https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 4: Linear Classification

Shen Shen

Feb 21, 2025

(11am, Room 10-250)

Recap:

"Learn" a model $\longrightarrow$

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

Supervised Learning Algorithm

Hypothesis class

Hyperparameters

Objective (loss) function

Regularization

$\longrightarrow$ $x \in \mathbb{R}^d$ $\downarrow$ $\boxed{h}$ regressor $\downarrow$ $y \in \mathbb{R}$

$\downarrow$ "Use" a model

Recap:

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

Supervised Learning Algorithm

🧠⚙️

Hypothesis class
Hyperparameters
Objective (loss) function
Regularization

$\longrightarrow \boxed{h}$ regressor

$x \in \mathbb{R}^d$
$\downarrow$
h
$\downarrow$
$y \in \mathbb{R}$

"Learn" a model $\longrightarrow$

train, optimize, learn, tune,
adjusting/updating model parameters
gradient based

$\downarrow$ "Use" a model

predict, test, evaluate, infer
applying the learned model
no gradients involved

Today:

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

Supervised Learning Algorithm

🧠⚙️

Hypothesis class

Hyperparameters

Objective (loss) function

Regularization

$\rightarrow$ h classifier

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

$\downarrow$

h

$\downarrow$

y

$\in$ a discrete set

$\{+1, 0\}$

{😍, 🥺}

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

Today:

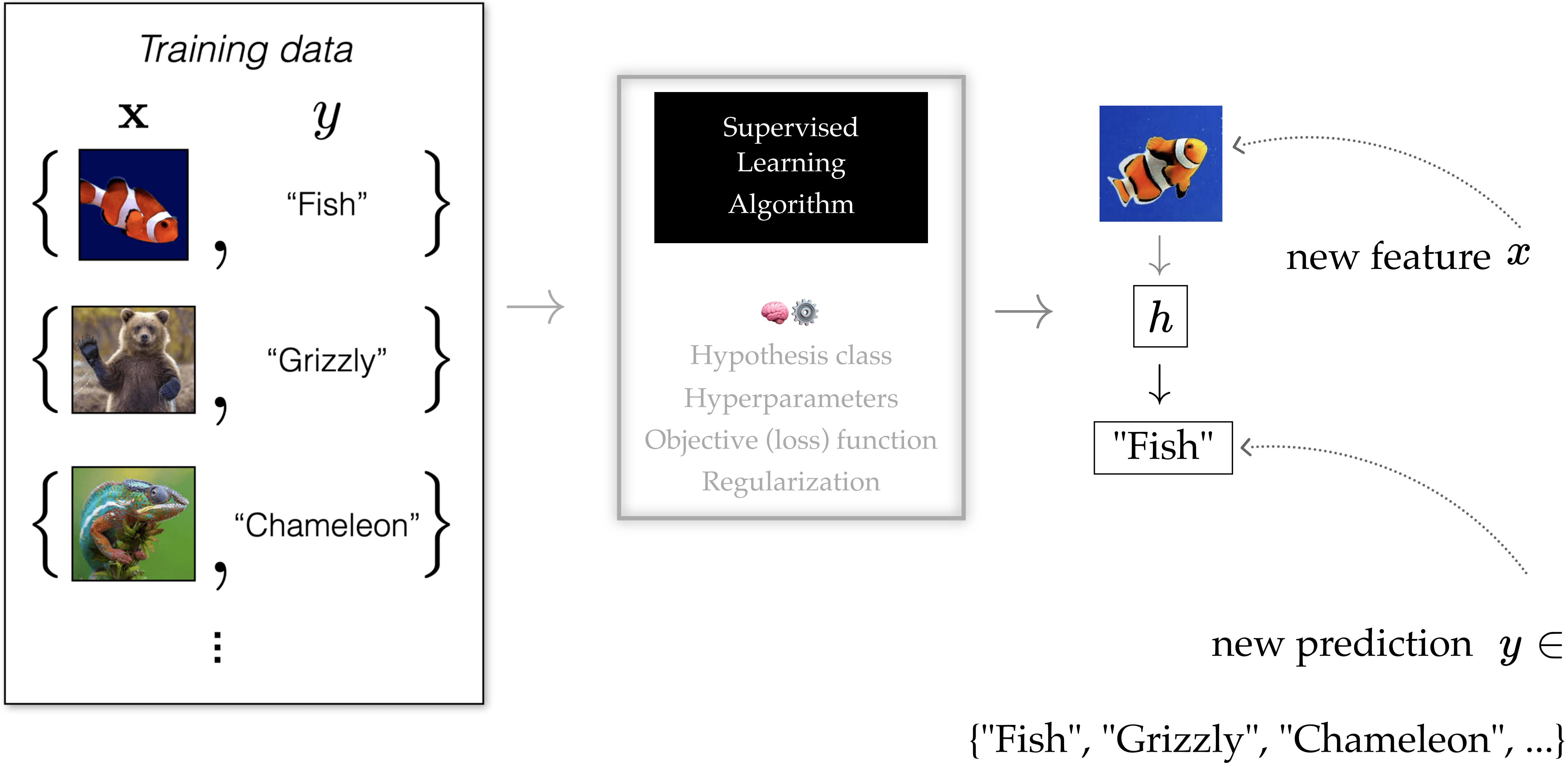


image adapted from Phillip Isola

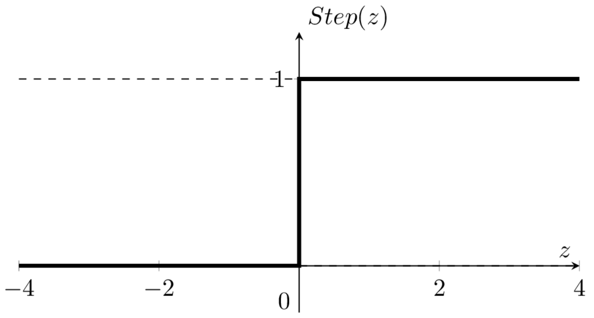
Outline

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

Outline

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

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

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

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

Outline

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

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

$$= \begin{cases} 0 & \text{if step}\left(\theta^\top x^{(i)} + \theta_0\right) = y^{(i)} \\ 1 & \text{otherwise} \end{cases}$$

- Very intuitive, and easy to evaluate 😍

https://shenshen.mit.edu/demos/01lossleftonly.html

- Very intuitive, and easy to evaluate 😍
- Very hard to optimize (NP-hard) 🥺
    - "Flat" almost everywhere (zero gradient)
    - "Jumps" elsewhere (no gradient)

