https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 4: Linear Classification (Logistic Regression)

Shen Shen

Feb 23, 2024

(some slides adapted from Tamara Broderick and Phillip Isola)

6.390

Introduction to Machine Learning (Spring 2024)

Announcements for Week 3 (Mon, Feb 19 - Fri, Feb 23)

Keep an eye out for weekly announcements on IntroML homepage.

- The Registrar has posted the MIT Final Exams Schedule. The 6.390 final will be held on Monday, May 20 2024, from 1:30 PM to 4:30 PM Eastern Time, at Johnson Track. The Registrar will also schedule a Conflict Exam for 6.390, and will announce that schedule after Drop Date. For cross-registered 6.390 students who have a schedule conflict with the May 20 final exam, we expect to make the 6.390 conflict exam time (as scheduled by the MIT Registrar) available to them as well.

- There is a student in our class who needs copies of class notes as an approved accommodation. If you're interested in serving as a paid note taker, please reach out to DAS, at 617-253-1674 or `das-student@mit.edu`. More details of the job can be found here.

Outline

- Recap (ML pipeline, regression, regularization, GD)
- Classification General Setup
- (vanilla) Linear Classifier
  - Understand a *given* linear classifier
  - Linear separator: geometric intuition
  - *Learn* a linear classifier via 0-1 loss?
- Linear Logistic Regression
  - Sigmoid function
  - Cross-entropy (negative log likelihood) loss
  - Optimizing the loss via gradient descent
  - Regularization, cross-validation still matter
- Multi-class classification

Data $\rightarrow$

ML algorithm

Hypothesis class

Hyperparameters

If/how to add regularization

Objective (loss) functions

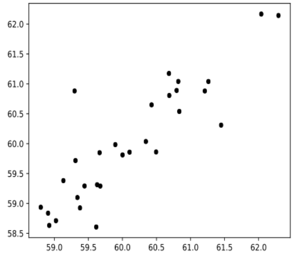
$\rightarrow h$

$\uparrow$

Compute/optimize

Training

Data

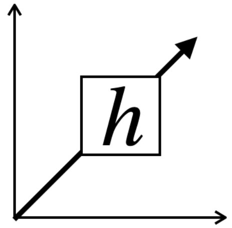


→

ML algorithm

Hypothesis class
Hyperparameters
If/how to add regularization
Objective (loss) functions

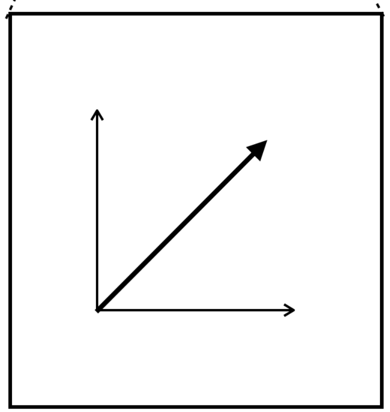
→



Recap:

- OLS can have analytical formula and "easy" prediction mechanism
- Regularization
- Cross-validation
- Gradient descent

Testing (predicting)

