https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 4: Linear Classification

Shen Shen

Feb 23, 2026

Async (Snow Day)

Slides and Lecture Recording

Recap:

"Learn" a model

$\longrightarrow$

hypothesis class
loss function
hyperparameters

Regression
Algorithm

$\mathcal{D}_{\text{train}} \longrightarrow$ [Regression Algorithm] $\longrightarrow \boxed{h}$ regressor

$x \quad \in \mathbb{R}^d$

$\downarrow$

$\boxed{h}$

$\downarrow$

$g \quad \in \mathbb{R}$

$\downarrow$ "Use" a model

Recap:

hypothesis class
loss function
hyperparameters

Regression Algorithm

$\mathcal{D}_{\text{train}} \longrightarrow$ [hypothesis class, loss function, hyperparameters, Regression Algorithm] $\longrightarrow \boxed{h}$ regressor

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

$\downarrow$

$\boxed{h}$

$\downarrow$

$g \in \mathbb{R}$

"Learn" a model $\longrightarrow$

train, optimize, tune, adapt ...

adjusting/updating/finding θ

gradient based

$\downarrow$ "Use" a model

predict, test, evaluate, infer ...

plug in the θ found

no gradients involved

Today:

$\mathcal{D}_{\text{train}} \longrightarrow$

🧠⚙️

hypothesis class

loss function

hyperparameters

Classification Algorithm

$\longrightarrow \boxed{h}$ classifier

$x \quad \in \mathbb{R}^d$

$\downarrow$

$\boxed{h}$ classifier

$\downarrow$

$g \quad \in$ a discrete set

$\{0, 1\}$

{😍, 🥺}

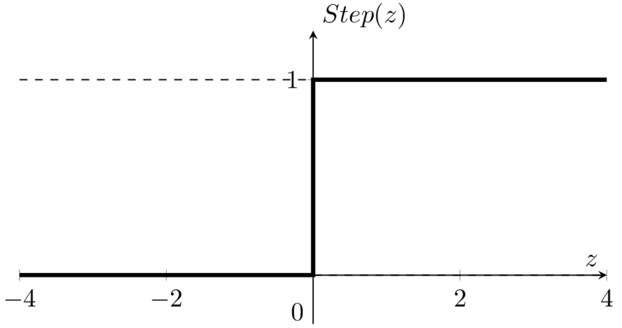
{"good", "better", "best", ...}

{"Fish", "Grizzly", "Chameleon", ...}

Outline

1. Linear (binary) classifiers
    - to use: **separator, normal vector**
    - to learn: very difficult!
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 combination	$\theta^T x + \theta_0 \quad = z$	
predict	$g = z$	$g = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$



today, we refer to $\theta^T x + \theta_0$ as z throughout.

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

Outline

1. Linear (binary) classifiers
    - to use: separator, normal vector
    - to learn: very difficult!
2. Linear logistic (binary) classifiers
3. Linear multi-class classifiers

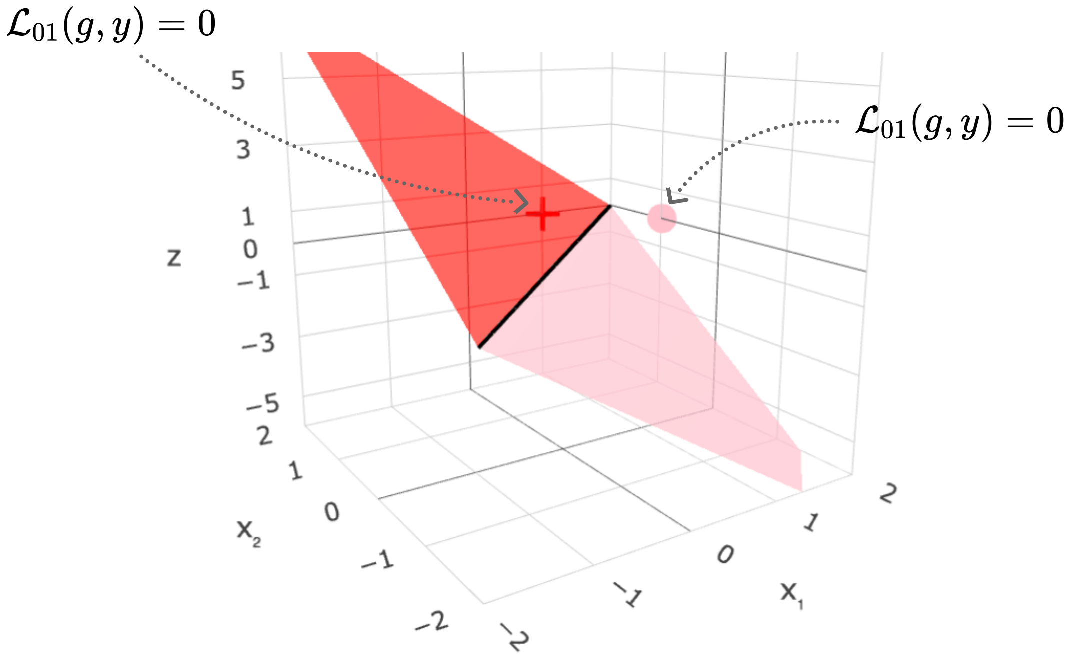
- To *learn* a model, need a loss function.

