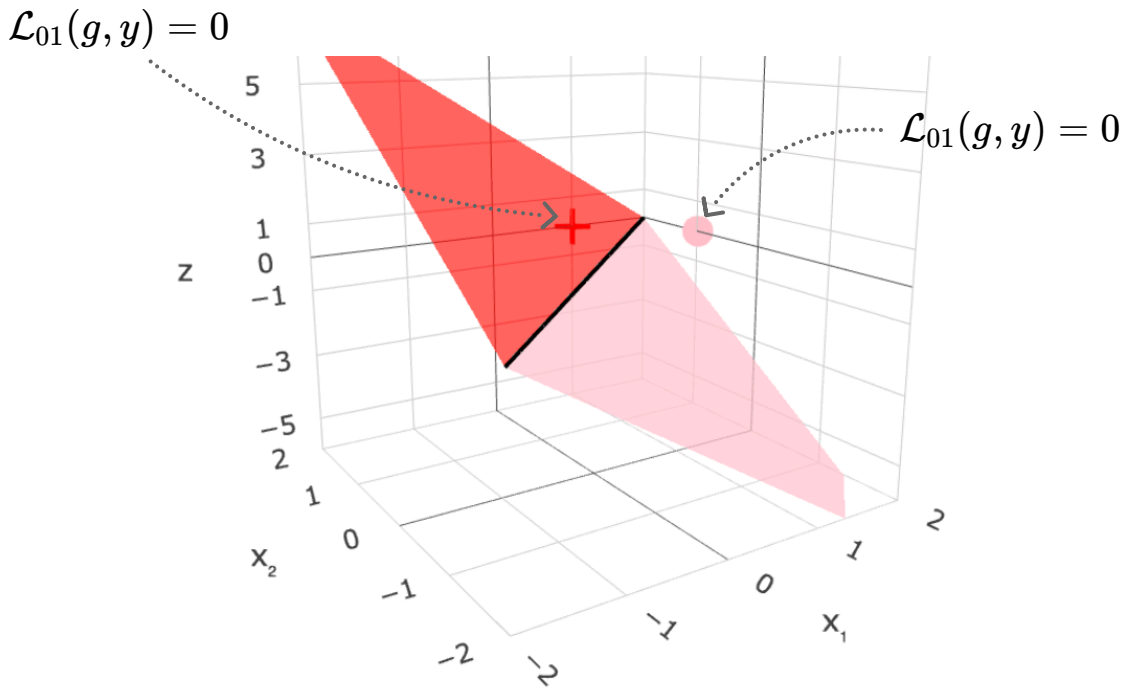
- To *learn* a model, we need a loss function.

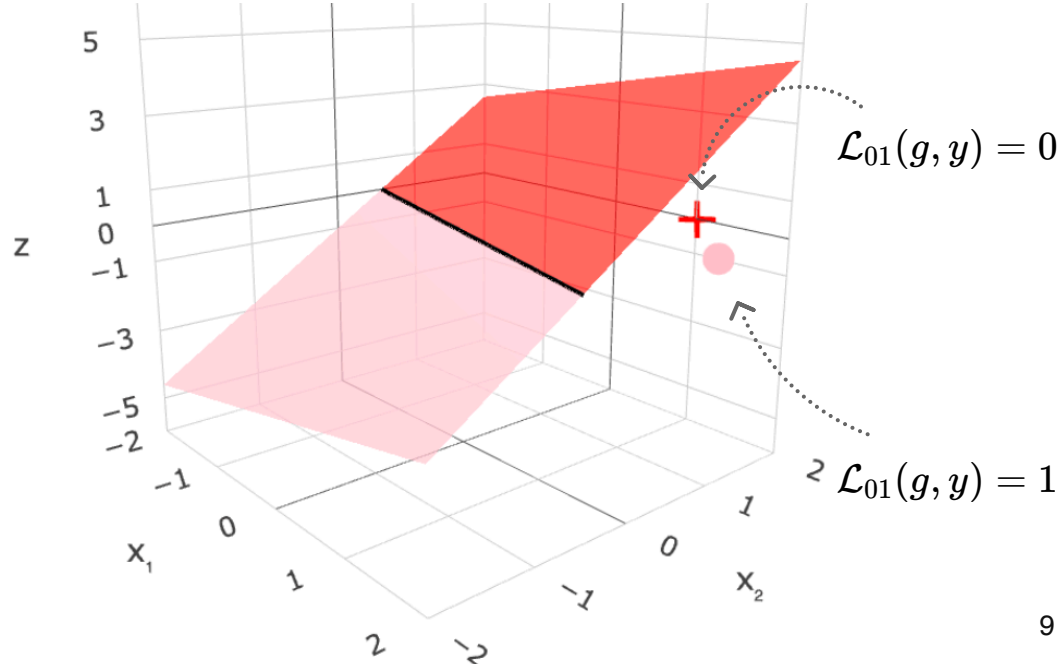
- One natural loss choice:

$$g = \text{step}(z) = \text{step}\left(\theta^T x + \theta_0\right) \qquad y$$

$$\mathcal{L}_{01}(g, y) = \begin{cases} 0 & \text{if guess} = \text{label} \\ 1 & \text{otherwise} \end{cases}$$

