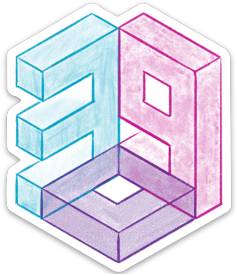


https://introml.mit.edu/

# 6.390 Intro to Machine Learning

## Lecture 4: Linear Classification

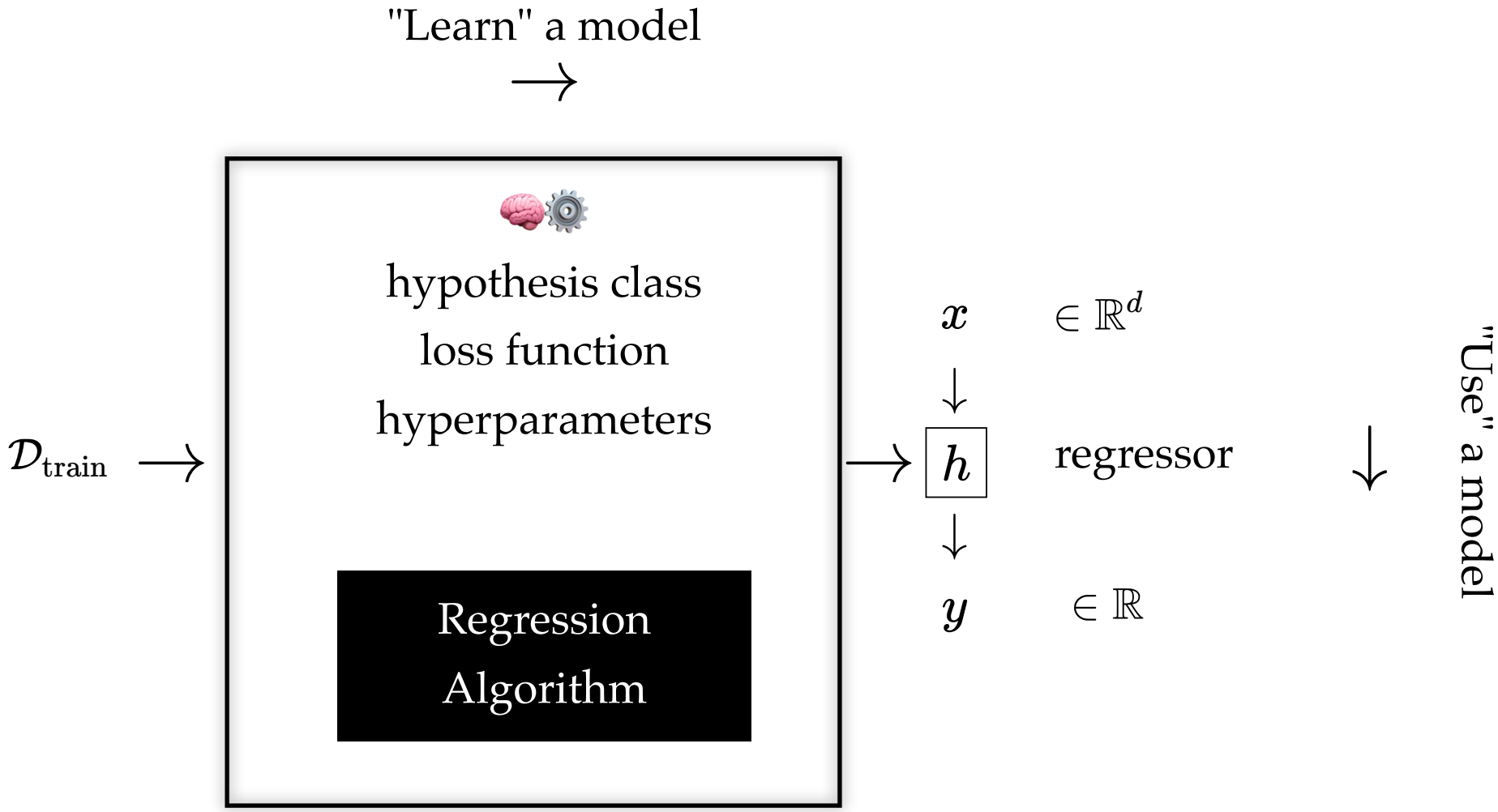
Shen Shen

Sept 25, 2025

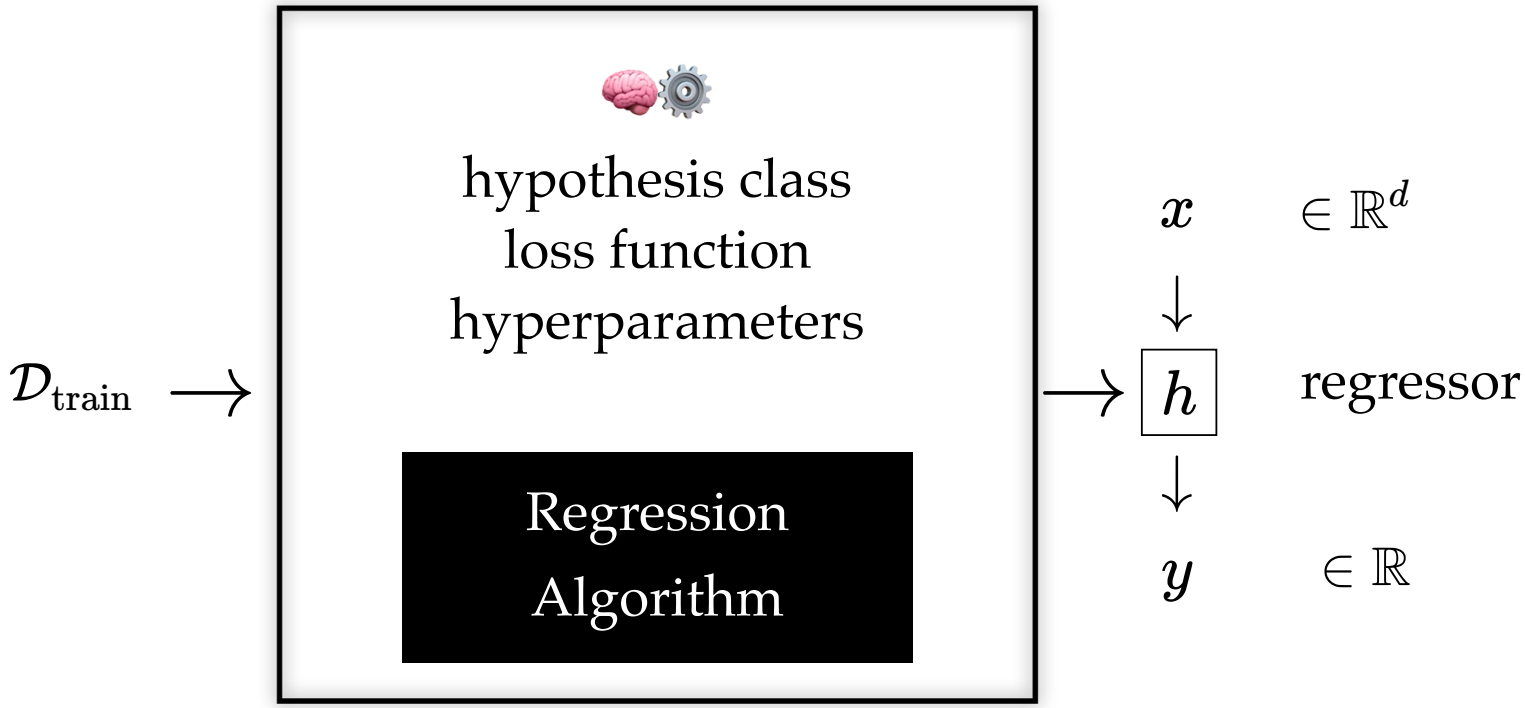
11am, Room 10-250

Interactive Slides and Lecture Recording

Recap:


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

Recap:




"Learn" a model $\longrightarrow$

train, optimize, tune, adapt ...