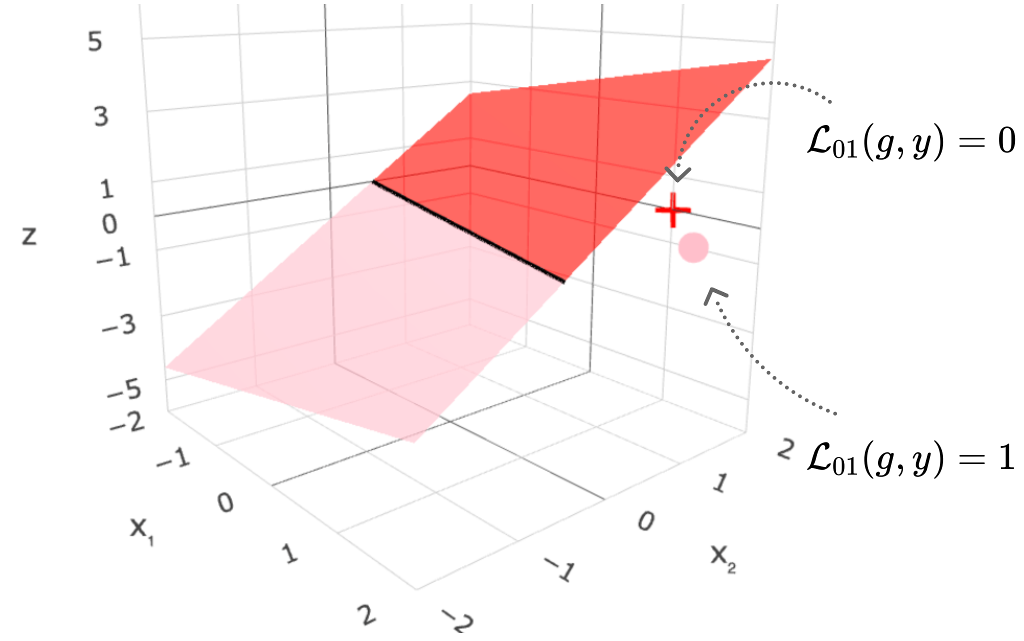
- One natural loss choice:

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

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

- Very intuitive, and easy to evaluate 😍





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

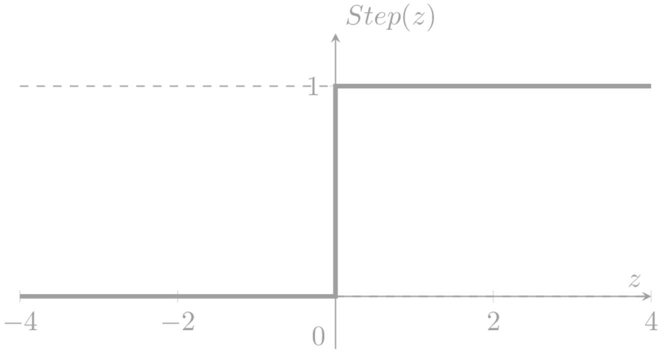
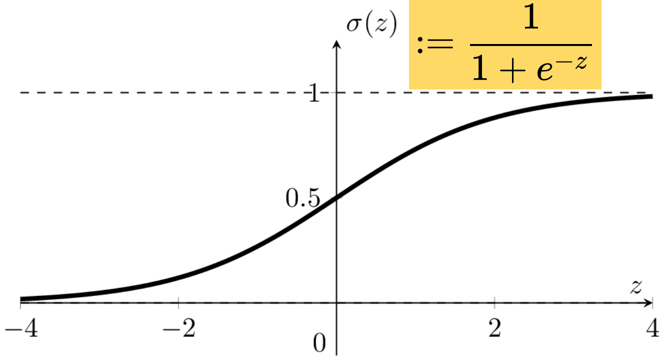
$J_{01}(\theta)$ very hard to optimize (NP-hard) 🥺