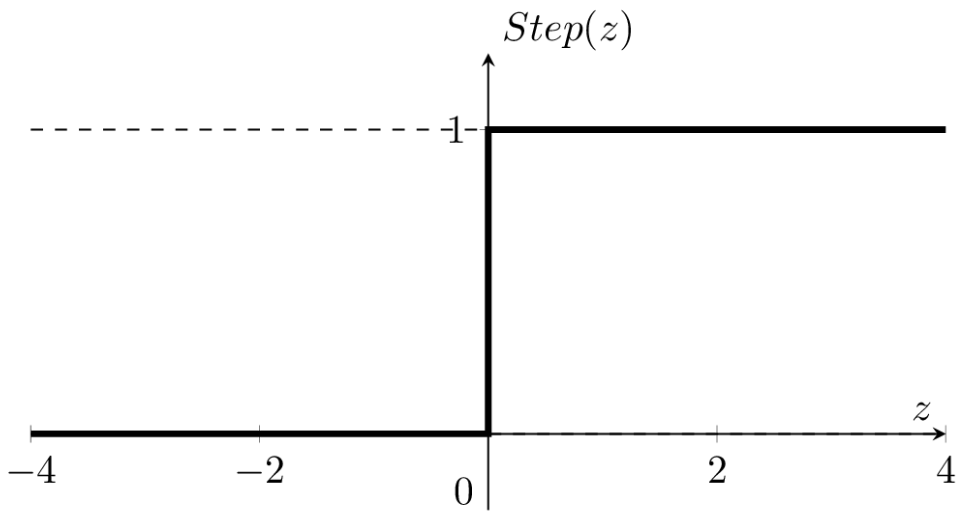
u/demos/01loss.html

Outline

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

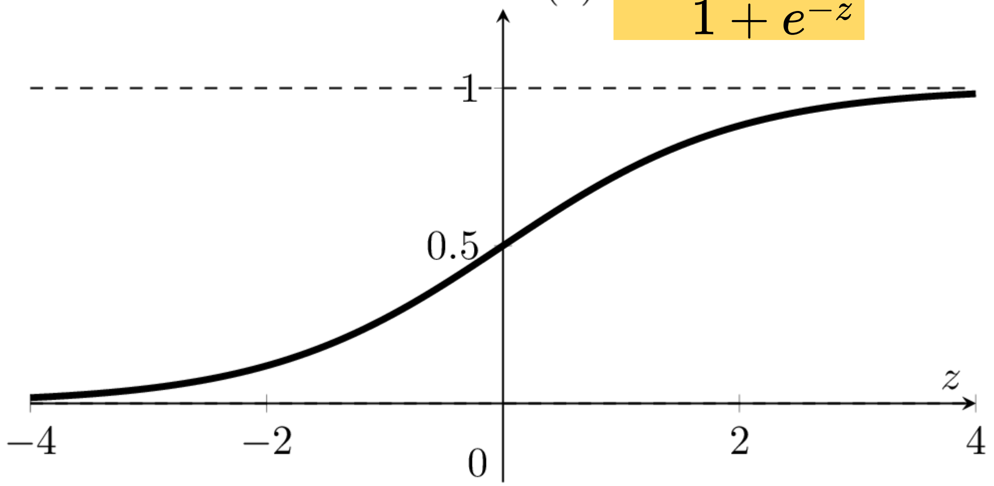
	linear binary classifier	linear *logistic* binary classifier
features	$x \in \mathbb{R}^d$	
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$	
linear combination	$\theta^T x + \theta_0 = z$	
predict	$\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1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

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



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

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



Sigmoid : a smooth step function

- Predict positive

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

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

- Sigmoid $\sigma(\cdot)$ outputs the *probability* or *confidence* that feature x has positive label.

- $\sigma(\cdot)$ between $(0, 1)$ *vertically*

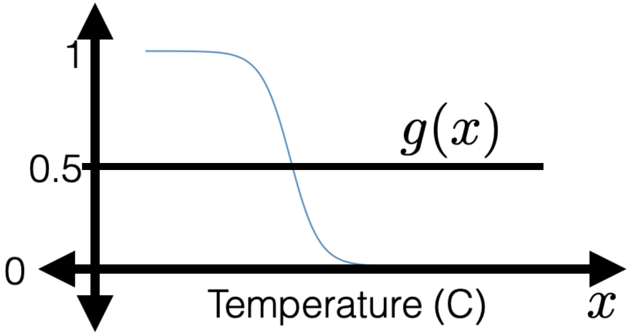
- θ, θ_0 can flip, squeeze, expand, or shift the $\sigma(\cdot)$ curve *horizontally*

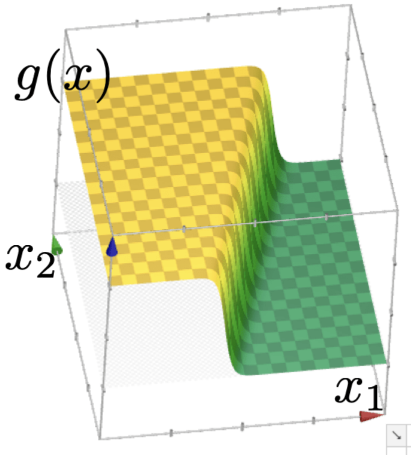
($\sigma(\cdot)$ monotonic, very elegant gradient (see hw/rec)

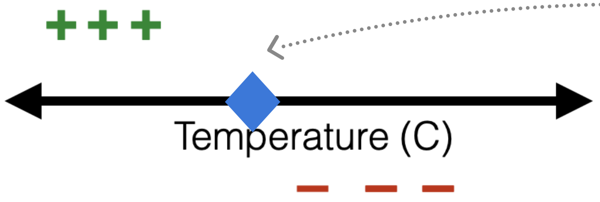
https://shenshen.mit.edu/demos/sigmoid.html e.g. to predict whether to bike to school using a *given* logistic classifier

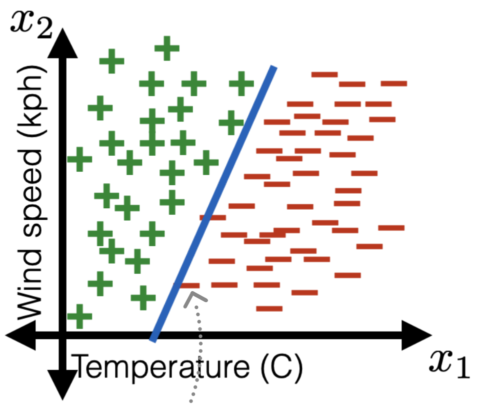
1 feature: $$g(x) = \sigma\left(\theta x + \theta_0\right) = \frac{1}{1 + \exp\left\{-\left(\theta x + \theta_0\right)\right\}}$$