adjusting/updating/finding $\theta$

gradient based

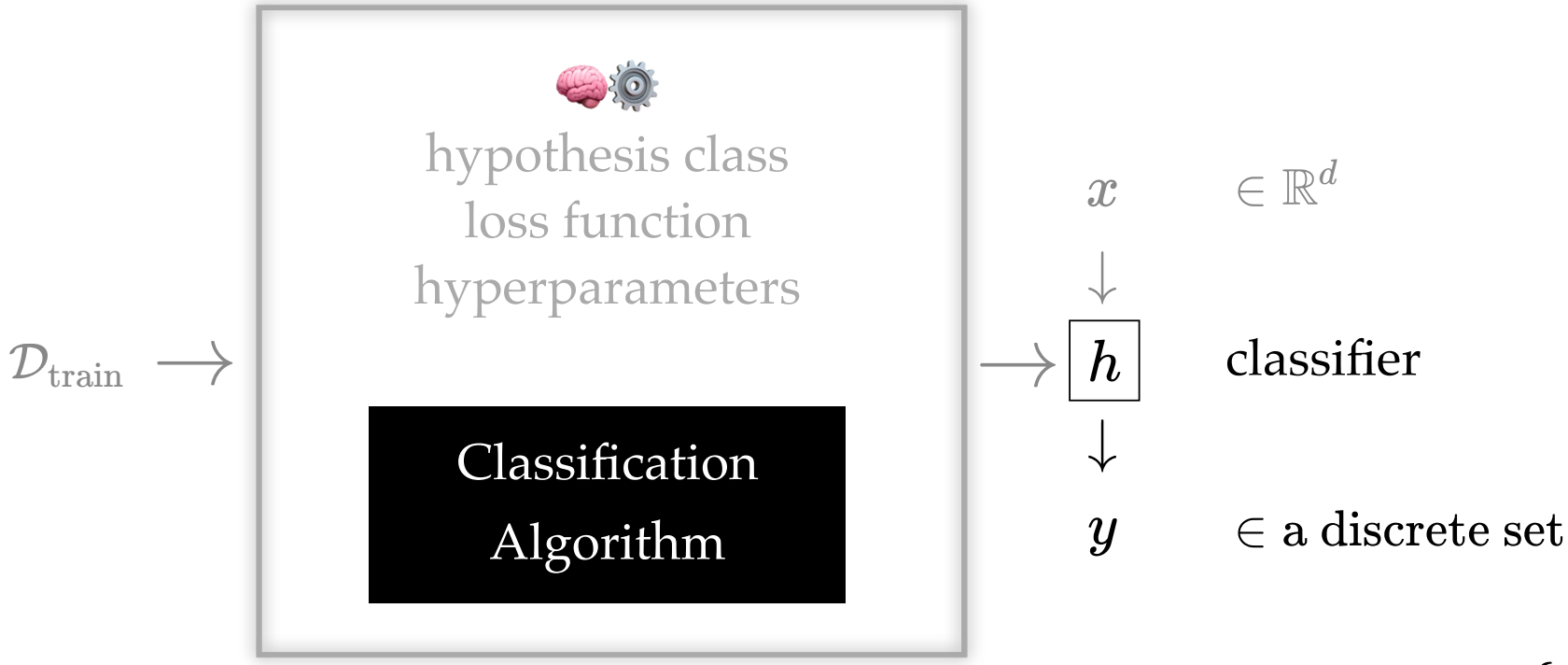
$\downarrow$ "Use" a model

predict, test, evaluate, infer ...

plug in the $\theta$ found

no gradients involved

Today:




$\{+1, 0\}$

$\{😍, 🥺\}$

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

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

*Training data*

features | label

 , "Fish"

 , "Grizzly"

 , "Chameleon"

, ⋮

hypothesis class
loss function
hyperparameters

**Classification Algorithm**

$x$

$\downarrow$

$h$ classifier

$\downarrow$

$y$ $\in$ a discrete set

"Fish"

$\in$

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

images adapted from Phillip Isola

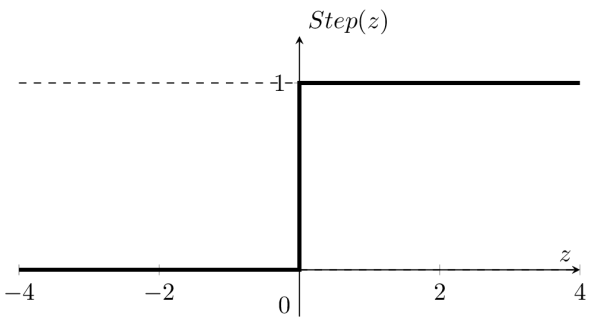
# Outline

1. Linear (binary) classifiers
    - to use: **separator, normal vector**
    - to learn: difficult! won't do
2. Linear logistic (binary) classifiers
    - to use: **sigmoid**
    - to learn: **negative log-likelihood loss**
3. Linear multi-class classifiers
    - to use: **softmax**
    - to learn: **one-hot encoding, cross-entropy loss**

# Outline

1. Linear (binary) classifiers
    - to use: **separator, normal vector**
    - to learn: difficult! won't do
2. Linear logistic (binary) classifiers
    - to use: sigmoid
    - to learn: negative log-likelihood loss
3. Linear multi-class classifiers
    - to use: softmax
    - to learn: one-hot encoding, cross-entropy loss

| | 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

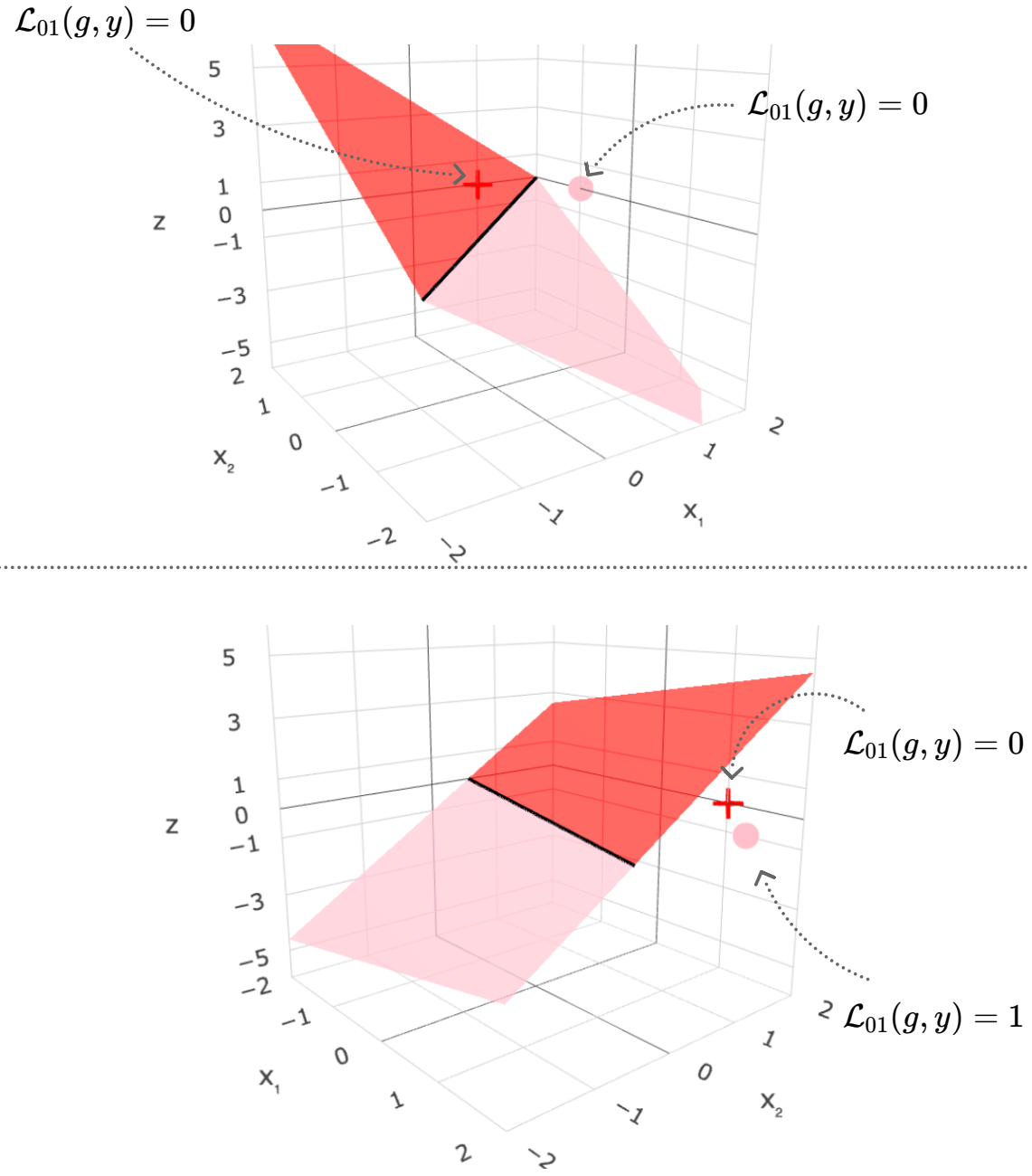
1. Linear (binary) classifiers
    - to use: separator, normal vector
    - to learn: difficult! won't do
2. Linear logistic (binary) classifiers
    - to use: sigmoid
    - to learn: negative log-likelihood loss
3. Multi-class classifiers
    - to use: softmax
    - to learn: one-hot encoding, cross-entropy loss

- To *learn* a model, need a loss function.
- One natural loss choice:

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

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

- Very intuitive, and easy to evaluate 😍



https://shenshen.mit.edu/demos/classification/01loss.html

Very hard to optimize (NP-hard) 🥺

- "Flat" almost everywhere (zero gradient)
- "Jumps" elsewhere (no gradient)