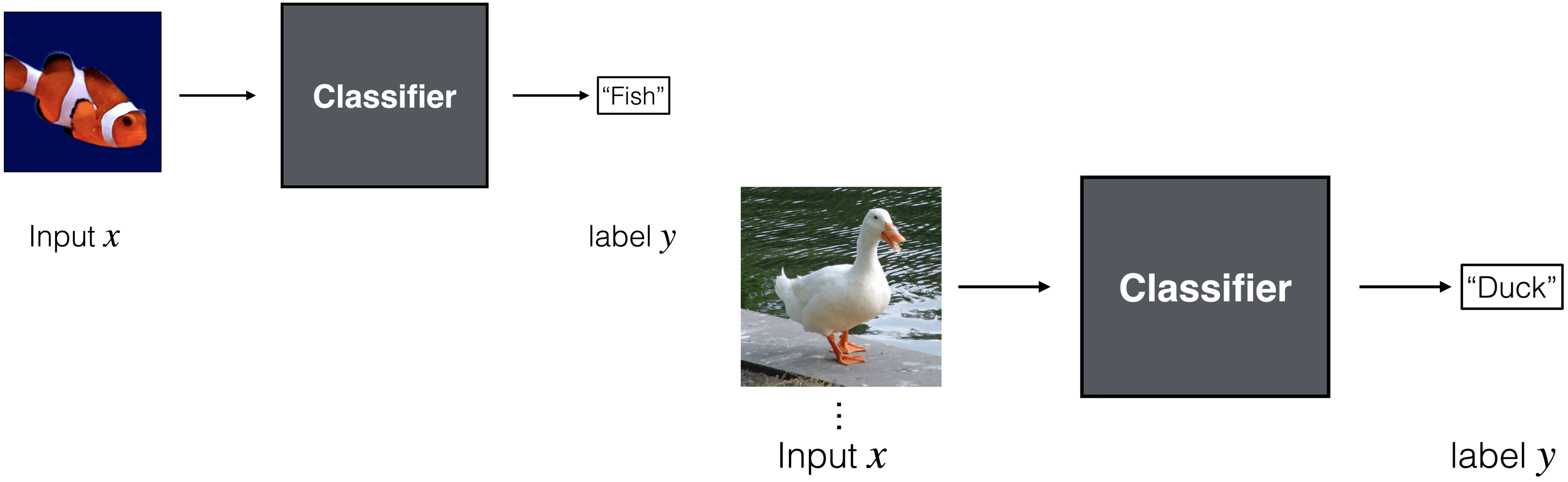
new input x →  → new prediction y

Outline

- Recap (ML pipeline, regression, regularization, GD)
- Classification General Setup
- (vanilla) Linear Classifier
  - Understand a *given* linear classifier
  - Linear separator: geometric intuition
  - *Learn* a linear classifier via 0-1 loss?
- Linear Logistic Regression
  - Sigmoid function
  - Cross-entropy (negative log likelihood) loss
  - Optimizing the loss via gradient descent
  - Regularization, cross-validation still matter
- Multi-class classification

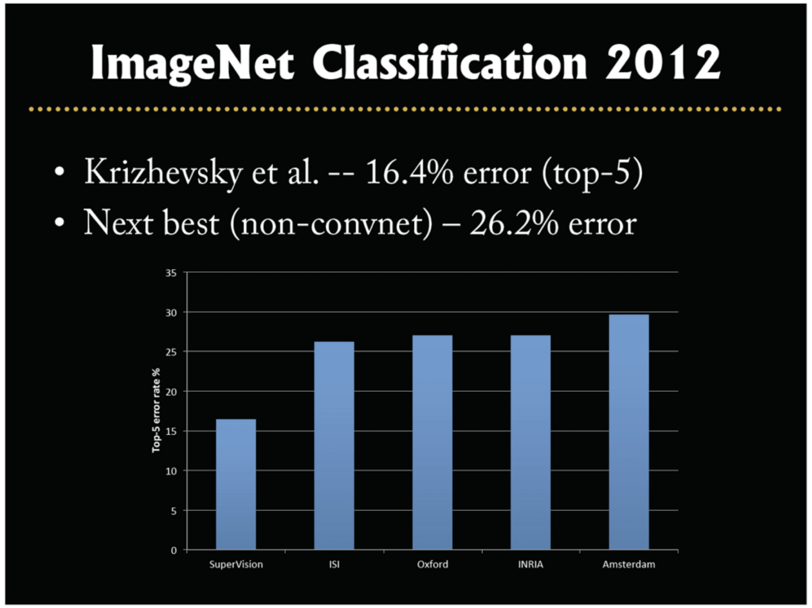
Classification Setup

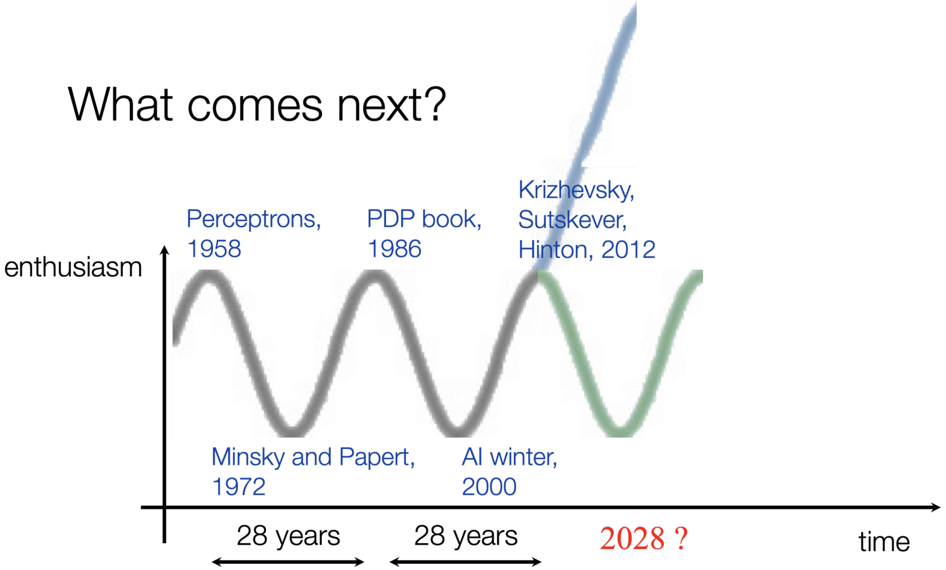
- General setup: Labels (and predictions) are in a discrete set