2 features: $$g(x) = \sigma\left(\theta^\top x + \theta_0\right) = \frac{1}{1 + \exp\left\{-\left(\theta^\top x + \theta_0\right)\right\}}$$









linear logistic classifier still results in the linear separator $\theta^T x + \theta_0 = 0$

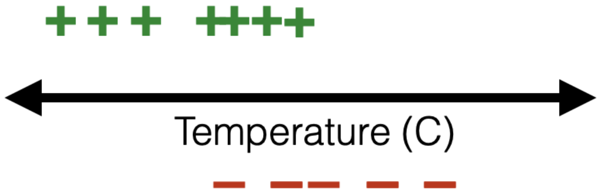
image credit: Tamara Broderick

Outline

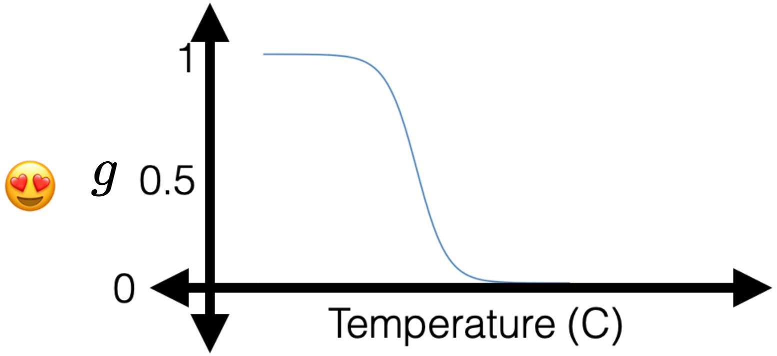
- Linear (binary) classifiers
  - to use: separator, normal vector
  - to learn: difficult! won't do
- Linear logistic (binary) classifiers
  - to use: sigmoid
  - to learn: negative log-likelihood loss
- Multi-class classifiers
  - to use: softmax
  - to learn: one-hot encoding, cross-entropy loss

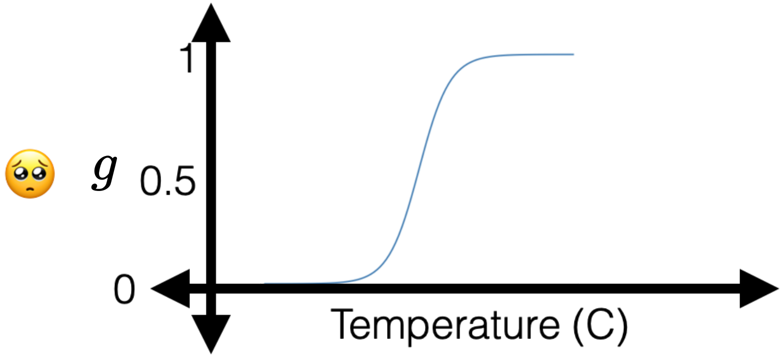
Recall, the labels $\boldsymbol{y} \in \{+1, 0\}$

training data:



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

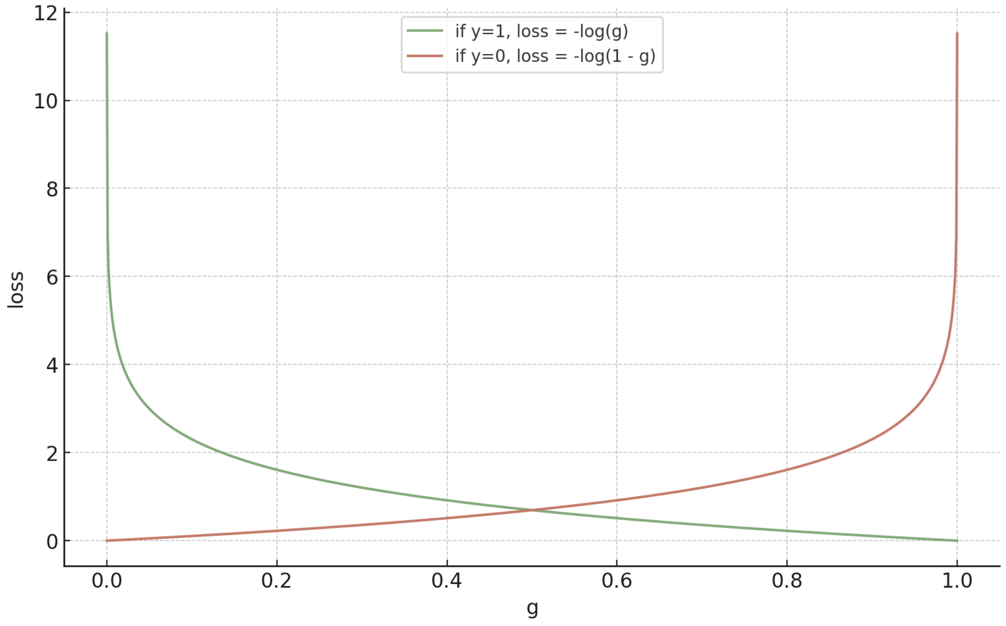




$$\mathcal{L}_{\text{nll}}\,(\text{ guess, actual })$$

$$= -[\text{ actual } \cdot \log(\text{ guess }) + (1 - \text{ actual }) \cdot \log(1 - \text{ guess })]$$

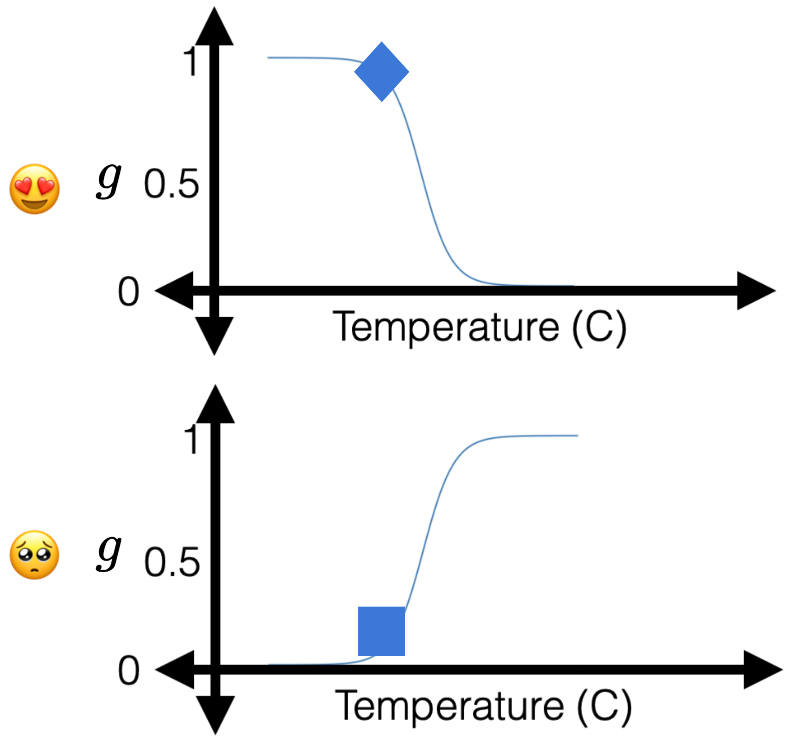
$$= -\left[y \log g + (1-y) \log\left(1-g\right)\right]$$