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

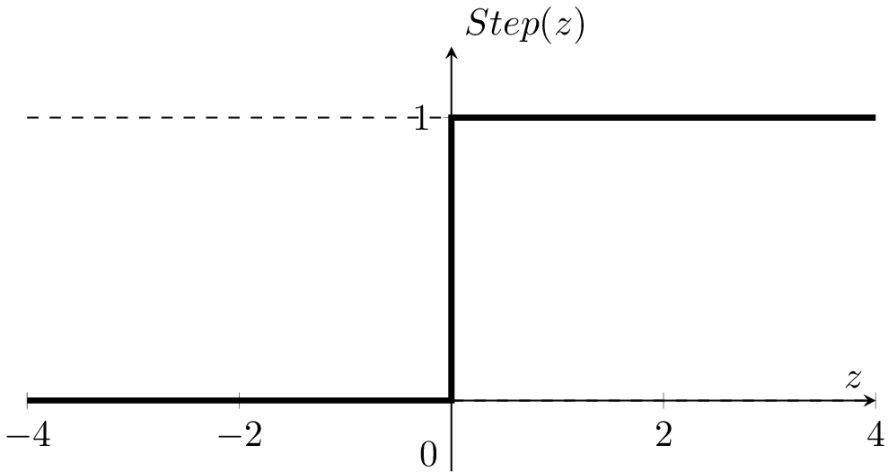
| | linear regressor | linear binary classifier |
|---|---|---|
| features | $x \in \mathbb{R}^d$ | |
| | $y \in \mathbb{R}$ | $y \in \{0, 1\}$ |
| parameters | $\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$ | |
| linear combo | $\theta^T x + \theta_0 \quad = z$ | |
| predict | $g = z$ | $g = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$ |
| loss | $(g - y)^2$ | $\begin{cases} 0 & \text{if } g = y \\ 1 & \text{otherwise} \end{cases}$ |
| optimize via | closed-form or gradient descent | both discrete |

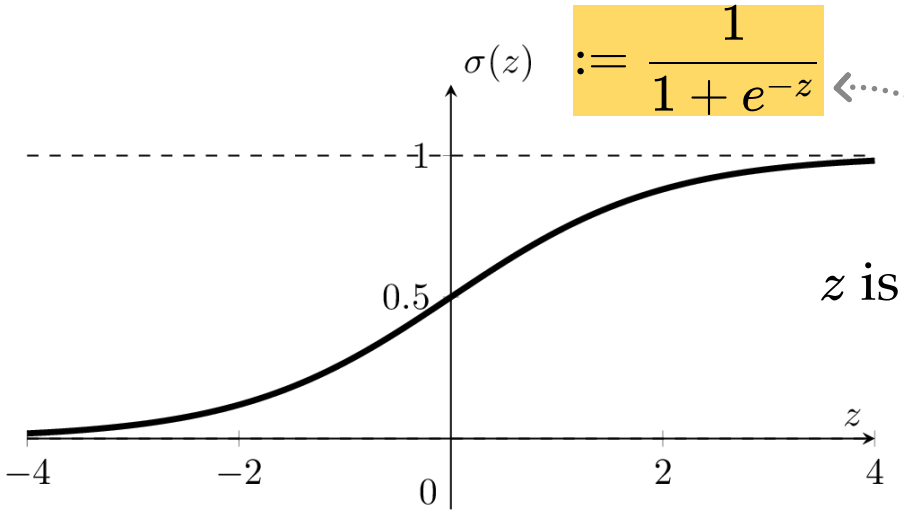
NP-hard to learn

# Outline

1. Linear (binary) classifiers
    - to use: separator, normal vector
    - to learn: difficult! won't do
2. Linear logistic (binary) classifiers
    - to use: **sigmoid**
    - to learn: **negative log-likelihood loss**
3. Linear multi-class classifiers
    - to use: softmax
    - to learn: one-hot encoding, cross-entropy loss

*Sigmoid* : a smooth step function



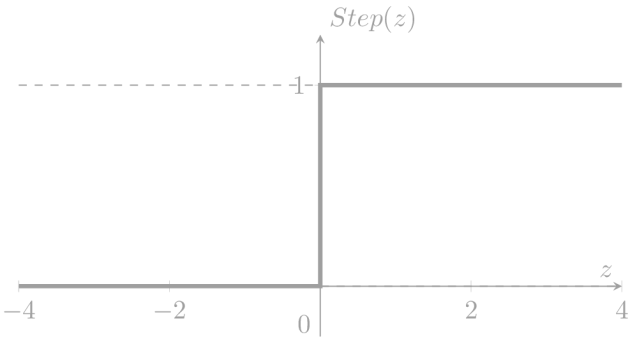




$z$ is called the *logit*

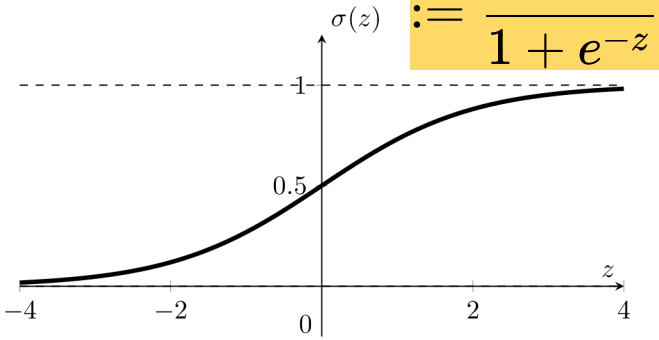
Predict $\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$

Predict $\begin{cases} 1 & \text{if } \sigma(z) > 0.5 \\ 0 & \text{otherwise} \end{cases}$

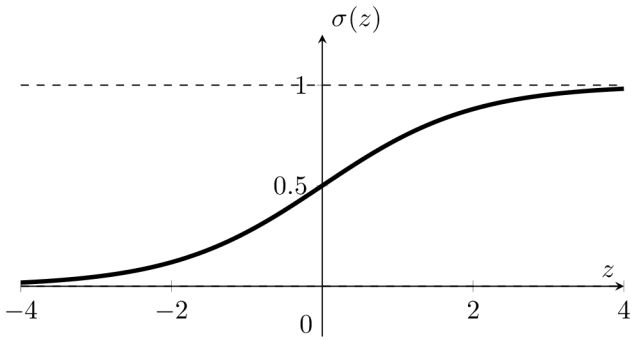
| | linear binary classifier | linear *logistic* binary classifier |
|---|---|---|
| features | $x \in \mathbb{R}^d$ | |
| parameters | $\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$ | |
| linear combo | $\theta^T x + \theta_0 = z$ | |
| predict | $\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise}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




$$:= \frac{1}{1+e^{-z}}$$



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




- The logit $z$ is a linear combo of $x$ via the parametes.
- Sigmoid squashes the logit $z$ into a number in $(0, 1)$.
- Predict positive class label 1

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

$$= \frac{1}{1+e^{-(\theta^\top x+\theta_0)}} > 0.5$$

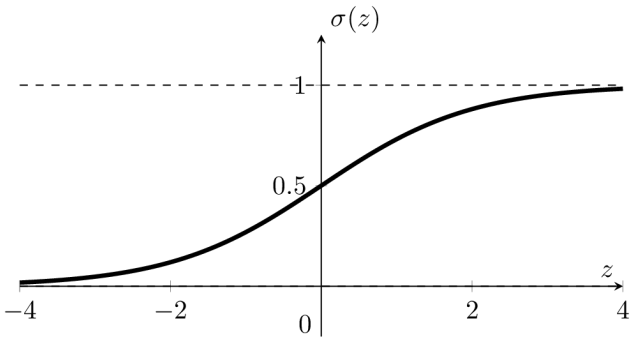
$\sigma(\cdot)$ : the model's *confidence* or *estimated likelihood* that feature $x$ belongs to the positive class.

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

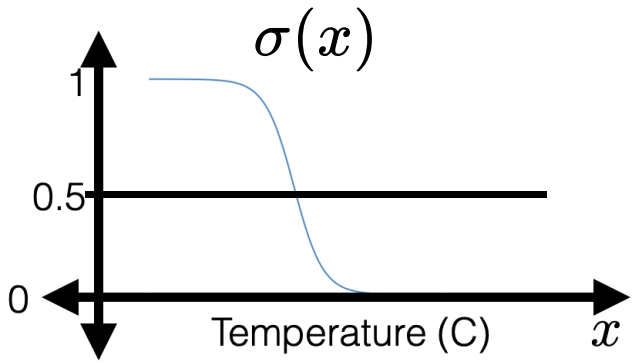
https://shenshen.mit.edu/demos/classification/step_sigmoid.html

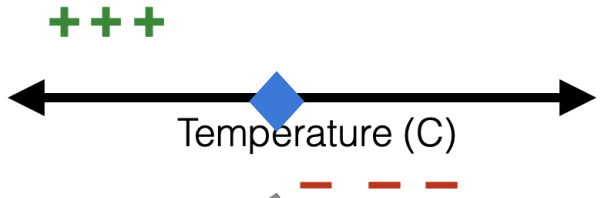
- $\theta$, $\theta_0$ can flip, squeeze, expand, or shift the $\sigma(x)$ graph *horizontally*
- $\sigma(\cdot)$ monotonic, very elegant gradient (see hw/lab)