Classification Setup

- General setup: Labels (and predictions) are in a discrete set





Outline

- Recap (ML pipeline, regression, regularization, GD)
- Classification General Setup
- (vanilla) Linear Classifier
  - Understand a *given* linear classifier
  - Linear separator: geometric intuition
  - *Learn* a linear classifier via 0-1 loss?
- Linear Logistic Regression
  - Sigmoid function
  - Cross-entropy (negative log likelihood) loss
  - Optimizing the loss via gradient descent
  - Regularization, cross-validation still matter
- Multi-class classification

(vanilla) Linear Classifier

- General setup: Labels (and predictions) are in a discrete set
- Simplest setup: linear binary classification. that is, two possible labels, e.g.$y \in$ {positive, negative} (or {dog, cat}, {pizza, not pizza}, {+1, 0}...)
  - given a data point with features $x_1, x_2, \ldots x_d$
  - do some linear combination, calculate $z = (\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d) + \theta_0$
  - make a prediction: predict positive class if $z > 0$ otherwise negative class.
- We need to understand what are:
  - Linear separator
  - Normal vector
  - Linear separability

(The demo won't embed in PDF. But the direct link below works.)

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

0-1 loss

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

- 😊 Very intuitive
- 😊 Easy to evaluate
- 🥺 Very hard to optimize (NP-hard)
    - "Flat" almost everywhere (those local gradient=0, not helpful)
    - Has "jumps" elsewhere (don't have gradient there)

(The demo won't embed in PDF. But the direct link below works.)

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

Outline

- Recap (ML pipeline, regression, regularization, GD)
- Classification General Setup
- (vanilla) Linear Classifier
  - Understand a *given* linear classifier
  - Linear separator: geometric intuition
  - *Learn* a linear classifier via 0-1 loss?
- Linear Logistic Regression
  - Sigmoid function
  - Cross-entropy (negative log likelihood) loss
  - Optimizing the loss via gradient descent
  - Regularization, cross-validation still matter
- Multi-class classification

Linear Logistic Regression

- Despite regression in the name, really a hypothesis class for classification
- Mainly motivated to solve the non-"smooth" issue of "vanilla" linear classifier (where we used sign() function and 0-1 loss)
- But has nice probabilistic interpretation too
- Concretely, we need to know:
    - Sigmoid function
    - Cross-entropy (negative log likelihood) loss
    - Optimizing the loss via gradient descent
    - Regularization, cross-validation still matter

Recall: (Vanilla) Linear Classifier

- calculate $z = (\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d) + \theta_0$
- predict positive class if $z > 0$ otherwise negative class.

Linear Logistic Regression

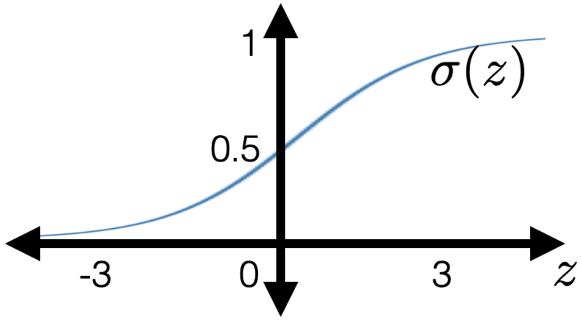
- calculate $z = (\theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d) + \theta_0$
- "squish" z with a sigmoid / logistic function:

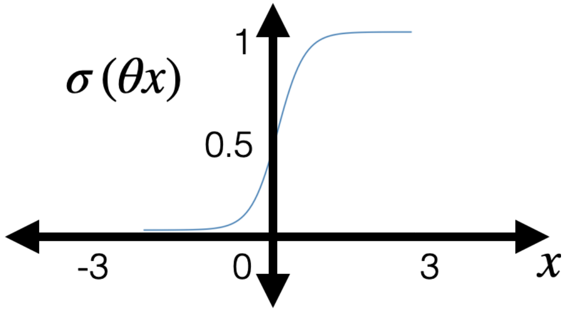
$$g = \sigma(z) = \frac{1}{1 + \exp(-z)}$$