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

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

Sigmoid $\sigma(\cdot)$: *confidence* or *estimated likelihood* that x belongs to the positive class https://shenshen.mit.edu/demos/classification/step_sigmoid.html

- θ, θ_0 can flip, squeeze, expand, or shift the $\sigma(x)$ graph *horizontally*
- $\sigma(\cdot)$ monotonic, very elegant gradient (see 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:

$$z = \theta^\top x + \theta_0$$

apply *sigmoid*:

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

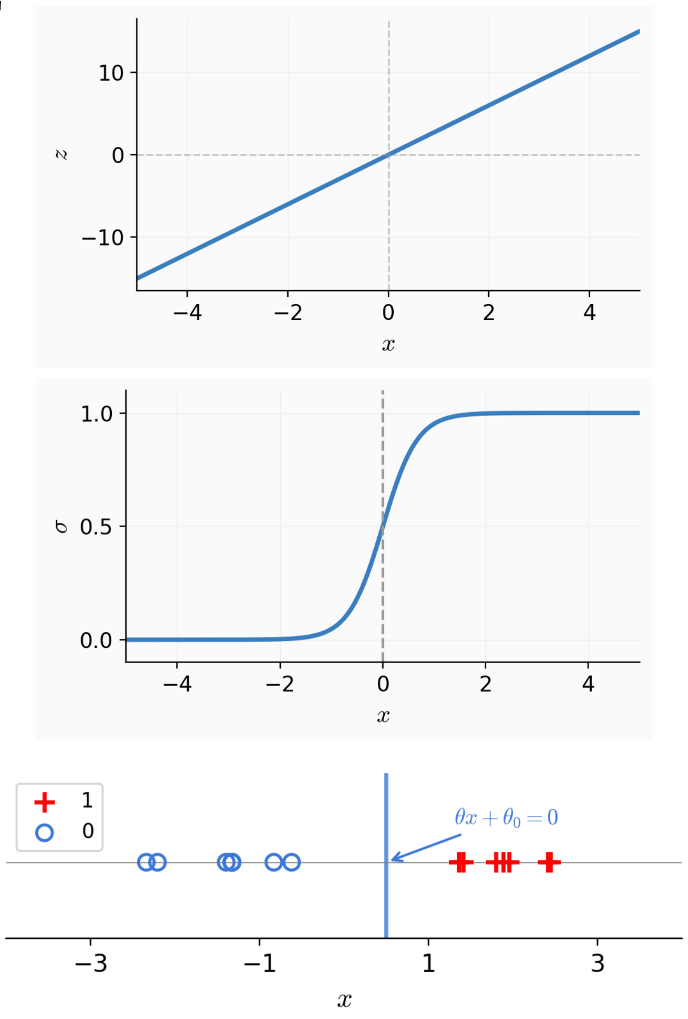
Predict 1 if $\sigma(z) > 0.5$, else 0.