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

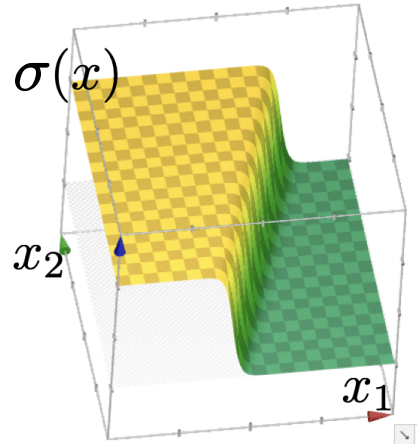


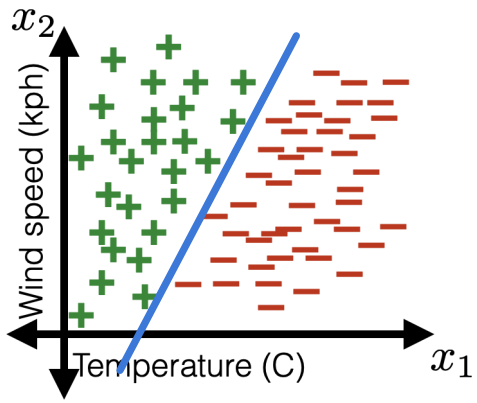

1d feature




2d feature




$z = \theta^T x + \theta_0 = 0$

linear separator

images credit: Tamara Broderick

# Outline

1. Linear (binary) classifiers
    - to use: separator, normal vector
    - to learn: difficult! won't do
2. Linear **logistic** (binary) classifiers
    - to use: **sigmoid**
    - to learn: **negative log-likelihood loss**
3. Linear multi-class classifiers
    - to use: softmax
    - to learn: one-hot encoding, cross-entropy loss

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

- if true label $y = 1$, we'd like the guess $g$ to just ***be*** 1... a difficult ask

now:

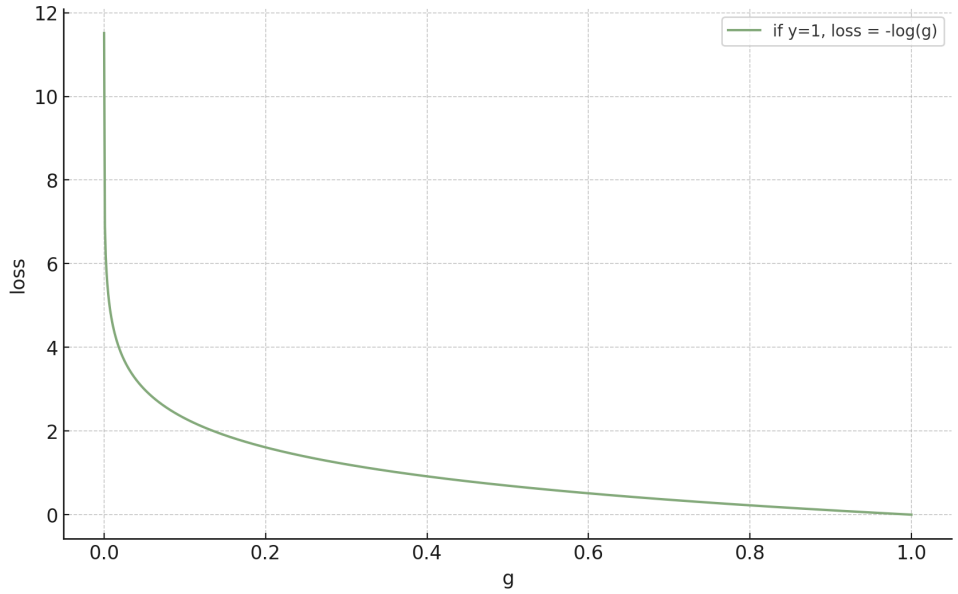
- if true label $y = 1$, we'd like the guess $g$ to ***gradually approach*** 1

👇

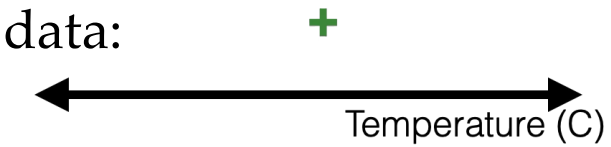
$$\mathcal{L}_{\text{nll}}(g, y) = \mathcal{L}_{\text{nll}}(g, 1) := -\log(g)$$

negative log likelihood

$g = \sigma(\cdot)$ : the model's *confidence* or *estimated likelihood* that feature $x$ belongs to the positive class.



training data:




true label $y$ is 1

👇

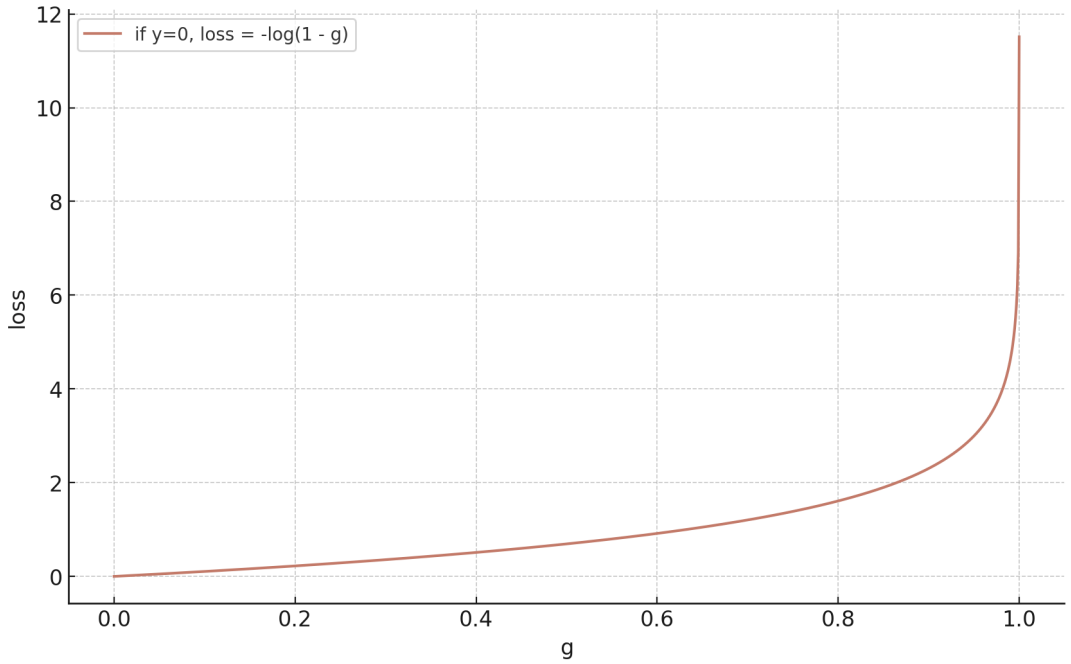
$$\mathcal{L}_{\text{nll}}(g, y) = \mathcal{L}_{\text{nll}}(g, 1) := -\log(g)$$

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

- if true label $y = 0$, we'd like the guess $g$ to ***gradually approach*** $0$

👇

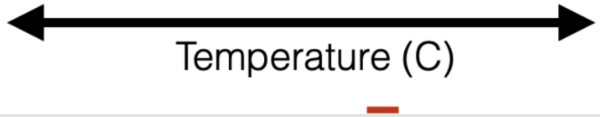
$$\mathcal{L}_{\text{nll}}(g, y) = \mathcal{L}_{\text{nll}}(g, 0) := -\log(1 - g)$$




$g = \sigma(\cdot)$ : the model's *confidence* or *estimated likelihood* that feature $x$ belongs to the **positive** class.

$1 - g = 1 - \sigma(\cdot)$ : the model's *confidence* or *estimated likelihood* that feature $x$ belongs to the **negative** class.

training data:




true label $y$ is 0

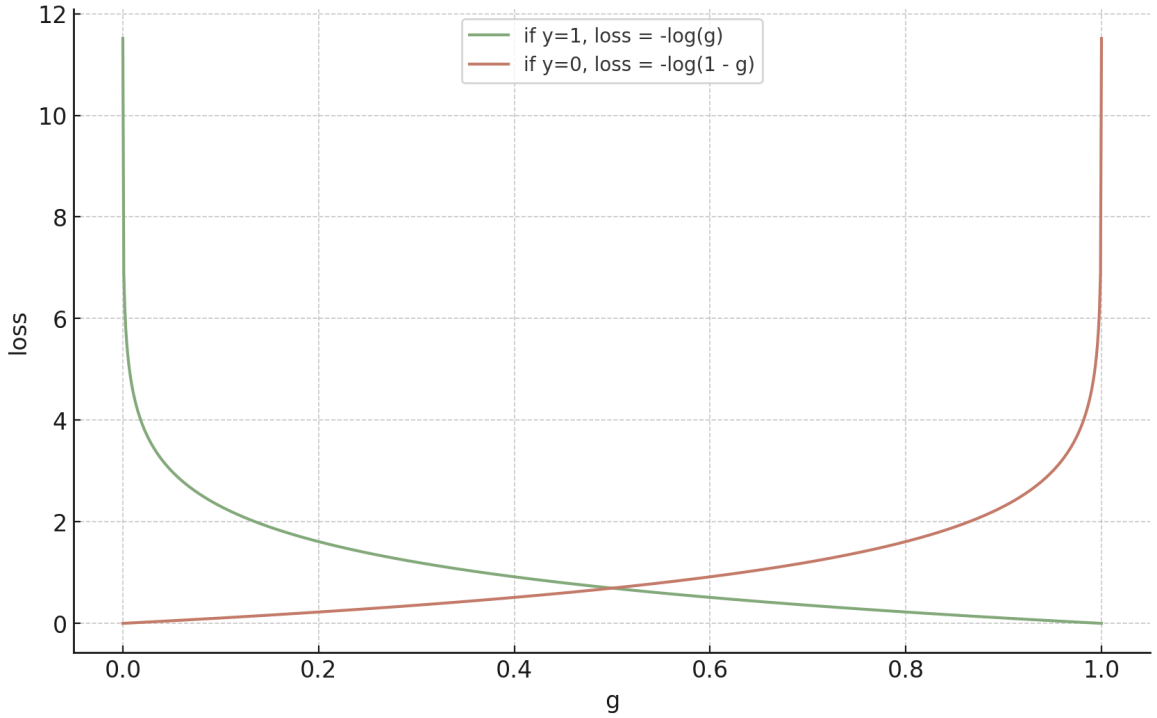
👇

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

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

Combining both cases, since the actual label $\boldsymbol{y \in \{+1, 0\}}$

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



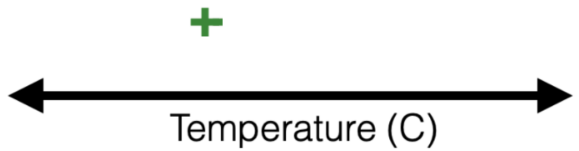
When $y = 1$

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

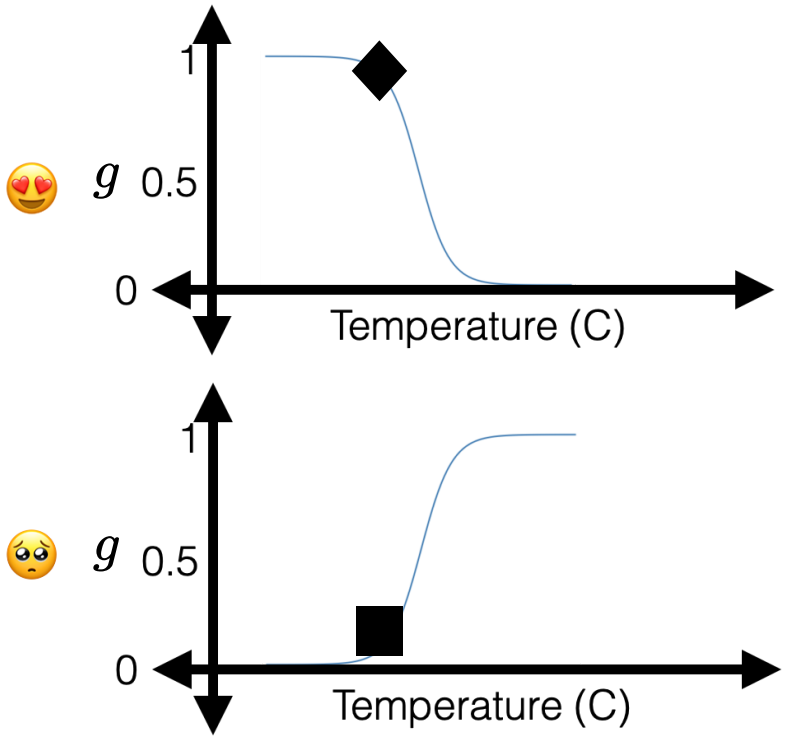
$$= -\left[\cancel{y}\log g + \cancel{(1-y)\log(1-g)}\right]$$

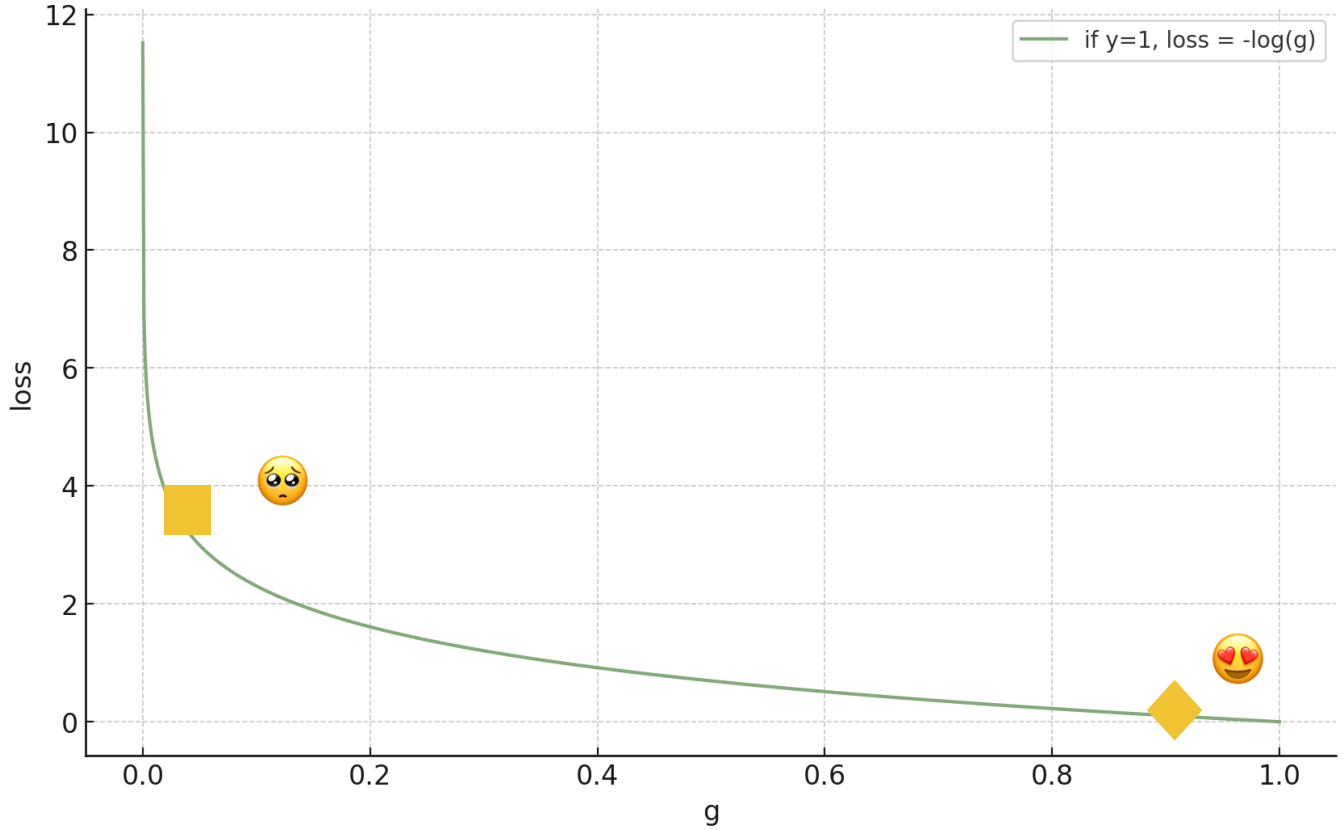
$$= -\log g$$

training data:




$$g(x) = \sigma(\theta x + \theta_0)$$



When $y = 0$

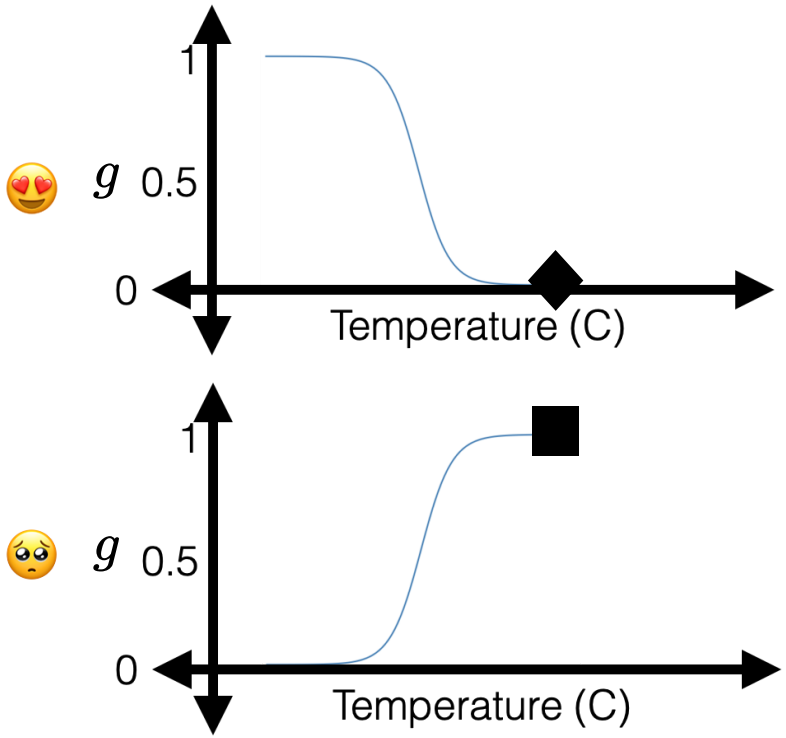
$$\mathcal{L}_{\text{nll}}(g, y)$$

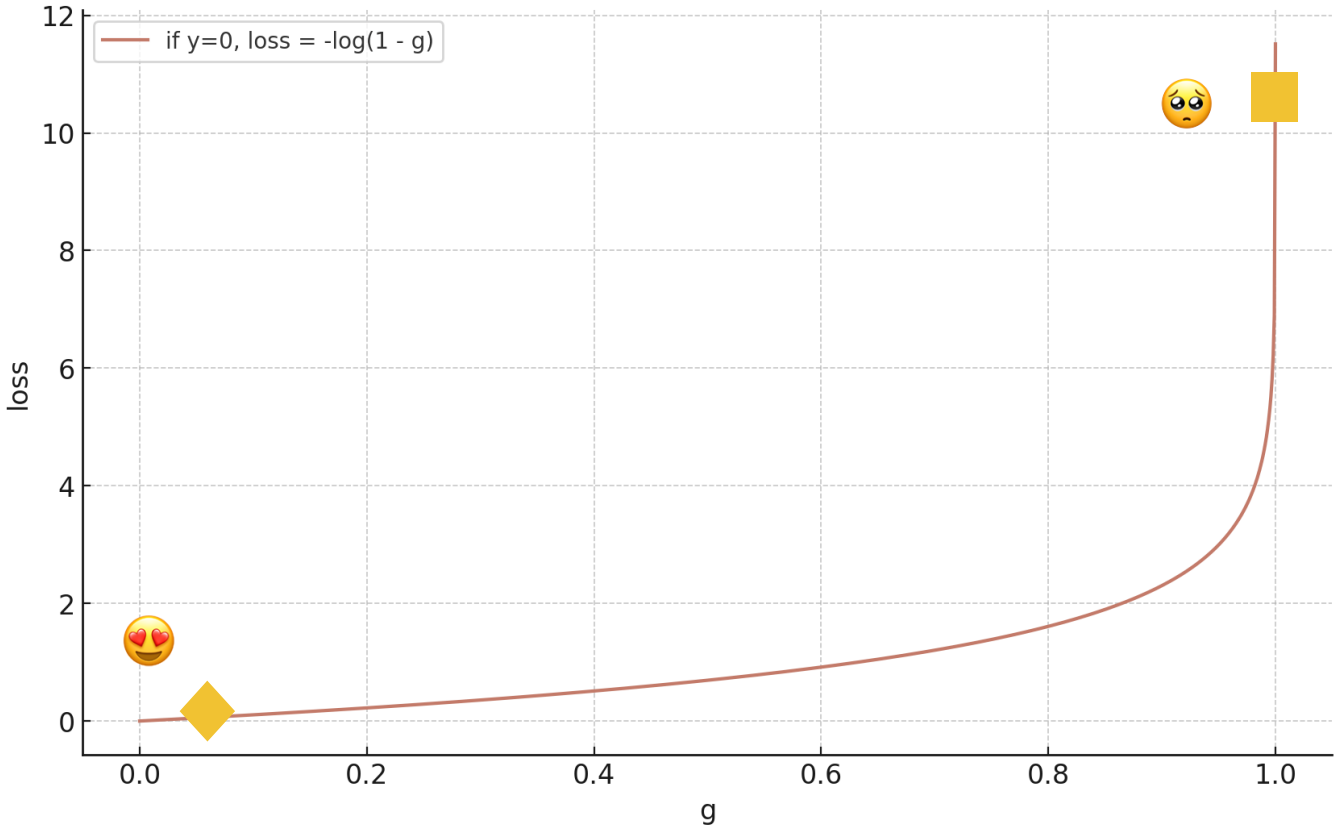
$$= -\left[\cancel{y \log g} + \cancel{(1-y)} \log(1-g)\right]$$

$$= -\left[\log(1-g)\right]$$

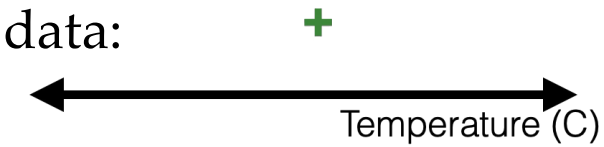
training data:


$$g(x) = \sigma(\theta x + \theta_0)$$



training data:




true label $y$ is 1

👇

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

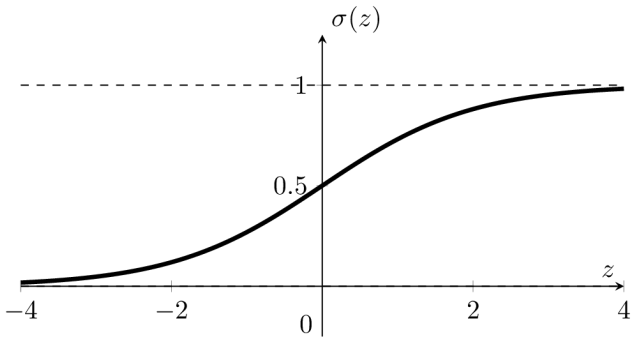
https://shenshen.mit.edu/demos/nlloverfit.html

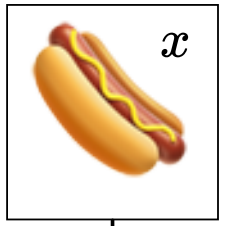
| | linear regressor | linear binary classifier | linear *logistic* binary classifier |
|---|---|---|---|
| features | $x \in \mathbb{R}^d$ | | |
| label | $y \in \mathbb{R}$ | $y \in \{0, 1\}$ | |
| parameters | $\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$ | | |
| linear combo | $\theta^T x + \theta_0 \quad = z$ | | |
| predict | $g = z$ | $\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$ | $\begin{cases} 1 & \text{if } g = \sigma(z) > 0.5 \\ 0 & \text{otherwise} \end{cases}$ |
| loss | $(g - y)^2$ | $\begin{cases} 0 & \text{if } g = y \\ 1 & \text{otherwise} \end{cases}$ | $-\left[y \log g + (1 - y) \log (1 - g)\right]$ |
| optimize via | closed-form or gradient descent | NP-hard to learn | • gradient descent<br>• regularization still important |

# Outline

1. Linear (binary) classifiers
    - to use: separator, normal vector
    - to learn: difficult! won't do
2. Linear logistic (binary) classifiers
    - to use: sigmoid
    - to learn: negative log-likelihood loss
3. Multi-class classifiers
    - to use: **softmax**
    - to learn: **one-hot encoding, cross-entropy loss**

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



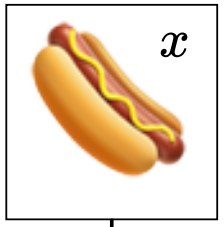




$z \in \mathbb{R}$ scalar logit (raw hotdog-ness)

$\sigma(z) \in \mathbb{R}$ scalar likelihood (normalized probability of hotdog-ness)

- $\sigma(\cdot)$ : the model's *confidence* or *estimated likelihood* that feature $x$ belongs to the **hotdog** class;
- $1 - \sigma(\cdot)$ : the **not-hotdog** class.

to predict hotdog or not,
a scalar logit suffices

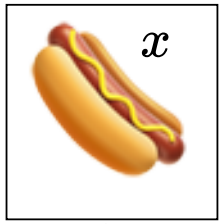



$z \in \mathbb{R}$

$\sigma(z) \in \mathbb{R}$

to predict among $K$ categories

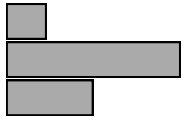
say $K = 3$ categories: {hot-dog, pizza, salad}



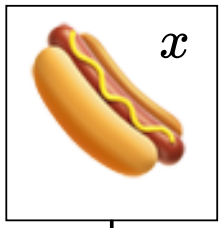

$z \in \mathbb{R}^3$

$K$ logits — raw likelihood of each category

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

$K$-class likelihood  distribution over the categories

two classes

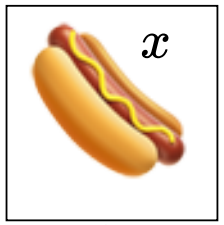



$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$

$z \in \mathbb{R}$

$\sigma(z) \in \mathbb{R}$

$K$ classes



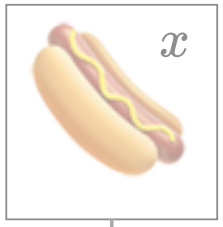

$\theta \in \mathbb{R}^{d \times K}, \theta_0 \in \mathbb{R}^K$

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

raw likelihood of each category

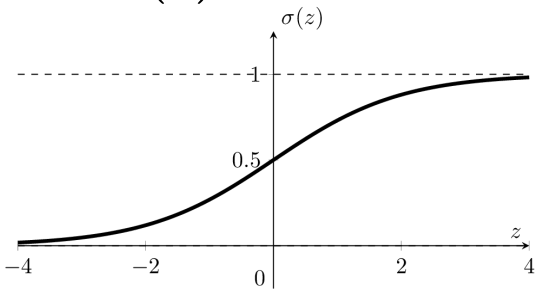
$\text{softmax}(z) \in \mathbb{R}^K$

two classes

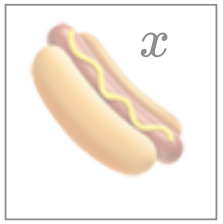



$z \in \mathbb{R}$

$\sigma(z) \in \mathbb{R}$

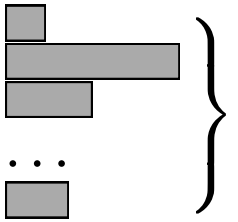



$K$ classes




$z \in \mathbb{R}^K$ raw likelihood of each category

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



distribution over $K$ categories

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

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

each output entry is between 0 and 1, and their sum is 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 in the input

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

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

predict the category with the highest softmax score

equivalently, predicting the category with the largest raw logit.