If $y = 1$

training data:

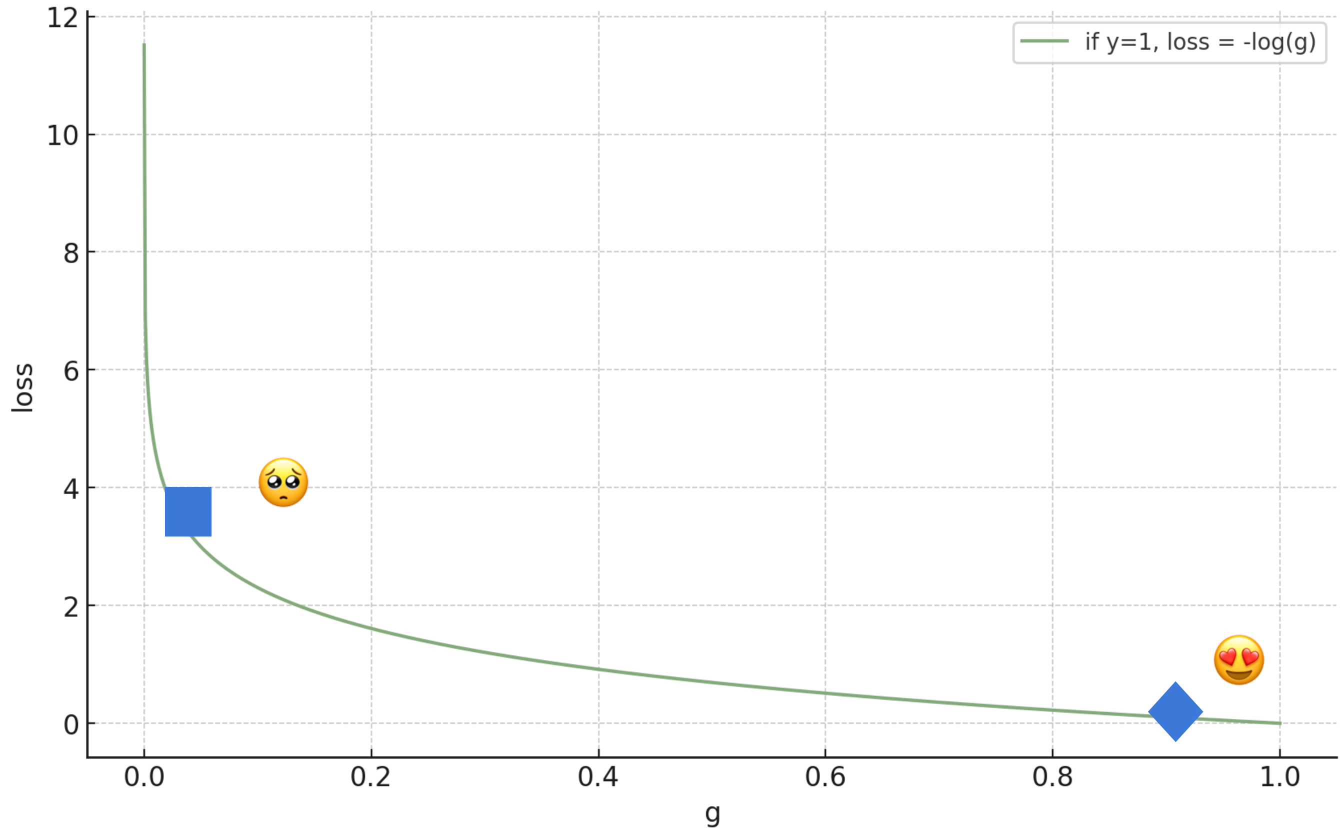
$$g(x) = \sigma(\theta x + \theta_0)$$



$$\mathcal{L}_{\text{nll}}(\text{ guess, actual })$$

$$= -[y \log g + (1 - y)\log(1 - g)]$$

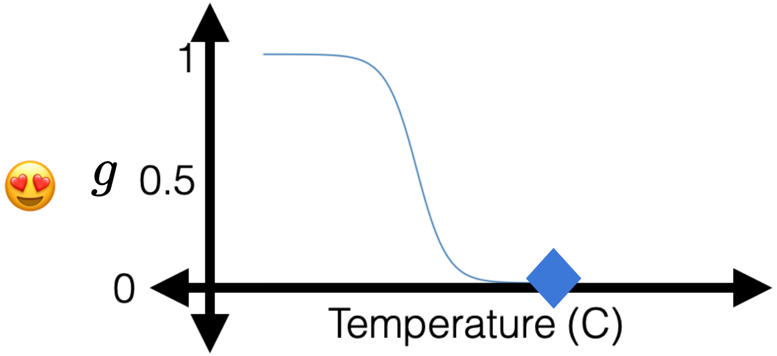
$$= -\log g$$

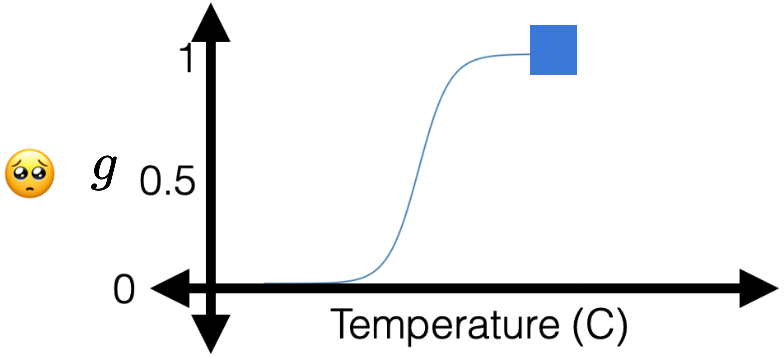


If $y = 0$

training data:

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

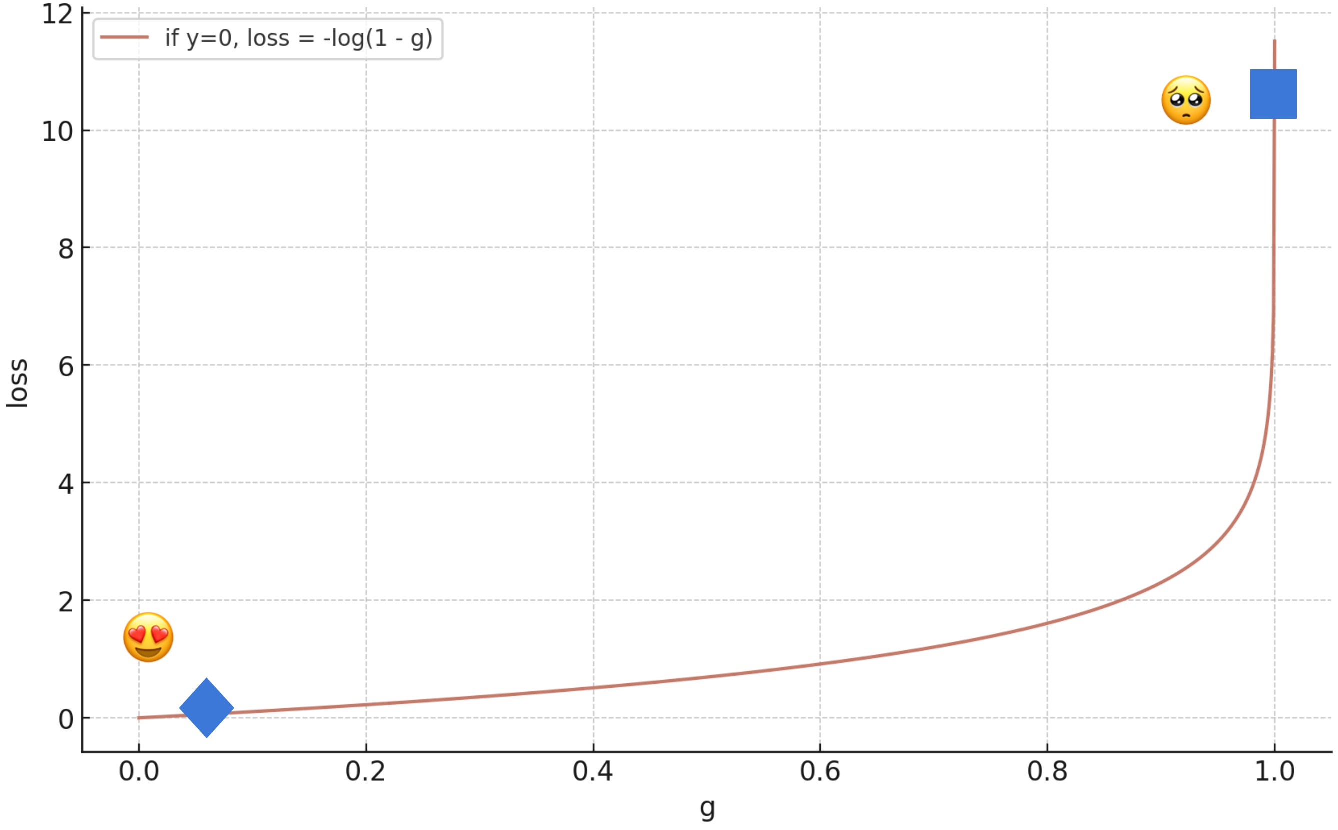




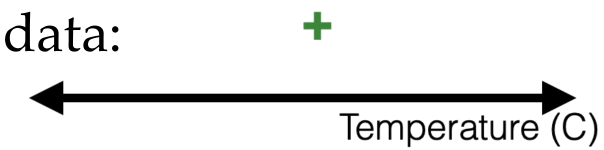
$$\mathcal{L}_{\text{nll}}(\text{ guess, actual })$$

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

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



training data:



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

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

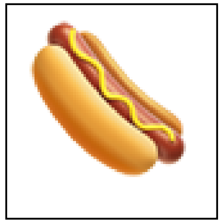
	linear regressor	linear binary classifier	linear *logistic* binary classifier
features	$x \in \mathbb{R}^d$		
parameters	$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}$		
linear combo	$\theta^T x + \theta_0 \quad = z$		
predict	z	$\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 & \text{if } g = \sigma(z) > 0.5 \\ 0 & \text{otherwise} \end{cases}$
loss	$(g-y)^2$	$\begin{cases} 0 & \text{if } g = a \\ 1 & \text{otherwise} \end{cases}$	$-\left[y \log g + (1-y) \log (1-g)\right]$
optimize via	closed-form or gradient descent	NP-hard to learn	• gradient descent only • need regularization to not overfit

Outline

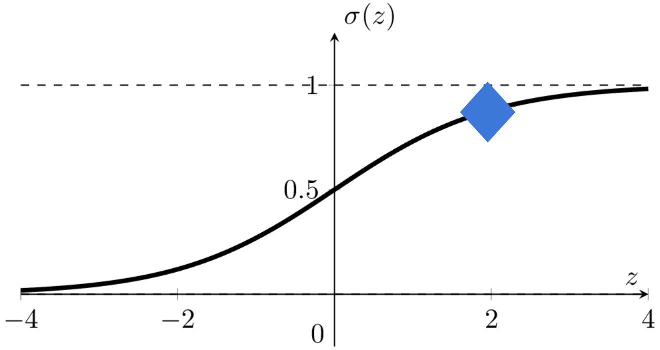
- Linear (binary) classifiers
  - to use: separator, normal vector
  - to learn: difficult! won't do
- Linear logistic (binary) classifiers
  - to use: sigmoid
  - to learn: negative log-likelihood loss
- Multi-class classifiers
  - to use: softmax
  - to learn: one-hot encoding, cross-entropy loss