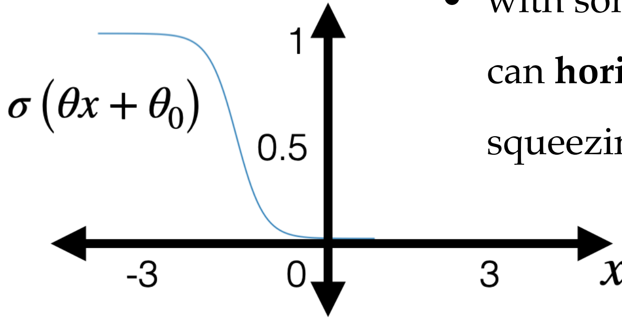
- predict positive class if $g > 0.5$, otherwise, negative class.

Comments about sigmoid

- **vertically** always monotonically "sandwiched" between 0 and 1 (and never quite get to either 0 or 1)
- **probabilistic** interpretation
- very **nice/elegant** gradient
- with some appropriate θ, θ_0 can **horizontally** flip, squeezing, expanding, shift





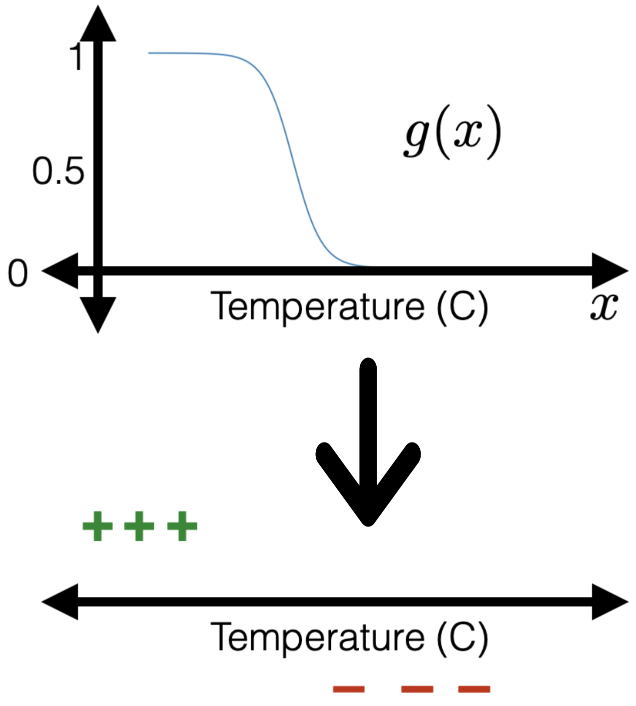


e.g. suppose, wanna predict whether to bike to school.

with **given** parameters, how do I make prediction?

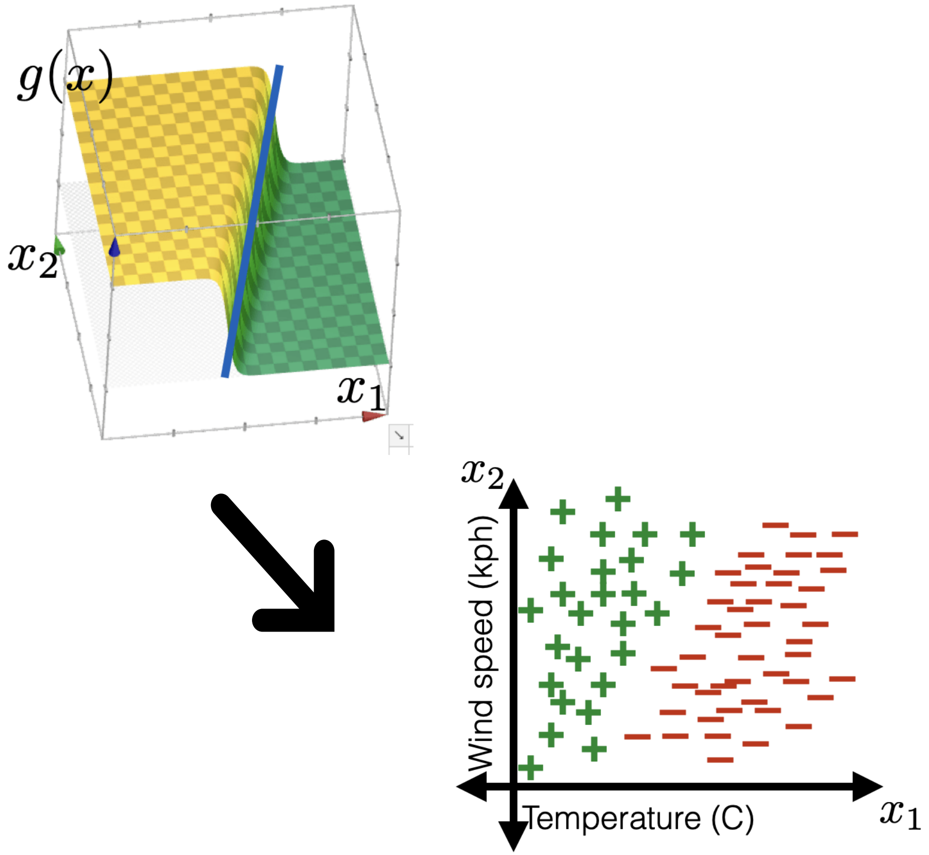
1 feature: $g(x) = \sigma\left(\theta x + \theta_0\right)$