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

# Outline

1. Linear (binary) classifiers
    - to use: separator, normal vector
    - to learn: difficult! won't do
2. Linear logistic (binary) classifiers
    - to use: sigmoid
    - to learn: negative log-likelihood loss
3. 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.

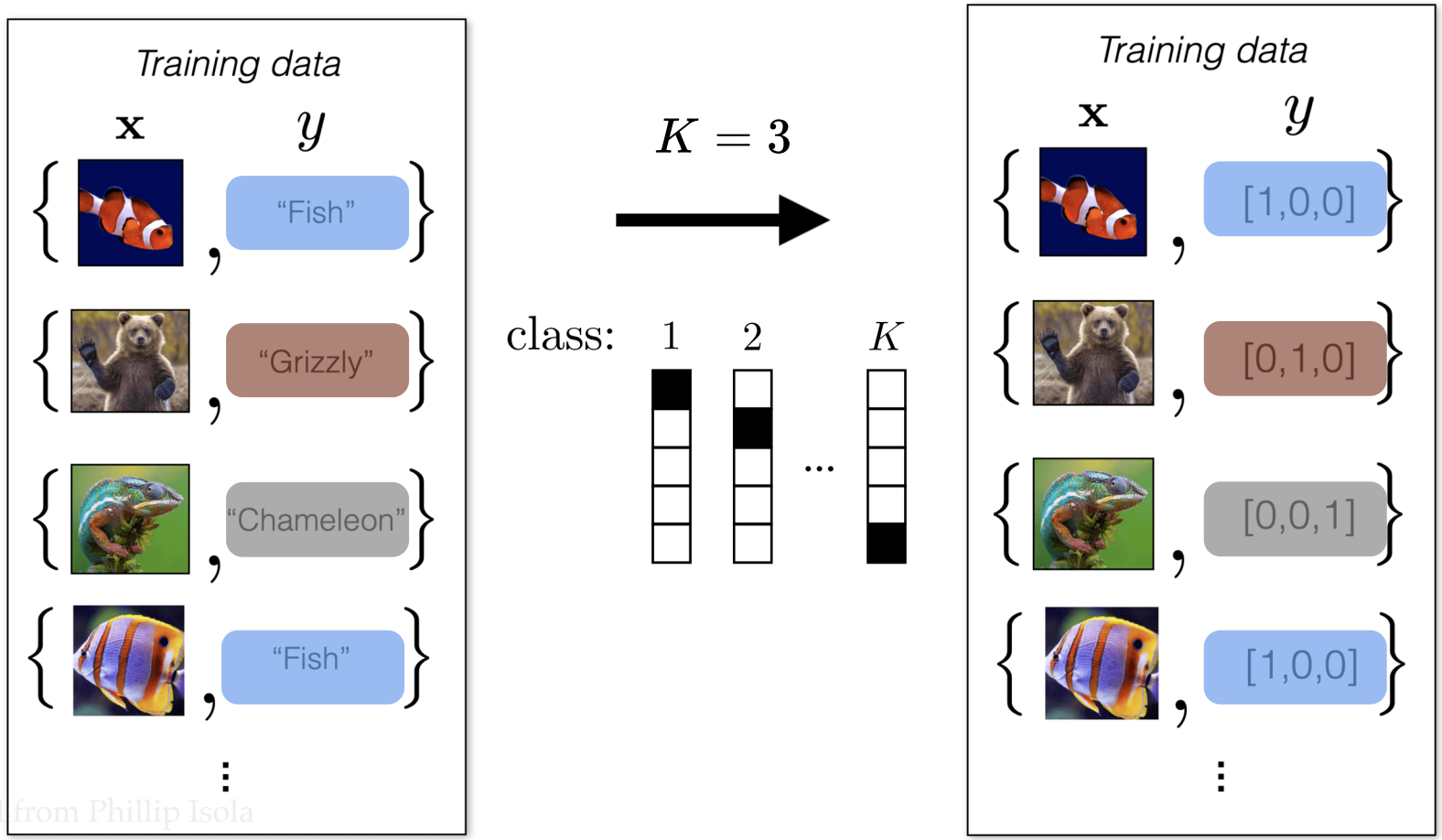



image adapted from Phillip Isola

column vectors, flipped
(for slide real estate)

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

$y_k$ : $k$th entry in $y$, either 0 or 1

- Generalizes negative log likelihood loss $\mathcal{L}_{\text{nll}}(g, y) = -\left[y \log g + (1-y) \log\left(1-g\right)\right]$
- Although this is written as a sum over $K$ terms, for a given training data point, only the term corresponding to its true class label contributes, since all other $y_k = 0$