$\sigma(z)$: model's confidence the input x is a hot-dog

$1 - \sigma(z)$: model's confidence the input x is not a hot-dog



$x \xrightarrow{\theta^T x + \theta_0} z \in \mathbb{R} \longrightarrow$



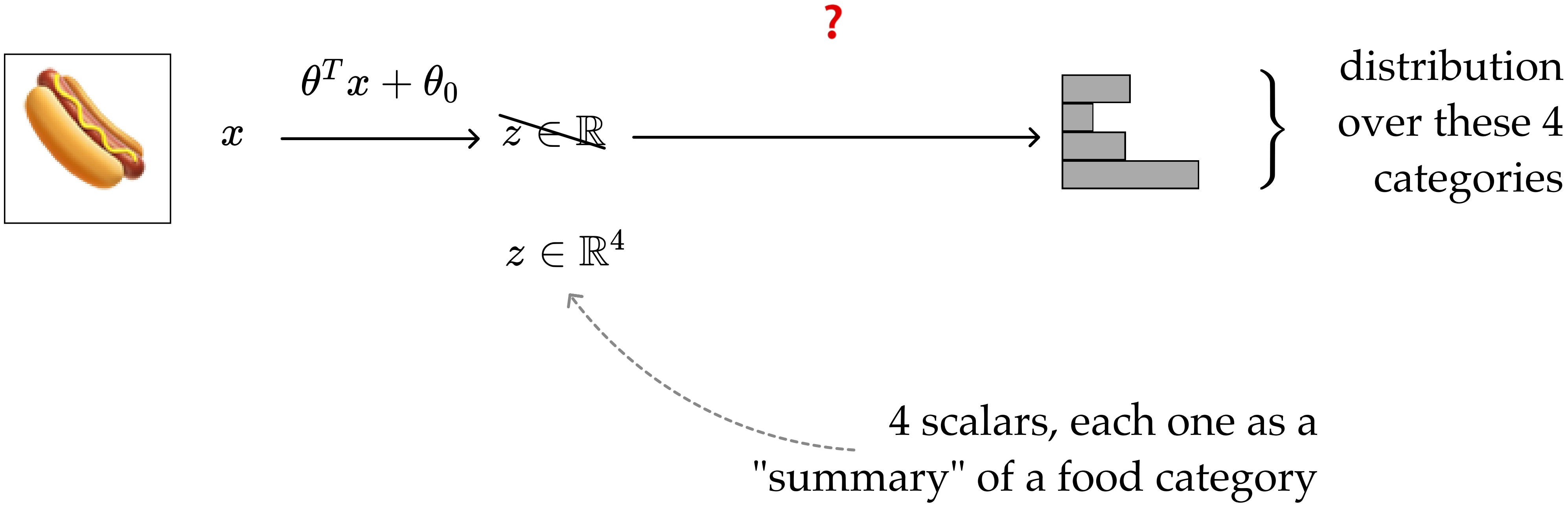
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

learned scalar "summary" of "hot-dog-ness"

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

fixed baseline of "non-hot-dog-ness"

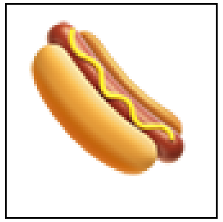
if we want to predict {hot-dog, pizza, pasta, salad}



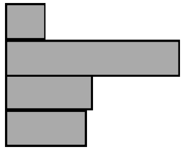
Outline

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

if we want to predict {hot-dog, pizza, pasta, salad}



$x \xrightarrow{\theta^T x + \theta_0} z \in \mathbb{R}^4 \xrightarrow{?}$



distribution over these 4 categories

$$\begin{bmatrix} -0.23 \\ 3.67 \\ 1.47 \\ 0.44 \end{bmatrix} \xrightarrow{\text{softmax}(z_j) = \frac{\exp(z_j)}{\sum_{i=1}^{4} \exp(z_i)}} \begin{bmatrix} 0.0173 \\ 0.8543 \\ 0.0947 \\ 0.0338 \end{bmatrix}$$

entries between (0,1),
sums up to 1

	linear logistic *binary* classifier	one-out-of-K classifier
training data	$x \in \mathbb{R}^d$, $y \in \{0, 1\}$	$x \in \mathbb{R}^d$, y : K-dimensional one-hot
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	$\sigma(z) = \frac{\exp(z)}{\exp(0) + \exp(z)}$ positive if $\sigma(z) > 0.5$	$\text{softmax}(z) = \begin{bmatrix} \exp(z_1) / \sum_i \exp(z_i) \\ \vdots \\ \exp(z_K) / \sum_i \exp(z_i) \end{bmatrix}$ category corresponding to the largest entry in softmax(z)

Outline

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

One-hot encoding:

- Encode the K classes as an $\mathbb{R}^K$ vector, with a single 1 (hot) and 0s elsewhere.
- Generalizes from {0,1} binary labels