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



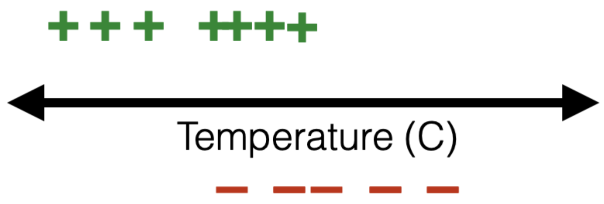
2 features: $g(x) = \sigma\left(\theta^\top x + \theta_0\right)$

$$= \frac{1}{1 + \exp\left\{-\left(\theta^\top x + \theta_0\right)\right\}}$$

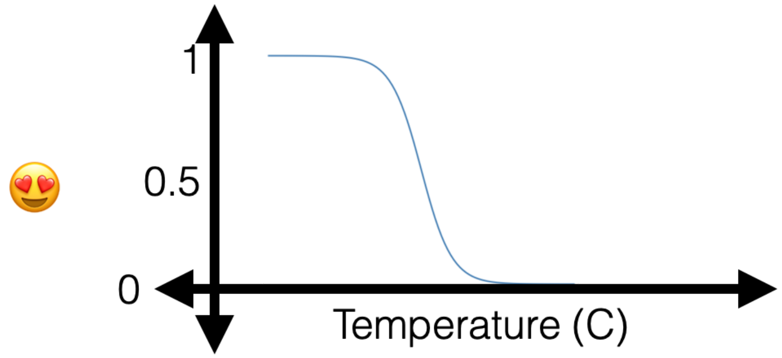


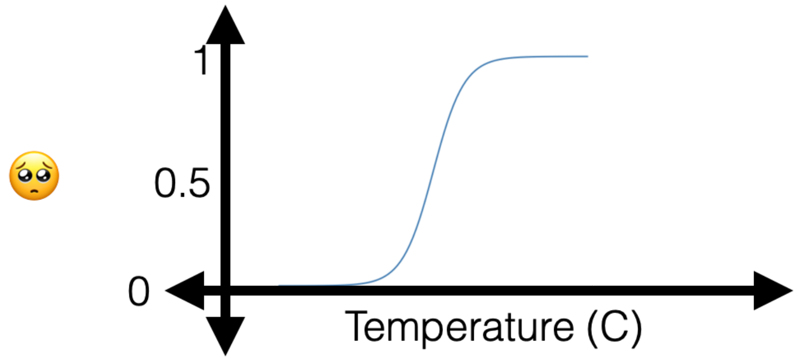
Learning a logistic regression classifier

training data:



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





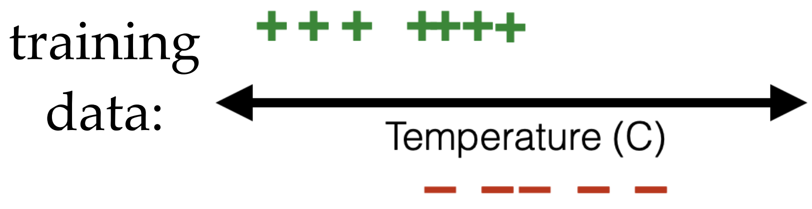
- Suppose labels $y \in \{+1, 0\}$
- When see a training datum i with $y^{(i)} = 1$, would like $g^{(i)}$ be high
- When see a training datum i with $y^{(i)} = 0$, would like $1 - g^{(i)}$ be high
- i.e. for ith training data point, want this probability (likelihood)

$$\begin{cases} g^{(i)} & \text{if } y^{(i)} = 1 \\ 1 - g^{(i)} & \text{if } y^{(i)} = 0 \end{cases}$$