- Very intuitive and easy to evaluate 😍





https://shenshen.mit.edu/demos/classification/ols-01-lr.html?models=ols,01

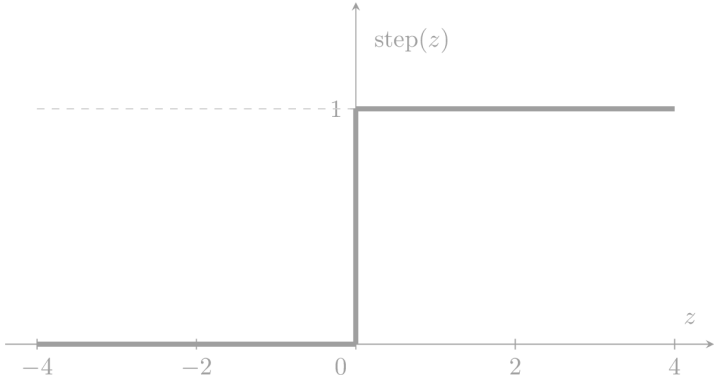
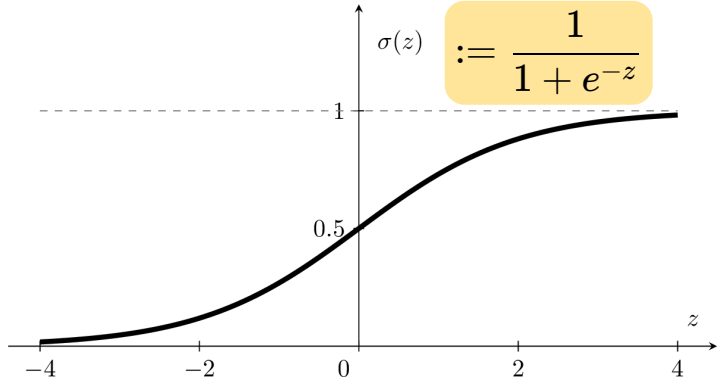
$J_{01}(\theta)$ is very hard to optimize (NP-hard) 🥺

- "Flat" almost everywhere (zero gradient $\nabla_\theta J_{01}(\theta)$)
- "Jumps" elsewhere (no gradient)

	linear regressor	linear binary classifier
features	$x \in \mathbb{R}^d$	
label	$y \in \mathbb{R}$	$y \in \{0, 1\}$
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$	
linear combo	$\theta^T x + \theta_0 \quad = z$	
predict	$g = z$	$g = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$ g is "flat" and discrete in θ
loss $\mathcal{L}(g, y)$	$\mathcal{L}_{\text{squared}} = (g - y)^2$	$\mathcal{L}_{01} = \begin{cases} 0 & \text{if } g = y \\ 1 & \text{otherwise} \end{cases}$ $\mathcal{L}_{01}(g, y)$ is "flat" and discrete in g
optimize method	• closed-form formula • gradient descent	training error almost "flat" w.r.t. θ, so the gradient gives very little info

Outline

1. Linear (binary) classifiers
2. Linear logistic (binary) classifiers
    - to use: **sigmoid**
    - to learn: **negative log-likelihood loss**
3. Linear multi-class classifiers

	linear binary classifier	linear *logistic* binary classifier
features	$x \in \mathbb{R}^d$	
label	$y \in \{0, 1\}$	
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$	
linear combo	$\theta^T x + \theta_0 \ = z$	
predict	 $\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$	 $\begin{cases} 1 & \text{if } \sigma(z) > 0.5 \\ 0 & \text{otherwise} \end{cases}$

$\sigma(z)$: *confidence*, or *estimated probability* that x belongs to the positive class

https://chenchen.mit.edu/demos/classification/step_sigmoid.html

- θ, θ_0 can flip, squeeze, expand, or shift $\sigma(\theta x + \theta_0)$ *horizontally*
- $\sigma(z)$ is monotonic and has a tidy gradient (derived in hw/lab)

linear *logistic* binary classifier

features: $x \in \mathbb{R}^d$

parameters: $\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$

the *logit* z: $$z = \theta^T x + \theta_0$$

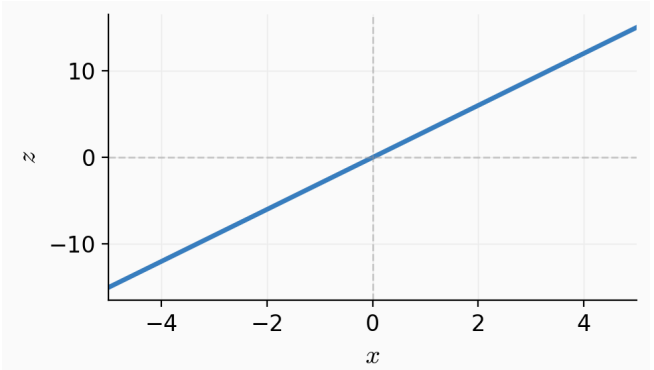
apply *sigmoid*:

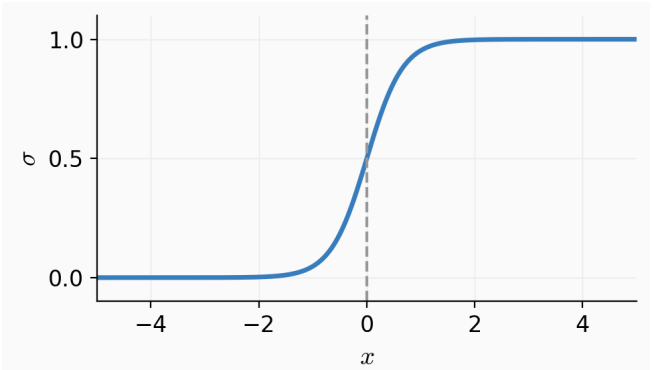
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

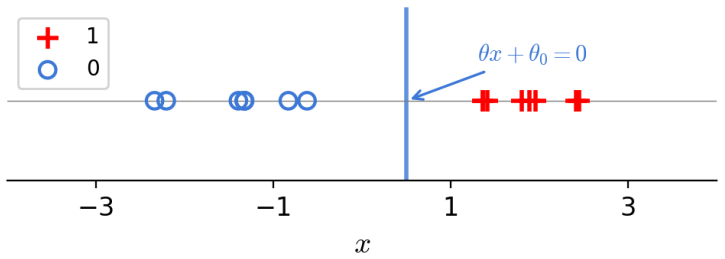
Predict 1 if $\sigma(z) > 0.5$, else 0.

$$\sigma(z) = 0.5 \iff z = 0 \iff \theta^T x + \theta_0 = 0$$

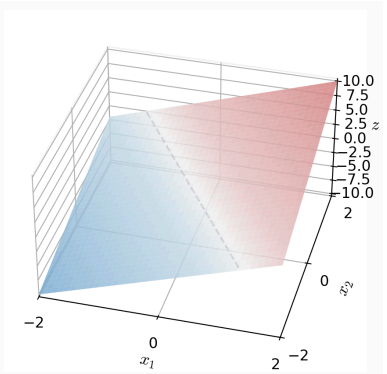
e.g., one feature

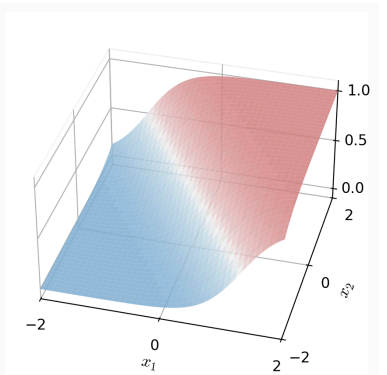


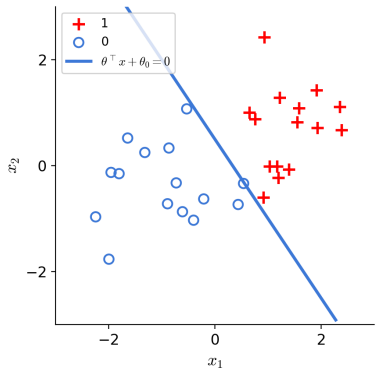




e.g., two features





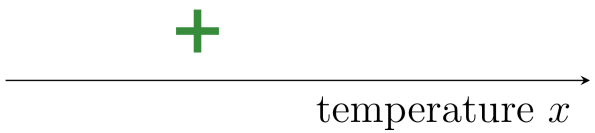


the separator is *linear* in x!

Outline

1. Linear (binary) classifiers
2. Linear **logistic** (binary) classifiers
    - to use: **sigmoid**
    - to learn: **negative log-likelihood loss**
3. Linear multi-class classifiers