$$\sigma(z) = 0.5 \iff z = 0$$

$$\iff \theta^\top x + \theta_0 = 0$$

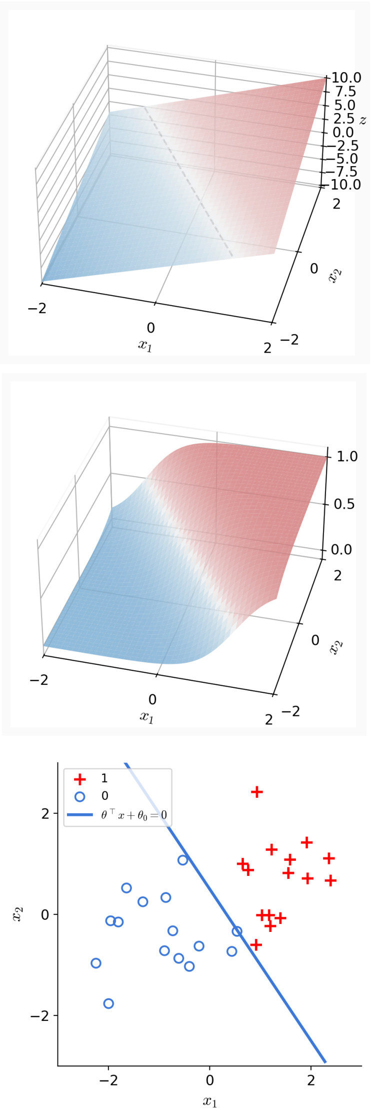
Text

1d feature



Text

2d features

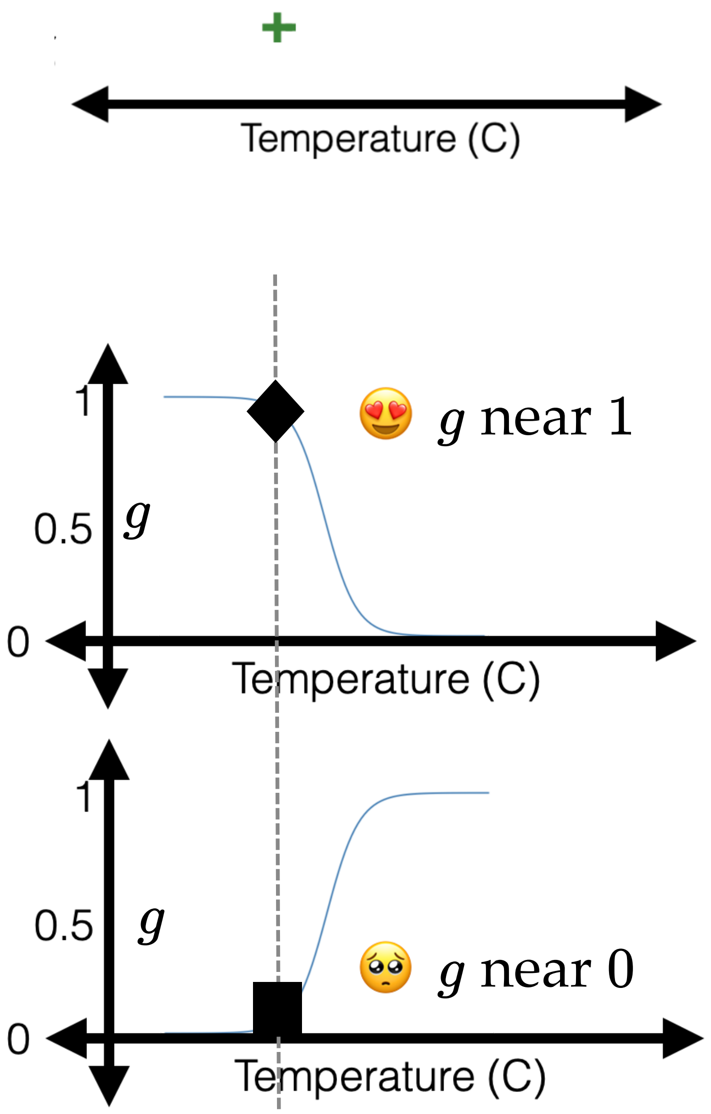


separator is *linear* in feature x!