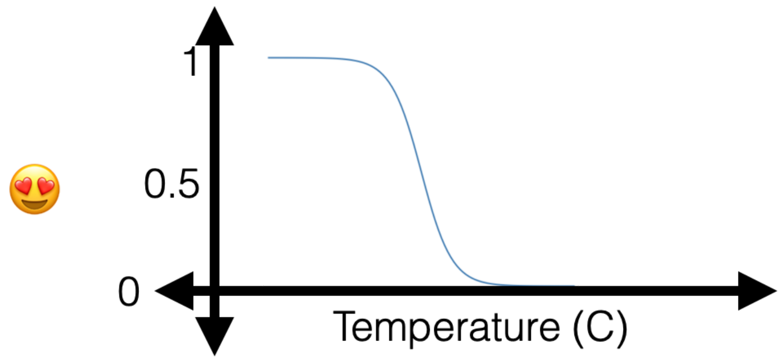
to be high.

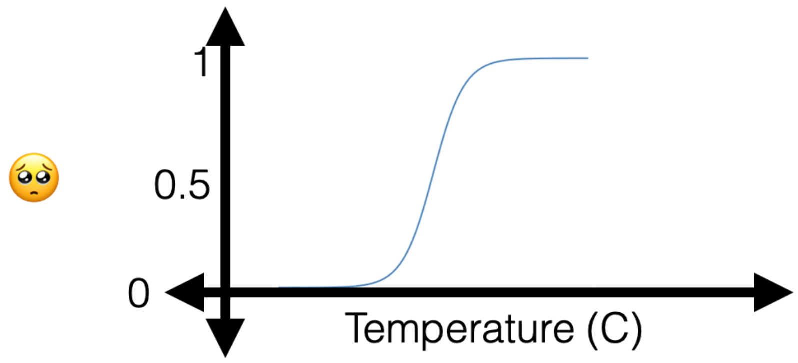
- or, equivalently, want $g^{(i)y^{(i)}}\left(1 - g^{(i)}\right)^{1-y^{(i)}}$ to be high

Learning a logistic regression classifier



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





- Suppose labels $y \in \{+1, 0\}$
- For training data point i, would like $g^{(i)y^{(i)}} \left(1 - g^{(i)}\right)^{1-y^{(i)}}$ to be high
- As logarithm is monotonic, would like $y^{(i)} \log g^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - g^{(i)}\right)$ to be high
- Add a negative sign, to turn the above into a loss

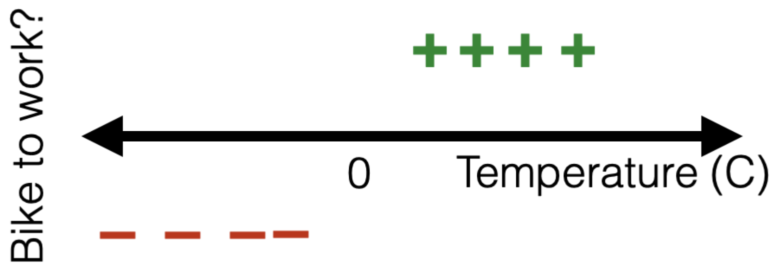
$$\mathcal{L}_{\text{nll}}\left(g^{(i)}, y^{(i)}\right) = \mathcal{L}_{\text{nll}}(\text{ guess, actual }) =$$

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

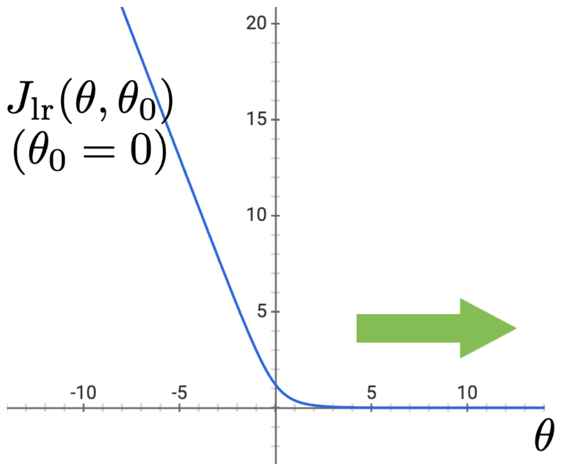
- Want the above to be low for all data points, under i.i.d. assumption, equivalently, wanna minimize $J_{lr} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\text{nll}}\left(g^{(i)}, y^{(i)}\right) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\text{nll}}\left(\sigma\left(\theta^{\top} x^{(i)} + \theta_0\right), y^{(i)}\right)$