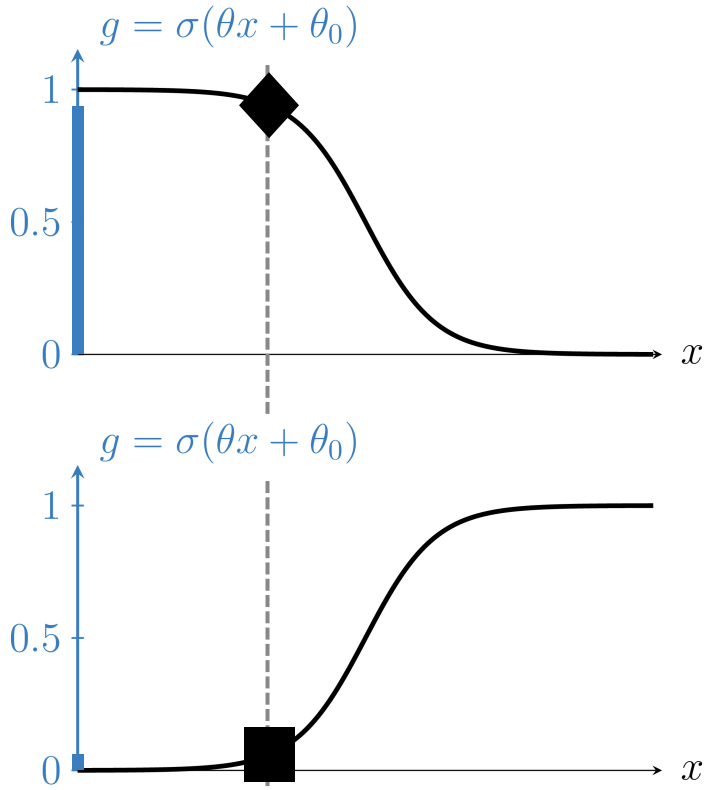
One training point, label $y = 1$

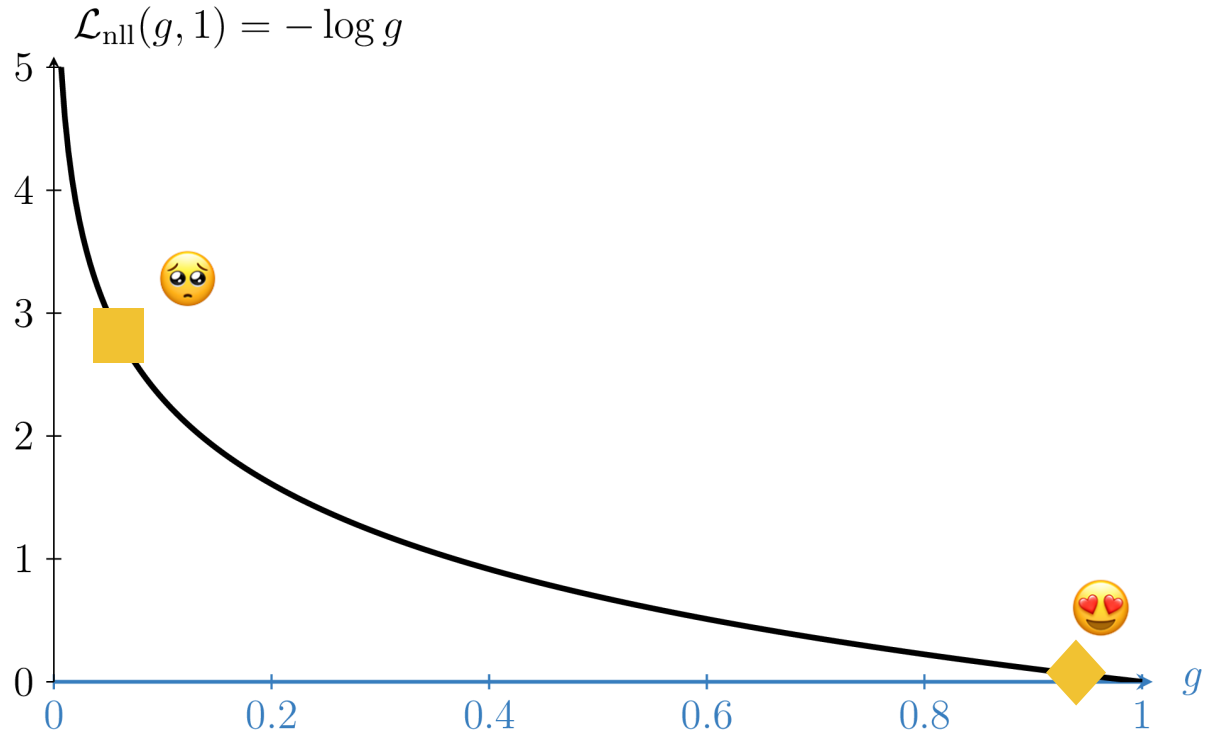


A smooth loss $\mathcal{L}(g, y)$ that shrinks as g nears y?