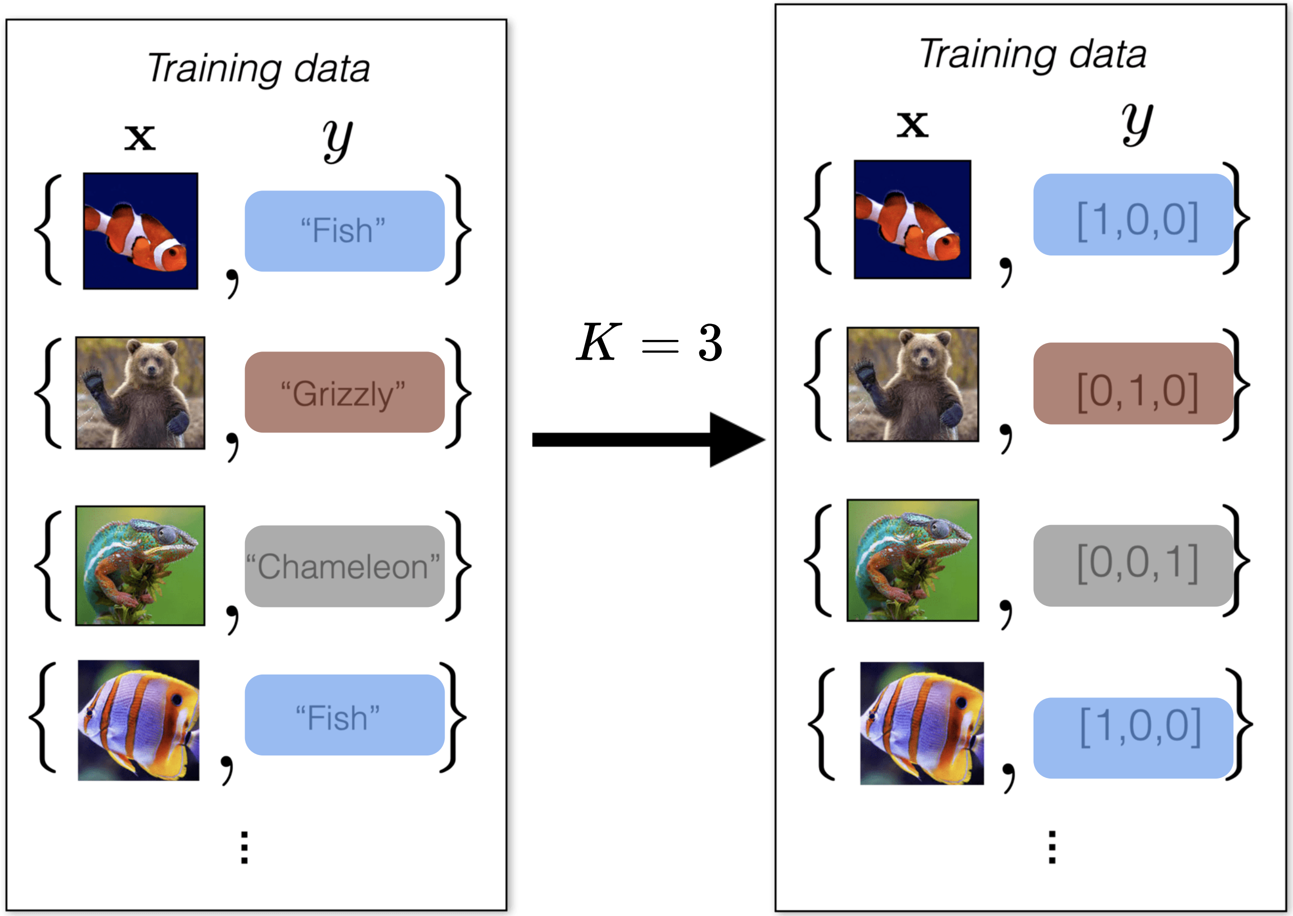


image adapted from Phillip Isola

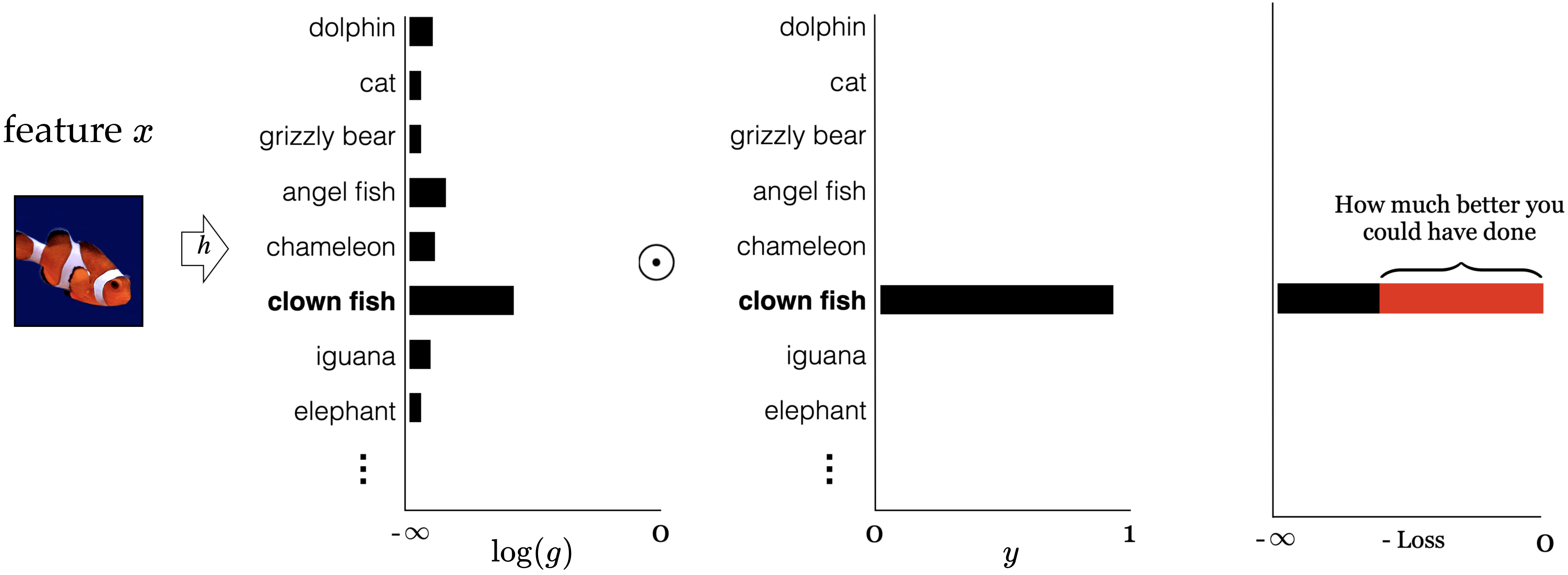
current prediction

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

true label y

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

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



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

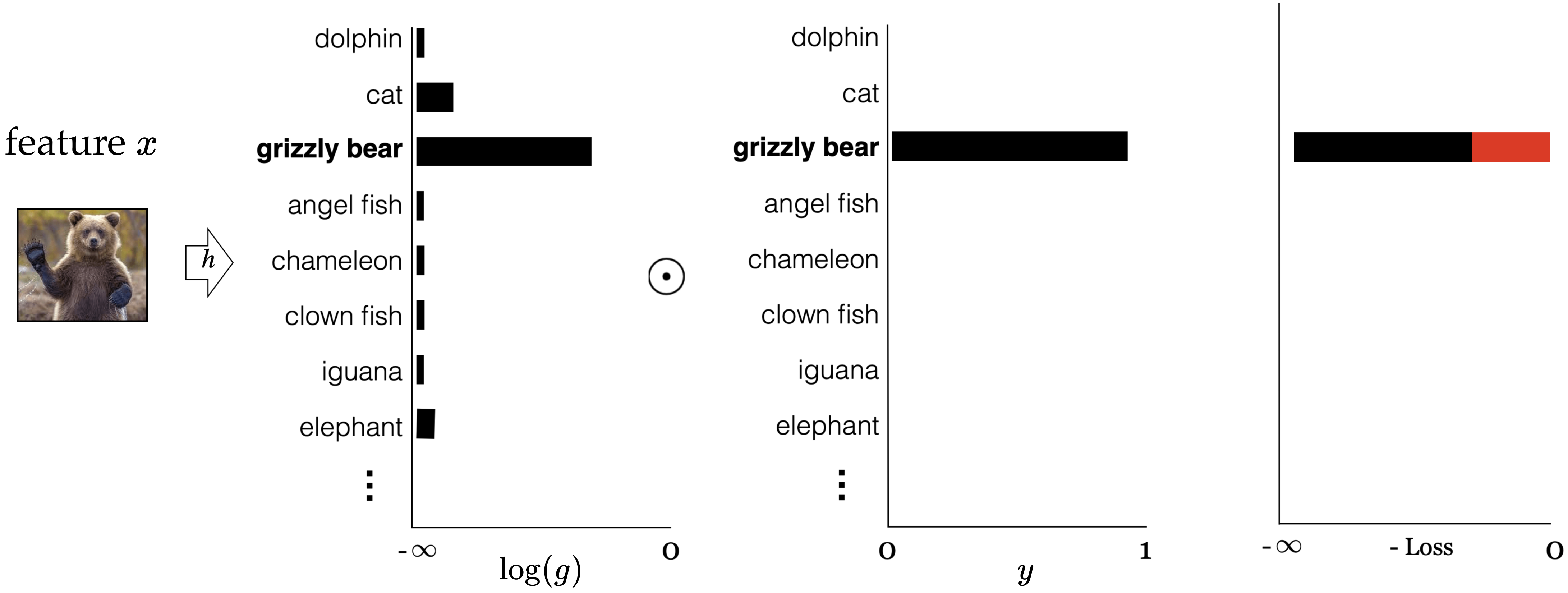
image adapted from Phillip Isola

current prediction

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

true label y

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



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

image adapted from Phillip Isola

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

g : softmax output