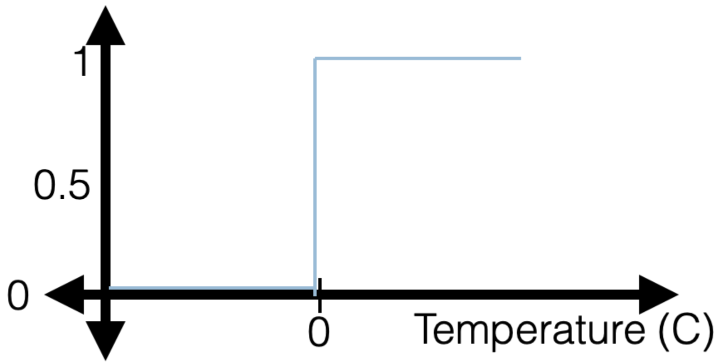
Comments about $J_{lr} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\text{nll}}\left(\sigma\left(\theta^\top x^{(i)} + \theta_0\right), y^{(i)}\right)$

- Also called cross-entropy loss
- Convex, differentiable with **nice** (elegant) gradients
- Doesn't have a closed-form solution
- Can still run gradient descent
- But, a gotcha: when training data is linearly separable



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

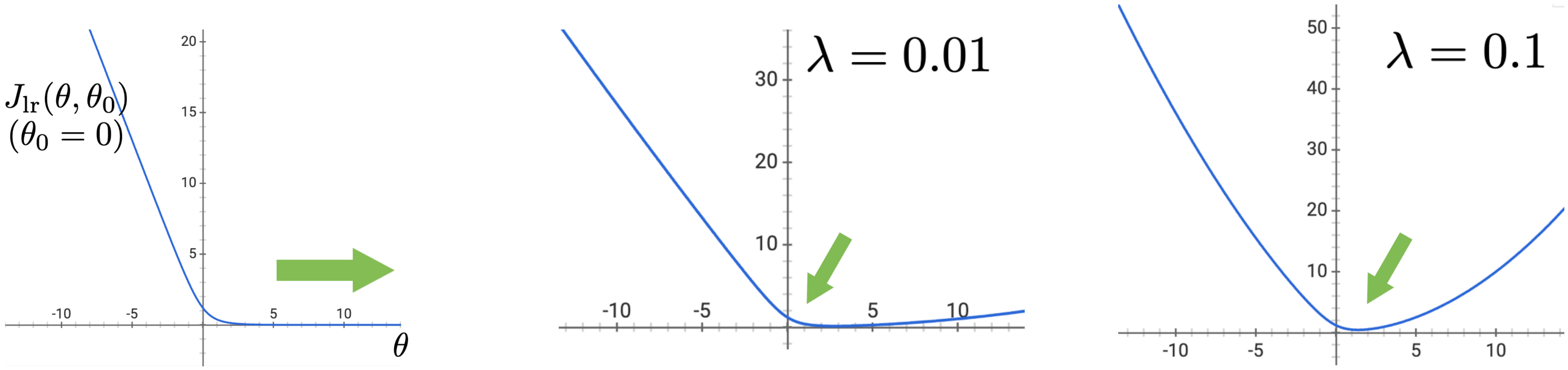




Regularization for Logistic Regression

$$\mathrm{J}_{\mathrm{lr}}\left(\theta, \theta_0; \mathcal{D}\right) = \left(\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_{\mathrm{nll}}\left(\sigma\left(\theta^\top x^{(i)} + \theta_0\right), y^{(i)}\right)\right) + \lambda\|\theta\|^2$$

- $\lambda \geq 0$
- No regularizing θ_0 (think: why?)
- Penalizes being overly certain
- Objective is still differentiable & convex (gradient descent)