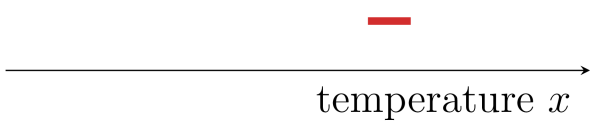
negative log likelihood

$$\mathcal{L}_{\text{nll}}(g, 1) := -\log g$$



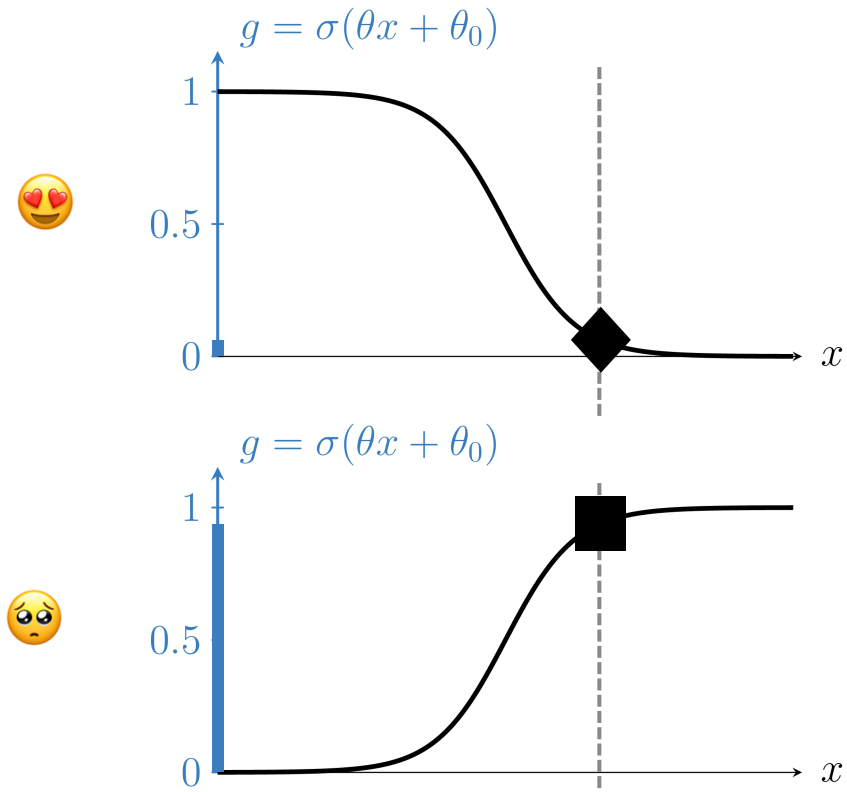


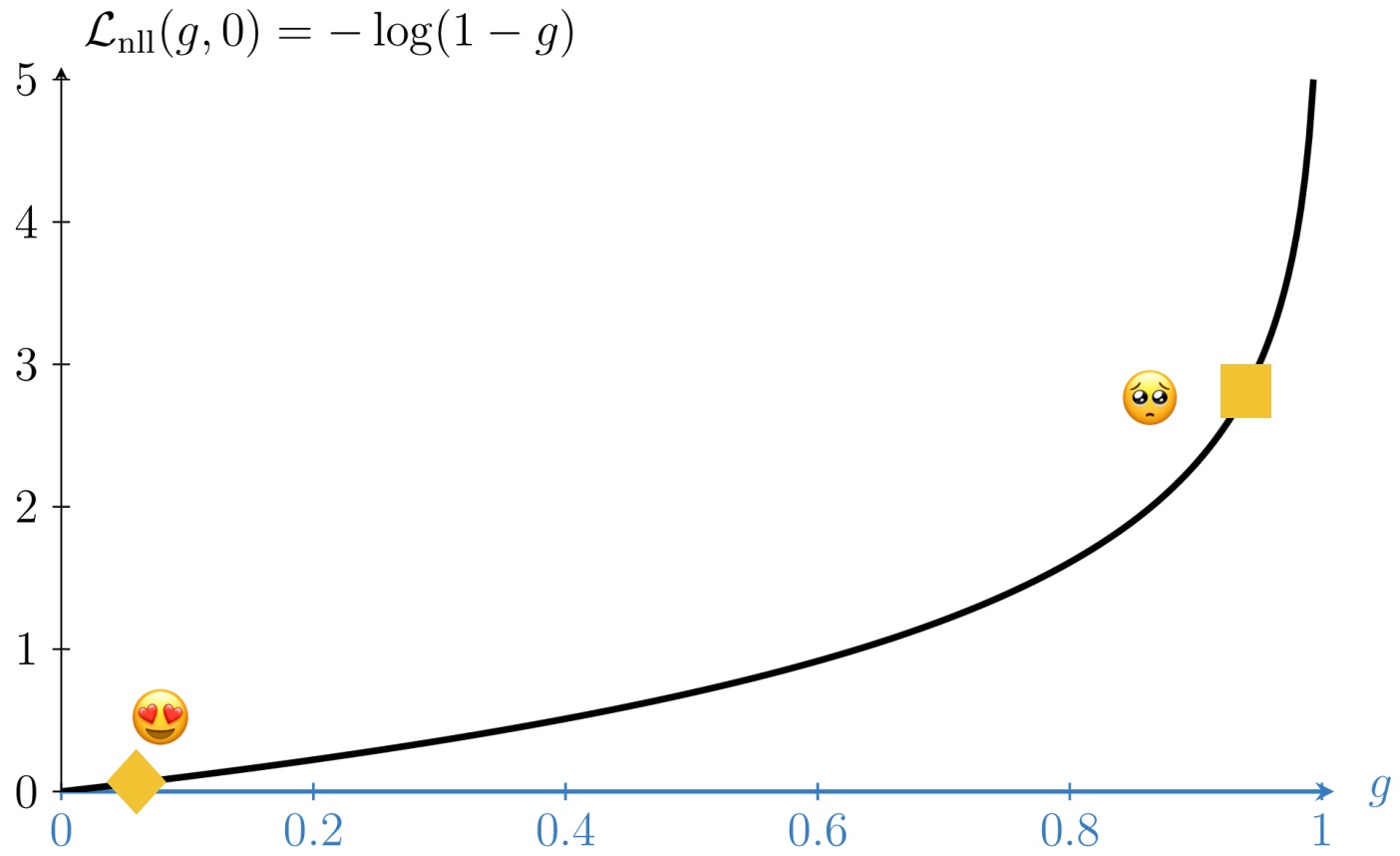
One training point, label $y = 0$



$1 - g$: the model's *estimated probability* that x belongs to the **negative** class.

$$\mathcal{L}_{\text{nll}}(g, 0) := -\log(1 - g)$$





https://shenshen.mit.edu/demos/classification/nll.html

Because the true label $\boldsymbol{y}$ is $\mathbf{0}$ or $\mathbf{1}$,

$$\mathcal{L}_{\text{nll}}(g, y) = \begin{cases} -\log(g) & \text{if } y = 1 \\ -\log(1-g) & \text{if } y = 0 \end{cases} \Leftrightarrow \boxed{-\left[y \log g + (1-y) \log(1-g)\right]}$$

- When $y = 1$: $-\left[y \log g + (1-y) \log (1-g)\right] = -\log g$
- When $y = 0$: $-\left[y \log g + (1-y) \log (1-g)\right] = -\left[\log (1-g)\right]$

Read as: $\sum$ (true label for class k) $\cdot -\log$(estimated probability of class k).