current prediction

$g = \text{softmax}(\cdot)$

true label $y$

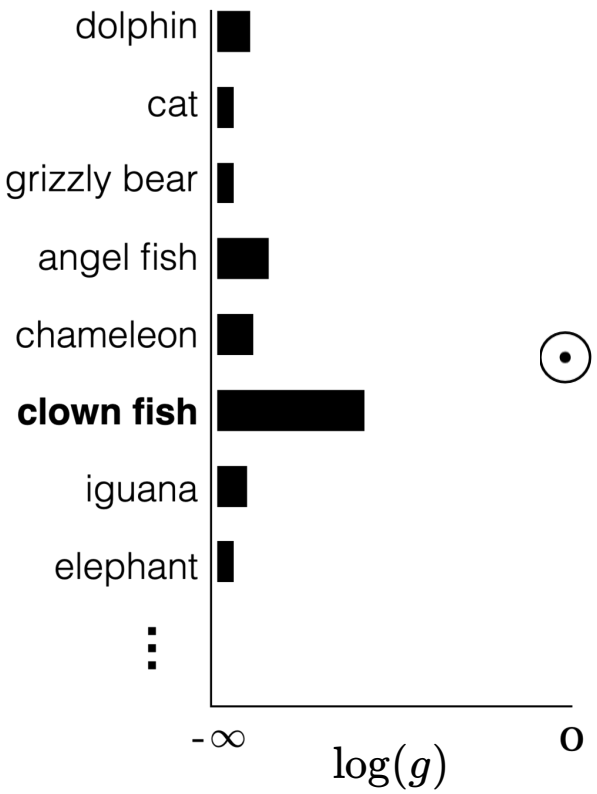
$[0, 0, 0, 0, 0, 1, 0, 0, \ldots]$

loss $\mathcal{L}_{\text{nllm}}(g, y)$

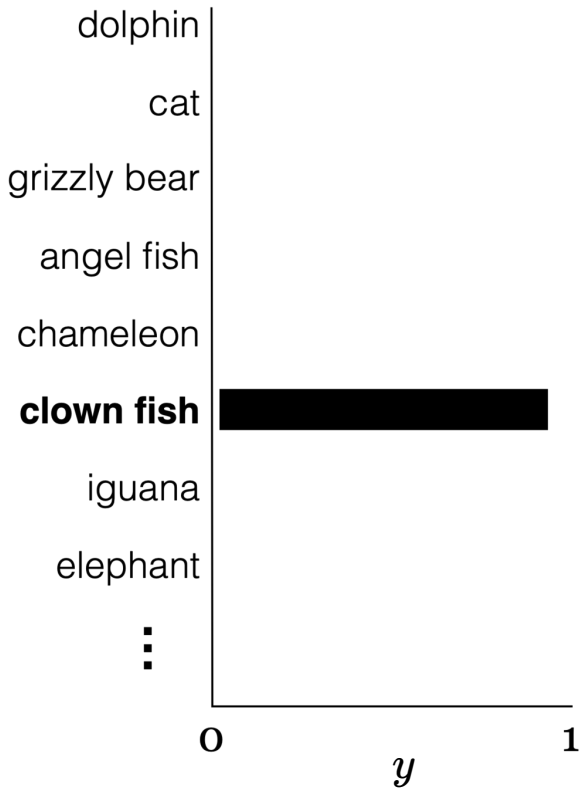
$= -\sum_{k=1}^{K} y_k \cdot \log(g_k)$

feature $x$

$h$




$\odot$



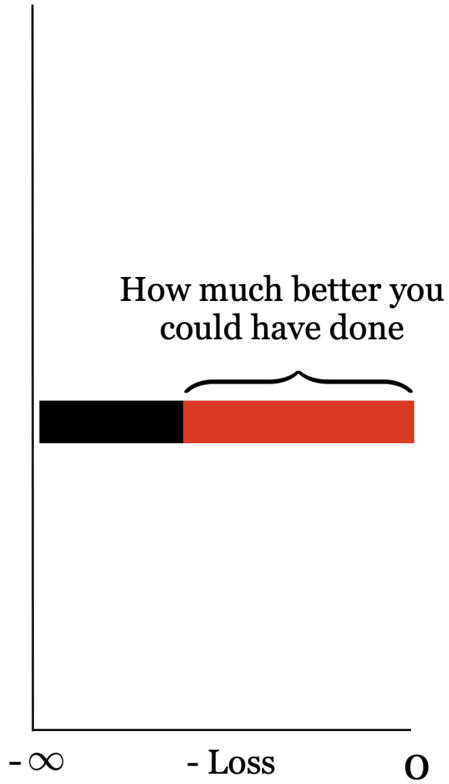




image adapted from Phillip Isola

current prediction

$g = \text{softmax}(\cdot)$

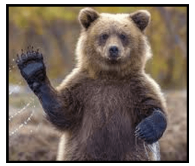
true label $y$

$[0, 0, 1, 0, 0, 0, 0, 0, \ldots]$

loss $\mathcal{L}_{\text{nllm}}(g, y)$

$= -\sum_{k=1}^{K} y_k \cdot \log(g_k)$

feature $x$



$h$


$\odot$

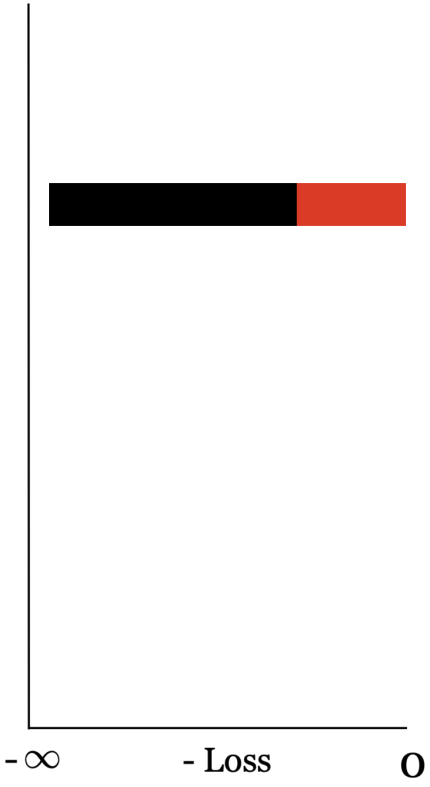
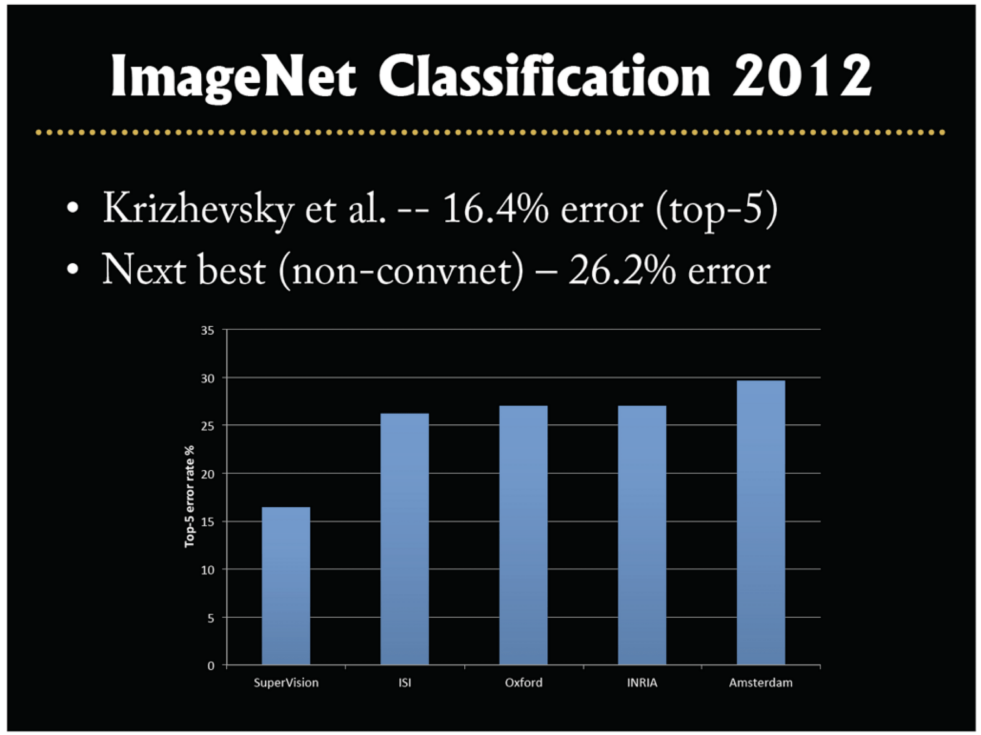
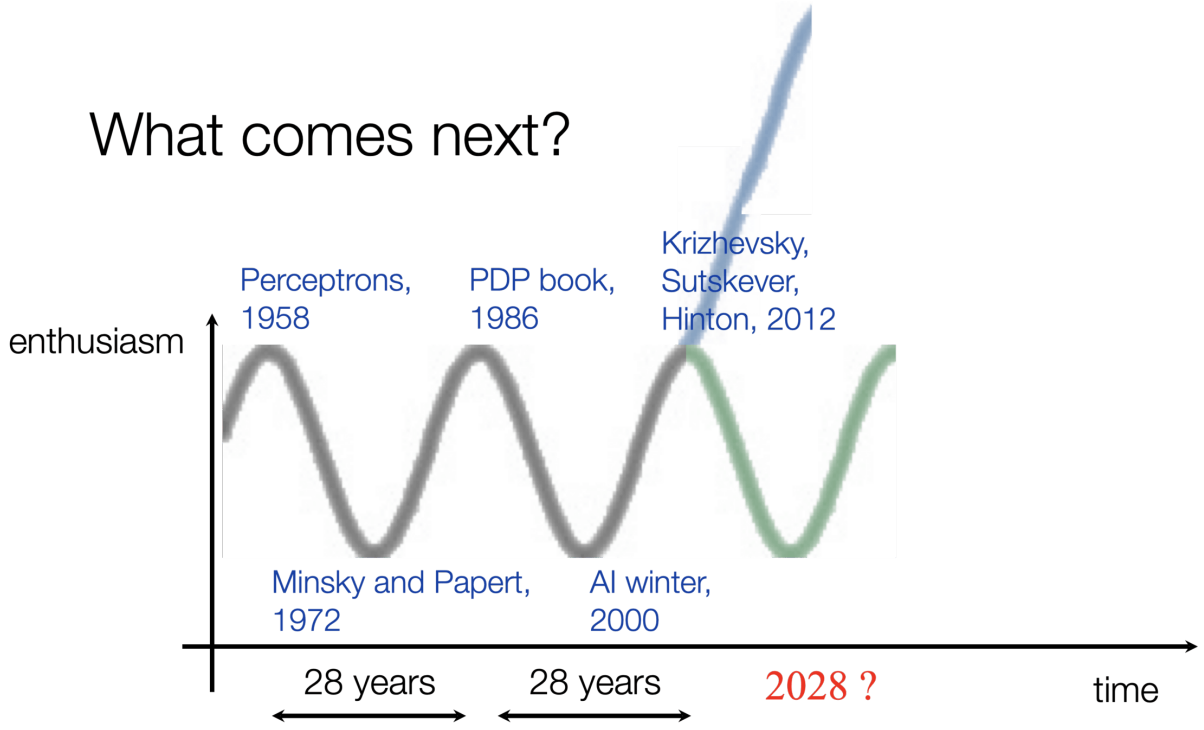




image adapted from Phillip Isola

# Classification

Image classification played a pivotal role in kicking off the current wave of AI enthusiasm.



# Summary

- Classification: a supervised learning problem, similar to regression, but where the output/label is in a discrete set.
- Binary classification: only two possible label values.
- Linear binary classification: think of $\theta$ and $\theta_0$ as defining a hyperplane that **cuts** the d-dimensional feature space into two half-spaces.
- 0-1 loss: a natural loss function for classification, BUT, hard to optimize.
- Sigmoid function: smoother and probabilistic step function, motivation and properties.
- NLL loss: smooth loss and has nice probabilistic motivations. Can optimize via (S)GD.
- Regularization is still important.
- The generalization to multi-class via (one-hot encoding, and softmax mechanism)
- Other ways to generalize to multi-class (see hw/lab)

https://docs.google.com/forms/d/e/1FAIpQLSfG1vnfaOvy8jugeVHrJWJQB-_15IWBq683-XI8zlAJf6YZNg/viewform?embedded=true

We'd love to hear
your thoughts.

# Thanks!