Outline

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

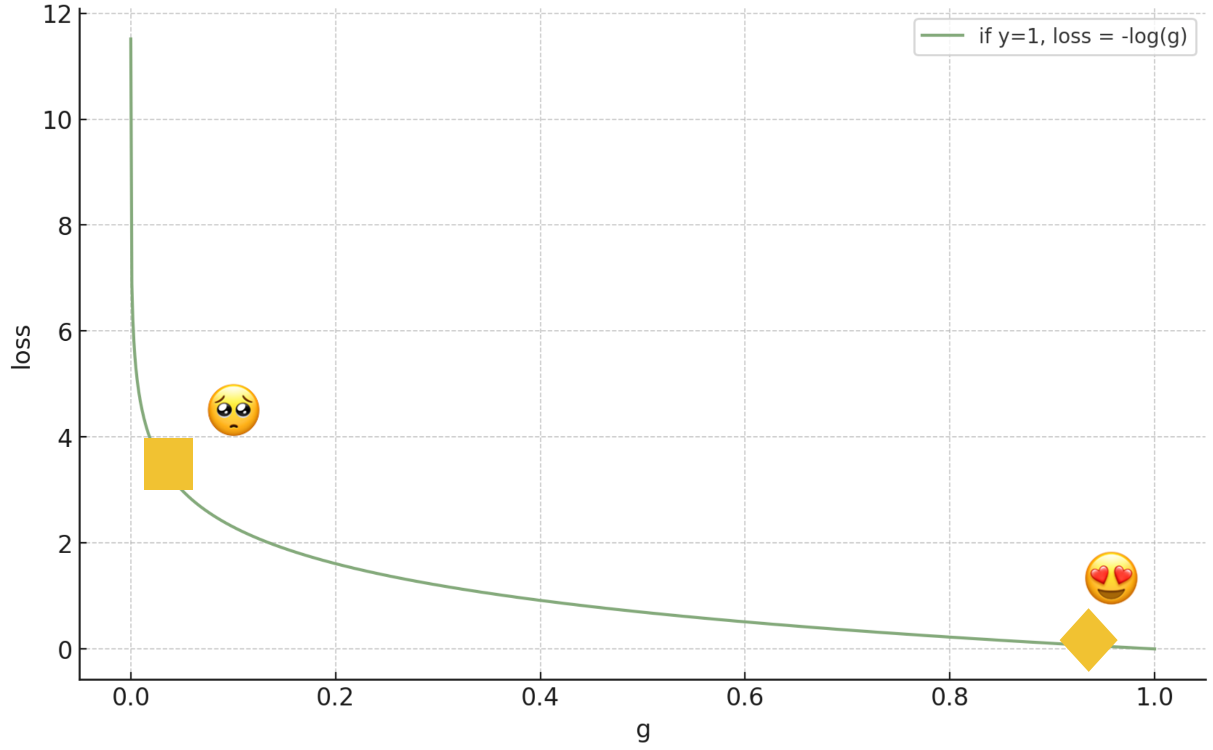
Example: a single training data from $y = 1$ class



want a smooth loss $\mathcal{L}(g, y)$ to reward g closer to y?