Since $y \in \{0, 1\}$, only the true class's term survives.

	linear binary classifier	linear *logistic* binary classifier
features	$x \in \mathbb{R}^d$	
label	$y \in \{0, 1\}$	
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$	
linear combo	$\theta^T x + \theta_0 = z$	
predict	$\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 & \text{if } g = \sigma(z) > 0.5 \\ 0 & \text{otherwise} \end{cases}$
loss	$\mathcal{L}_{01} = \begin{cases} 0 & \text{if } g = y \\ 1 & \text{otherwise} \end{cases}$	$\mathcal{L}_{\text{nll}} = \begin{cases} -\log(g) & \text{if } y = 1 \\ -\log(1-g) & \text{if } y = 0 \end{cases}$ $\Leftrightarrow -\left[y \log g + (1-y) \log(1-g)\right]$
optimize via	no efficient method (NP-hard)	gradient descent

https://shenshen.mit.edu/demos/classification/ols-01-lr.html?models=01,lr&embed

One training point, $x = 1$, label $y = 1$

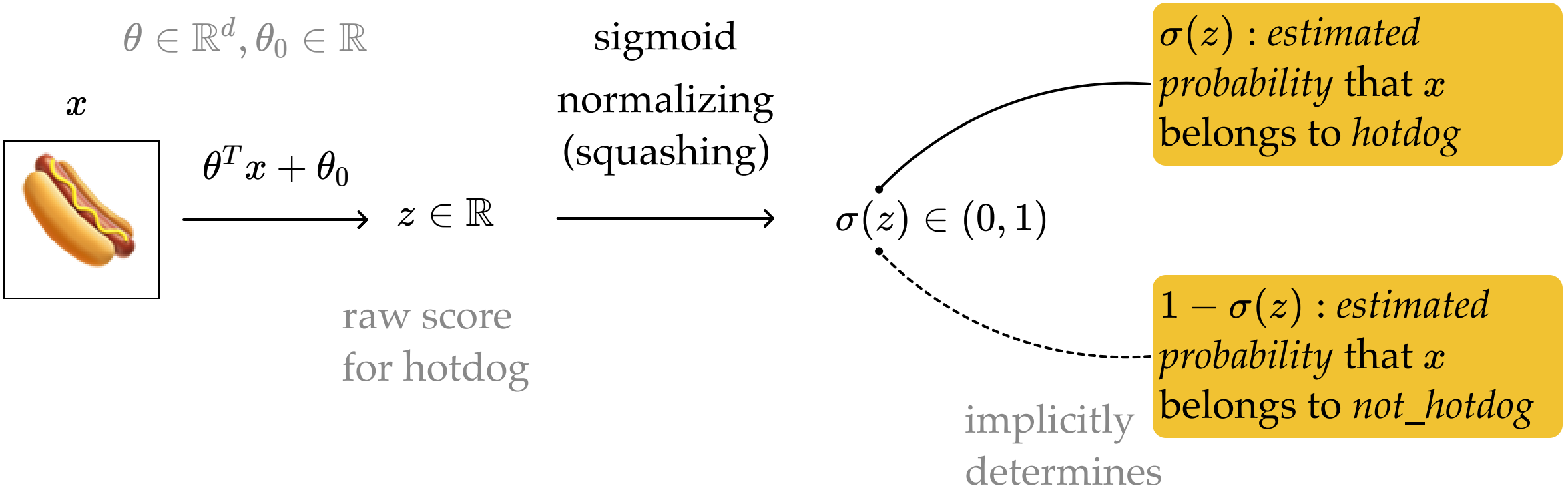
https://shenshen.mit.edu/demos/classification/nlloverfit.html

- If the data set is linearly separable, the logistic loss has no finite minimizing θ.
- In theory, θ grows without bound, so the model gets overconfident.
- It is common to add a ridge penalty $\lambda \|\theta\|^2$.

Outline

1. Linear (binary) classifiers
2. Linear logistic (binary) classifiers
3. Linear multi-class classifiers
    - to use: **softmax**
    - to learn: **one-hot encoding, cross-entropy loss**

for two classes, {*hotdog*, *not_hotdog*}, one scalar logit z suffices



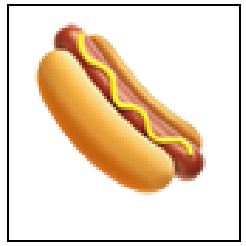
for $K > 2$ classes, one scalar z no longer suffices, so we use K logits, one per class

for K classes, use K logit scores.

e.g. $K = 3$: {*hotdog, pizza, veggie*}

$\theta \in \mathbb{R}^{d \times K}$, $\theta_0 \in \mathbb{R}^K$

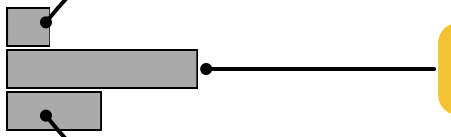
x



$\theta^T x + \theta_0$

$z \in \mathbb{R}^3$

normalizing (squashing)



estimated probability that x belongs to *hotdog*

… to *pizza*

… to *veggie*

$\text{softmax}(z) \in \mathbb{R}^3$

in general K logits

one raw score per class

softmax: K logits to a distribution over K classes $\mathbb{R}^K \to \mathbb{R}^K$