g_k : probability or confidence in class k

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

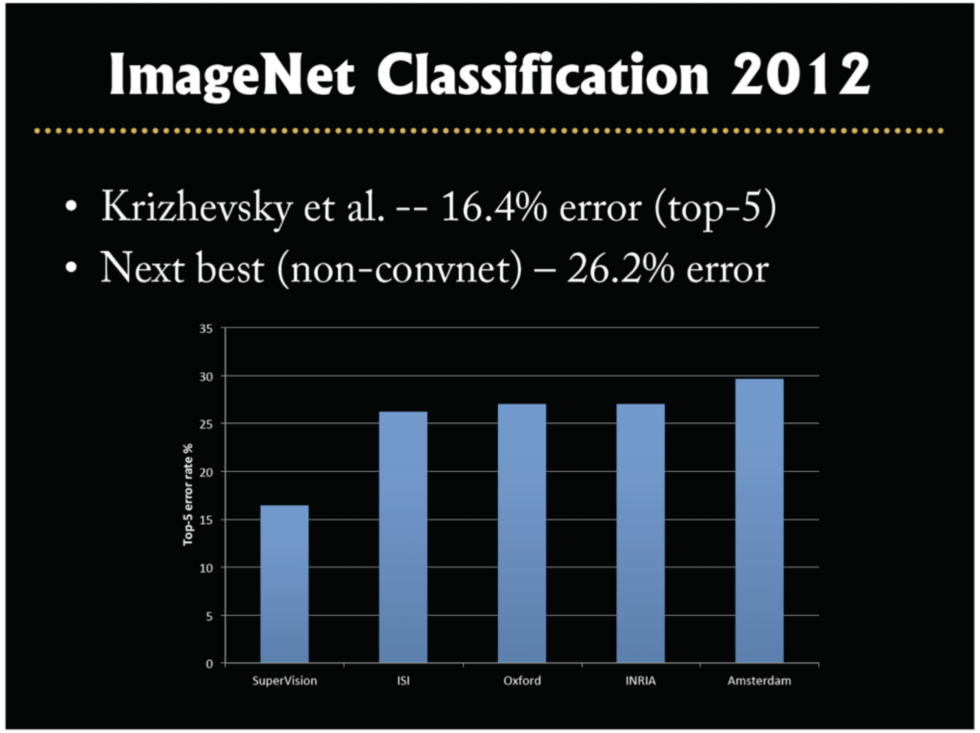
y :one-hot encoding label

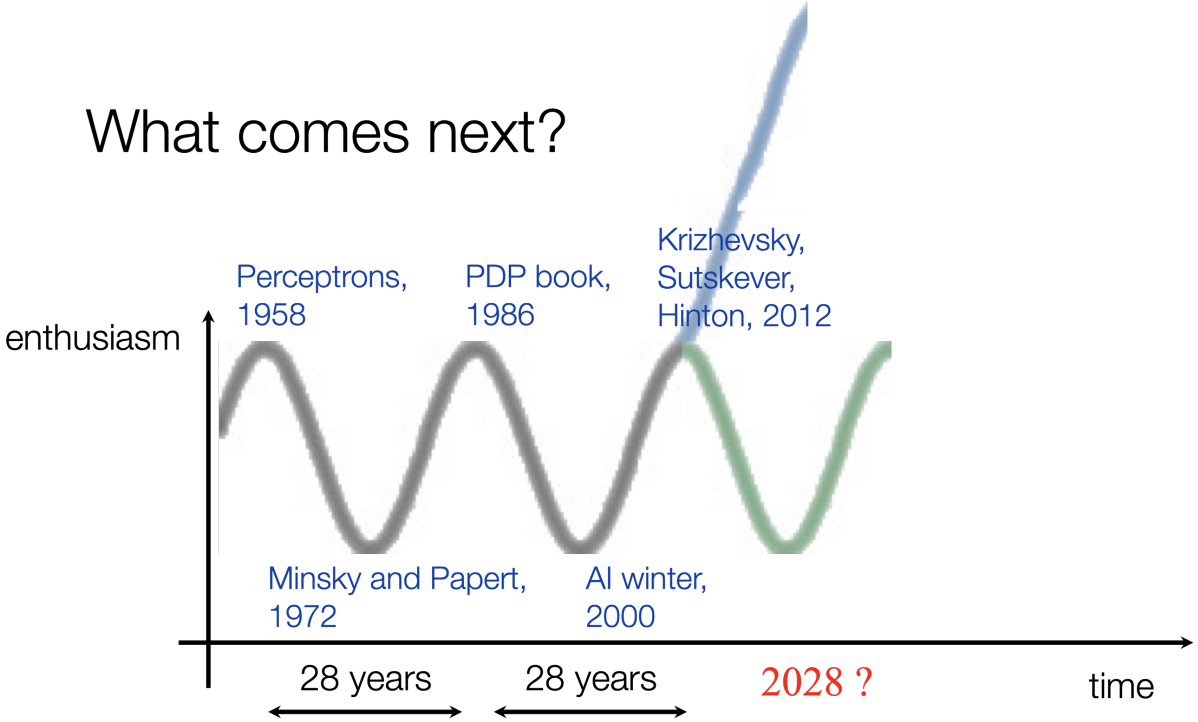
y_k : either 0 or 1

- Generalizes negative log likelihood loss $\mathcal{L}_{\text{nll}}(g, y) = -\left[y \log g + (1-y) \log\left(1-g\right)\right]$
- Appears as summing K terms, but
- for a given data point, only the term corresponding to its true class label matters.

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 θ and θ_0 as defining a d-1 dimensional 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: motivation and properties.
- Negative-log-likelihood loss: smoother and has nice probabilistic motivations. We 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!