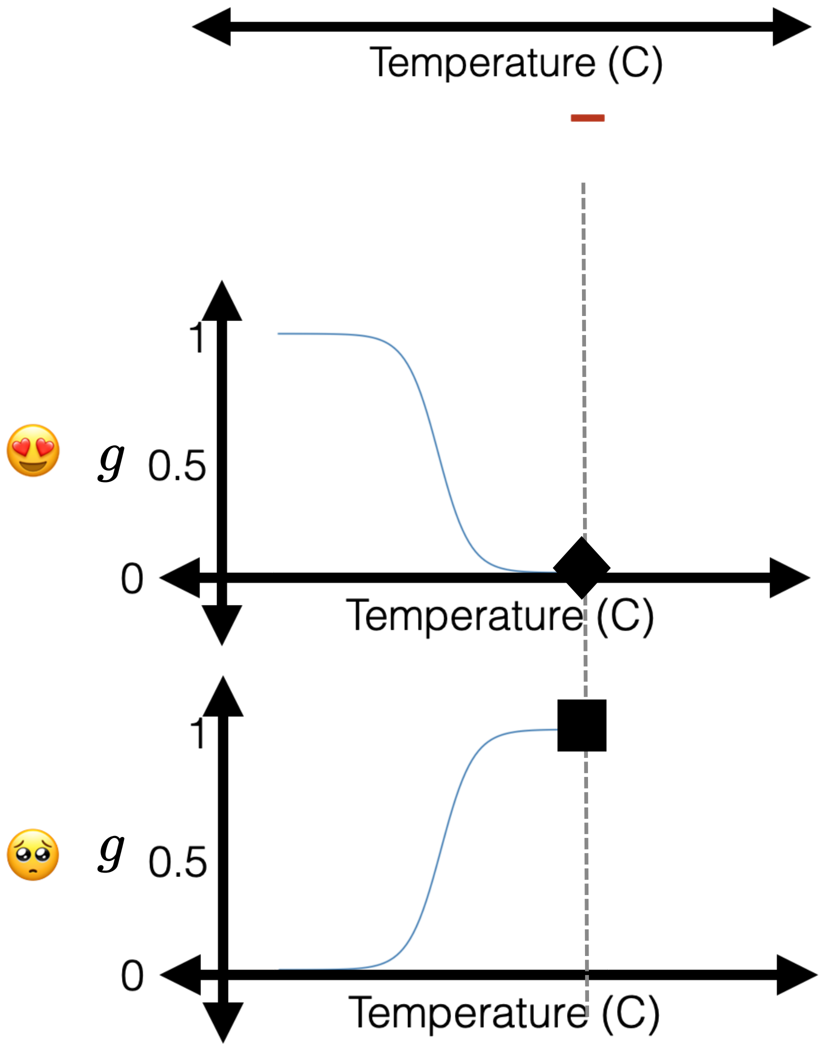
negative log likelihood

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

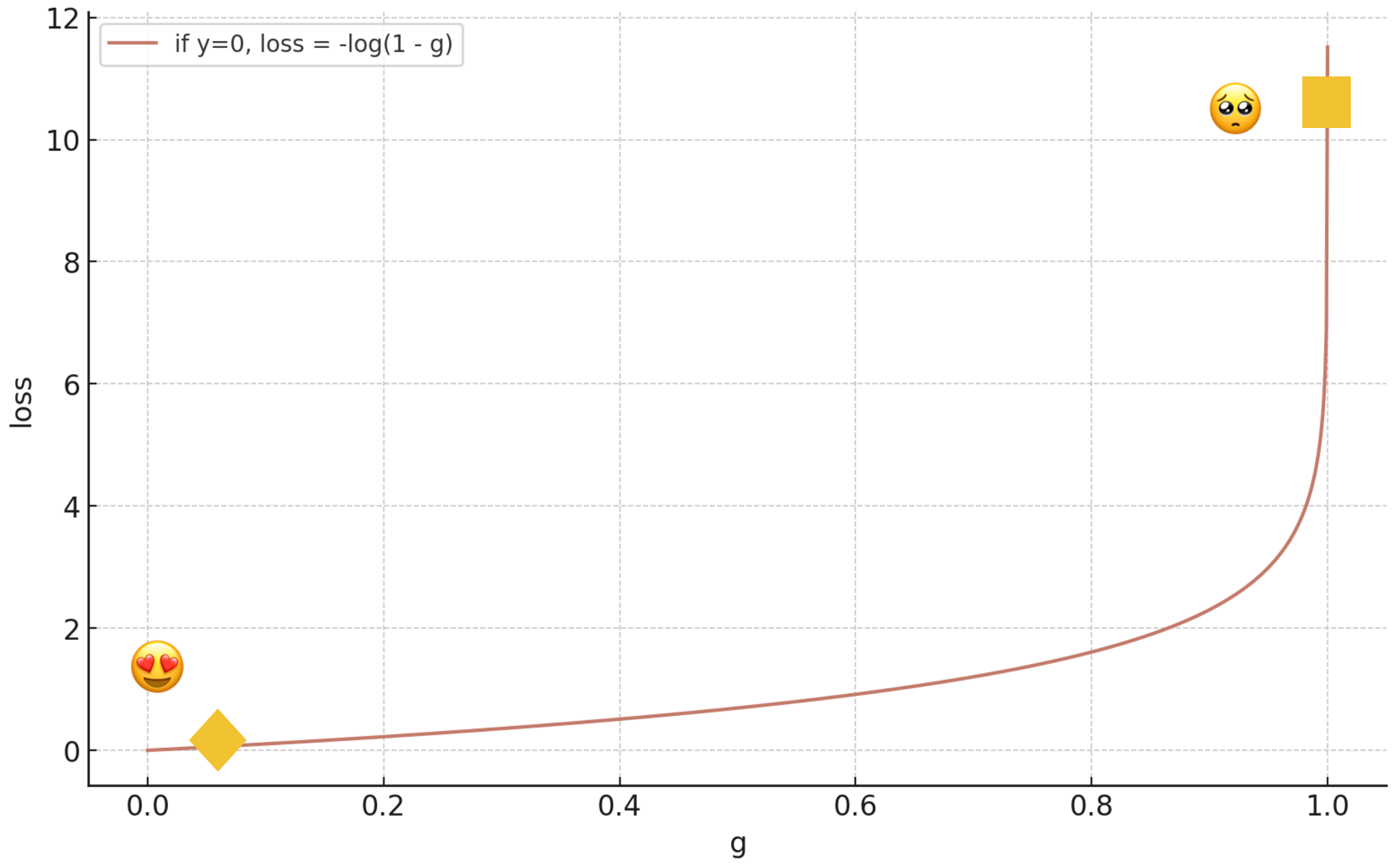


Example: a single training data from $y = 0$ class

$1 - g = 1 - \sigma(\cdot)$: the model's *predicted likelihood* 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 actual label $\boldsymbol{y \in \{0, 1\}}$,

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

- 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(predicted prob 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	$\begin{cases} 0 & \text{if } g = y \\ 1 & \text{otherwise} \end{cases}$	$\begin{cases} -\log(g) & \text{if } y = 1 \\ -\log(1-g) & \text{if } y = 0 \end{cases}$ $\Leftrightarrow -\left[y \log g + (1-y)\log(1-g)\right]$
optimize via	NP-hard to learn	gradient descent

training data: $x = 1, y = 1$

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

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

- If the data set is linearly separable, nll has no minimizer
- in theory, θ tends to have large magnitude => overly confident
- common to add ridge penalty $\lambda||\theta||^2$

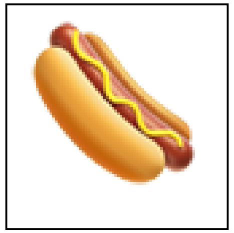
Outline

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**

to predict {*hotdog*, or *not_hotdog*}, a scalar logit z suffices

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

x



$\xrightarrow{\theta^T x + \theta_0} z \in \mathbb{R} \xrightarrow{\text{normalizing (squashing)}} \sigma(z) \in \mathbb{R}$

raw score
for hotdog

$\sigma(\cdot)$: *predicted probability* that x belongs to *hotdog*

implicitly
determines

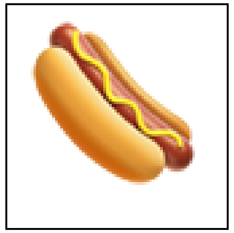
$1 - \sigma(\cdot)$: *predicted probability* that x belongs to *not_hotdog*

for $K > 2$ classes, a single scalar z no longer suffices — use K logit scores to keep track for K classes, use K logit scores.