$$\text{softmax}(z) := \begin{bmatrix} \frac{\exp(z_1)}{\sum_{k=1}^{K} \exp(z_k)} \\ \vdots \\ \frac{\exp(z_K)}{\sum_{k=1}^{K} \exp(z_k)} \end{bmatrix}$$

outputs lie in $(0, 1)$ and sum to 1

e.g.,

$$\text{softmax}\left(\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}\right) = \begin{bmatrix} \frac{e^1}{e^1+e^2+e^3} \\ \frac{e^2}{e^1+e^2+e^3} \\ \frac{e^3}{e^1+e^2+e^3} \end{bmatrix} = \begin{bmatrix} 0.0900 \\ 0.2447 \\ 0.6653 \end{bmatrix}$$

max among the K logits

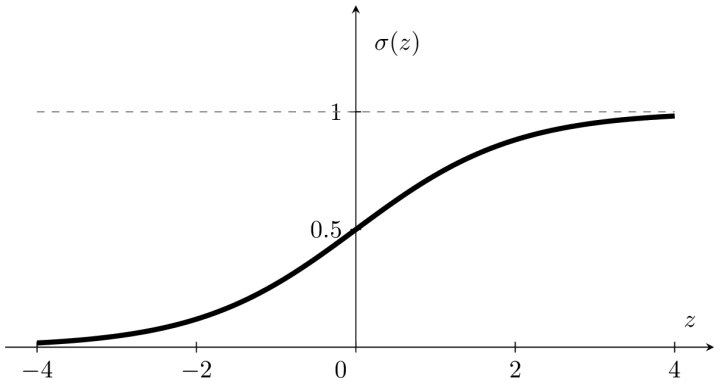
"soft" max: the largest logit gets the largest probability

sigmoid $\mathbb{R} \to \mathbb{R}$

$$\sigma(z) := \frac{1}{1+\exp(-z)} = \frac{\exp(z)}{\exp(z)+\exp(0)}$$

implicit logit for the negative class

predict positive if $\sigma(z) > 0.5 = \sigma(0)$



softmax: $\mathbb{R}^K \to \mathbb{R}^K$

$$\text{softmax}(z) := \begin{bmatrix} \frac{\exp(z_1)}{\sum_{k=1}^K \exp(z_k)} \\ \vdots \\ \frac{\exp(z_K)}{\sum_{k=1}^K \exp(z_k)} \end{bmatrix}$$

predict the class with the largest softmax score

unifying rule: predict the class with the largest logit (= largest softmax score)

	linear logistic *binary* classifier	one-out-of-K classifier
features	$x \in \mathbb{R}^d$	
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$	$\theta \in \mathbb{R}^{d \times K},\ \theta_0 \in \mathbb{R}^K$
linear combo	$\theta^T x + \theta_0 = z \in \mathbb{R}$	$\theta^T x + \theta_0 \ = z \in \mathbb{R}^K$
predict	$\sigma(z) = \dfrac{\exp(z)}{\exp(0) + \exp(z)}$	$\text{softmax}(z) = \begin{bmatrix} \frac{\exp(z_1)}{\sum_{k=1}^K \exp(z_k)} \\ \vdots \\ \frac{\exp(z_K)}{\sum_{k=1}^K \exp(z_k)} \end{bmatrix}$
	predict positive if $\sigma(z) > \sigma(0)$	predict the class with the largest softmax score

Outline

1. Linear (binary) classifiers
2. Linear logistic (binary) classifiers
3. Linear multi-class classifiers
    - to use: **softmax**
    - to learn: **one-hot encoding, cross-entropy loss**

One-hot encoding:

- Generalizes the binary labels $\{0, 1\}$
- Encodes the K classes as an $\mathbb{R}^K$ vector, with a single **one** (*hot*) and zeros elsewhere

Training data

x	y
	"hotdog"
	"pizza"
	"veggie"
	"veggie"
⋮	

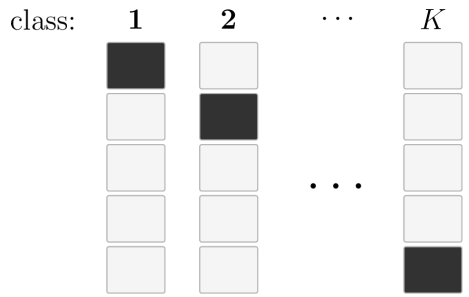
$K = 3$

$\longrightarrow$

Training data

x	y
	$\begin{bmatrix}1\\0\\0\end{bmatrix}$
	$\begin{bmatrix}0\\1\\0\end{bmatrix}$
	$\begin{bmatrix}0\\0\\1\end{bmatrix}$
	$\begin{bmatrix}0\\0\\1\end{bmatrix}$
⋮	

in general, for K classes:



Negative log-likelihood multi-class loss (also called cross-entropy)

g : softmax output

g_k : estimated probability that x belongs to class k

$$\mathcal{L}_{\text{nllm}}(g, y) = -\sum_{k=1}^{K} y_k \cdot \log(g_k)$$

y : one-hot encoding label

y_k : kth entry in y, either 0 or 1

- Generalizes the binary negative log-likelihood loss

$$\mathcal{L}_{\text{nll}}(g, y) = -[y \log g + (1 - y) \log(1 - g)]$$

- Only the true class's term survives the K-term sum, since every other $y_k = 0$

end-to-end pipeline

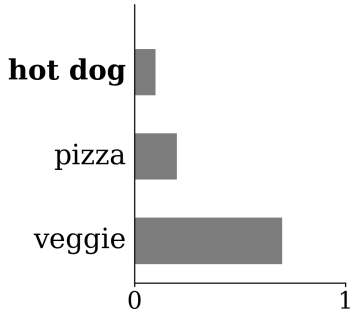
current prediction
$g = \text{softmax}(z)$

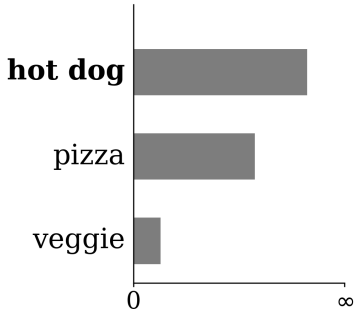
$\xrightarrow{-\log}$

$-\log(g)$

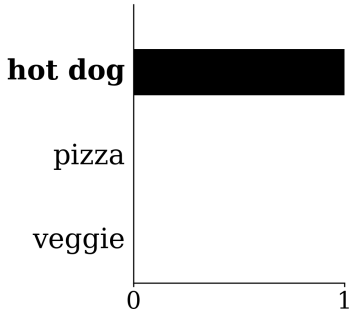
y (true label)

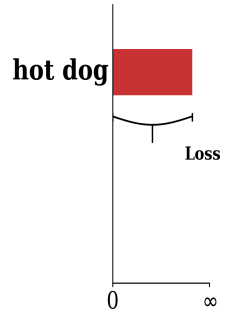
$$\mathcal{L}_{\text{nllm}} = -\sum_k y_k \cdot \log(g_k)$$





$\odot$





$= [0.1,\ 0.2,\ 0.7]$

$\approx [2.30,\ 1.61,\ 0.36]$

$= [1,\ 0,\ 0]$

$\text{Loss} \approx 2.30$

To reduce the loss, g_{hotdog} needs to go up.
That signal flows smoothly back to θ through $-\log$ and softmax, so gradient descent can train θ.

contrived predictions

end-to-end pipeline

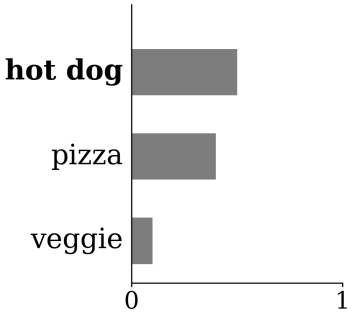
current prediction
$g = \text{softmax}(z)$

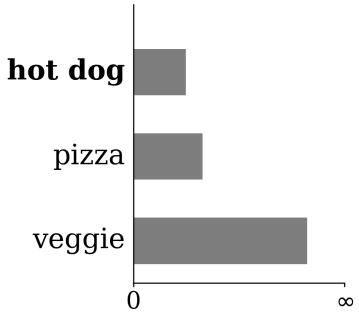
$\xrightarrow{-\log}$

$-\log(g)$

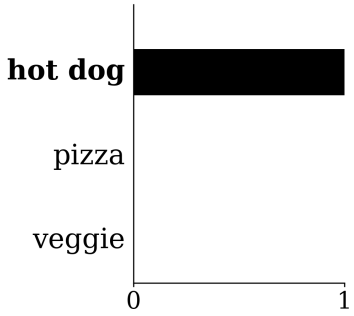
y (true label)

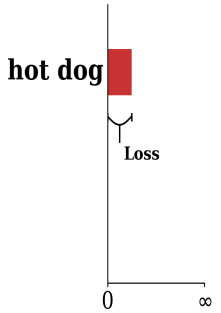
$$\mathcal{L}_{\text{nllm}} = -\sum_k y_k \cdot \log(g_k)$$





$\odot$





$= [0.5,\ 0.4,\ 0.1]$

$\approx [0.69,\ 0.92,\ 2.30]$

$= [1,\ 0,\ 0]$

$\text{Loss} \approx 0.69$

contrived predictions

Summary

$$\mathcal{L}_{\text{nll}}(g, y) = -\left[y \log g + (1 - y) \log(1 - g)\right], \quad g = \sigma(\theta^T x + \theta_0)$$

Linear (binary) classifiers

separator | normal vector | 0-1 loss

Linear logistic (binary) classifiers

sigmoid | logit | negative log-likelihood | linearly separable

Linear multi-class classifiers

softmax | one-hot encoding | cross-entropy

Summary

- **Linear (binary) classifiers**
  - to use: predict 1 when $z = \theta^T x + \theta_0 > 0$; the separator $z = 0$ is a hyperplane with normal θ.
  - to learn: the 0-1 loss is flat in θ, so there is no gradient to follow.
- **Linear logistic (binary) classifiers**
  - to use: $g = \sigma(z)$ is a probability, and $g > 0.5$ exactly when $z > 0$, so the separator stays linear.
  - to learn: the negative log-likelihood is smooth, so gradient descent works; on separable data, a ridge penalty keeps θ finite.
- **Linear multi-class classifiers**
  - to use: softmax turns one score per class into a distribution; sigmoid is its two-class case.
  - to learn: one-hot labels and the cross-entropy loss, $-$ log of the true class's probability.