e.g. $K = 3$: $\{$*hot-dog, pizza, veggie*$\}$

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

x



$\theta^T x + \theta_0$

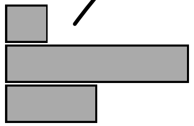
$z \in \mathbb{R}^3$

K logits

one raw score
each category

normalizing
(squashing)

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



predicted probability that x belongs to *hotdog*

predicted probability that x belongs to *pizza*

predicted probability that x belongs to *veggie* softmax: $\mathbb{R}^K \to \mathbb{R}^K$

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

outputs all $\in [0, 1]$, sum to 1

e.g.

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

max among the K logits

"soft" max'd in the output

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

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

predict the category with the highest softmax score

unifying rule: predict the class with the largest logit

	linear logistic *binary* classifier	one-out-of-K classifier
features	$x \in \mathbb{R}^d$	
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$	$\boldsymbol{\theta} \in \mathbb{R}^{d \times K},\ \boldsymbol{\theta}_0 \in \mathbb{R}^K$
linear combo	$\theta^T x + \theta_0 = z \in \mathbb{R}$	$\boldsymbol{\theta}^T \boldsymbol{x} + \boldsymbol{\theta}_0 = \boldsymbol{z} \in \mathbb{R}^K$
predict	$\sigma(z) = \dfrac{\exp(z)}{\exp(0) + \exp(z)}$ predict positive if $\sigma(z) > \sigma(0)$	$\mathrm{softmax}(z) = \begin{bmatrix} \frac{\exp(z_1)}{\sum_{k=1}^K \exp(z_k)} \\ \vdots \\ \frac{\exp(z_K)}{\sum_{k=1}^K \exp(z_k)} \end{bmatrix}$ predict the class with the highest 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 from $\{0, 1\}$ binary labels
- Encode the K classes as an $\mathbb{R}^K$ vector, with a single 1 (*hot*) and 0s elsewhere

Training data

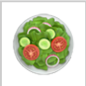
x	y
( ,	"hot-dog")
( ,	"pizza")
( ,	"veggie")
( ,	"veggie")

$\vdots$

$K = 3$

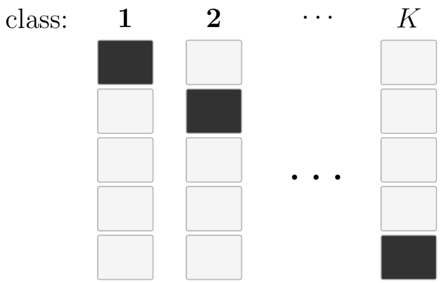
$\longrightarrow$

Training data

x	y
( ,	$\begin{bmatrix}1\\0\\0\end{bmatrix}$)
( ,	$\begin{bmatrix}0\\1\\0\end{bmatrix}$)
( ,	$\begin{bmatrix}0\\0\\1\end{bmatrix}$)
( ,	$\begin{bmatrix}0\\0\\1\end{bmatrix}$)

$\vdots$

in general, for K classes:



Negative log-likelihood $K-$ classes loss (aka, cross-entropy)

g : softmax output

g_k : probability or confidence of belonging in class k

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

y :one-hot encoding label

y_k : kth entry in y, either 0 or 1

- Generalizes negative log likelihood loss $\mathcal{L}_{\text{nll}}(g, y) = -\left[y \log g + (1-y) \log\left(1-g\right)\right]$
- Despite the $K-$term sum, only the term corresponding to its true class label contributes, since all other $y_k = 0$

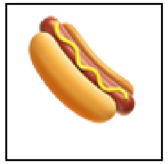
current prediction $g =$ softmax$(\cdot)$

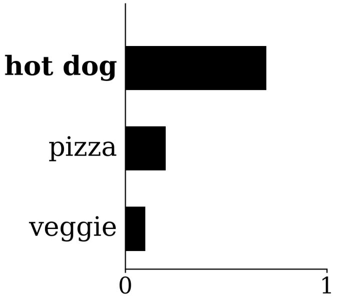
$\xrightarrow{-\log}$

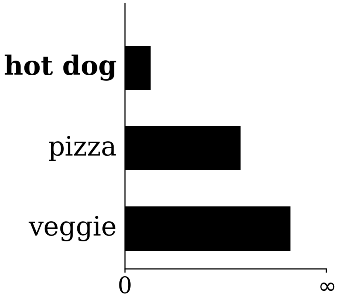
$-\log(g)$

y (true label)

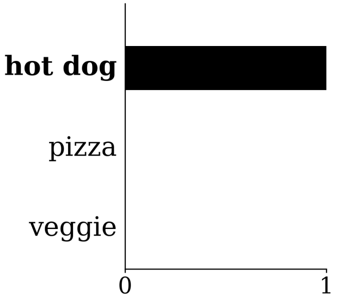
$$\mathcal{L}_{\text{nllm}} = -\sum_k y_k \cdot \log(g_k)$$

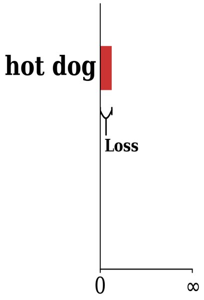






$\odot$





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

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

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

Loss ≈ 0.36

To reduce the loss, *g*🌭 needs to go up — this signal flows smoothly back to θ through $-\log$ and softmax, so we can optimize via gradient descent.

Summary

- Classification predicts a label from a discrete set; a linear binary classifier separates the feature space with a hyperplane defined by θ, θ_0.
- The 0-1 loss is natural for classification but NP-hard to optimize.
- The sigmoid $\sigma(z)$ gives a smooth, probabilistic step function; paired with the NLL loss, we can train via (S)GD.
- Regularization remains important for logistic classifiers.
- Multi-class classification generalizes via one-hot encoding and softmax.