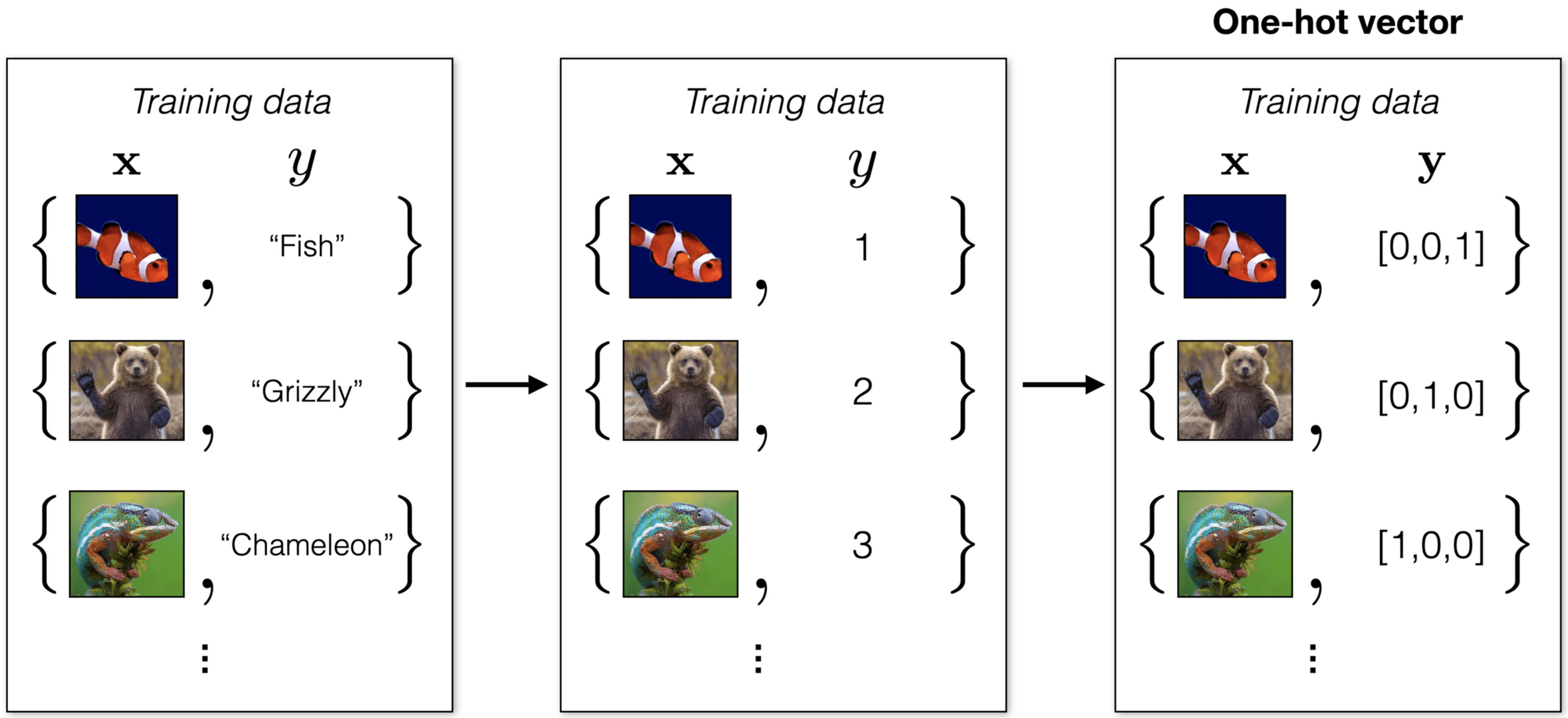


Outline

- Recap (ML pipeline, regression, regularization, GD)
- Classification General Setup
- (vanilla) Linear Classifier
  - Understand a *given* linear classifier
  - Linear separator: geometric intuition
  - *Learn* a linear classifier via 0-1 loss?
- Linear Logistic Regression
  - Sigmoid function
  - Cross-entropy (negative log likelihood) loss
  - Optimizing the loss via gradient descent
  - Regularization, cross-validation still matter
- Multi-class classification

How to represent class labels?

Suppose K classes, then it's convenient to let y be a K-dimensional one-hot vector



Generalize sigmoid to softmax

two classes

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

scalar

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

scalar

K classes

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

K-by-1

$$g = \text{softmax}(z) = \begin{bmatrix} \exp(z_1) / \sum_i \exp(z_i) \\ \vdots \\ \exp(z_K) / \sum_i \exp(z_i) \end{bmatrix}$$

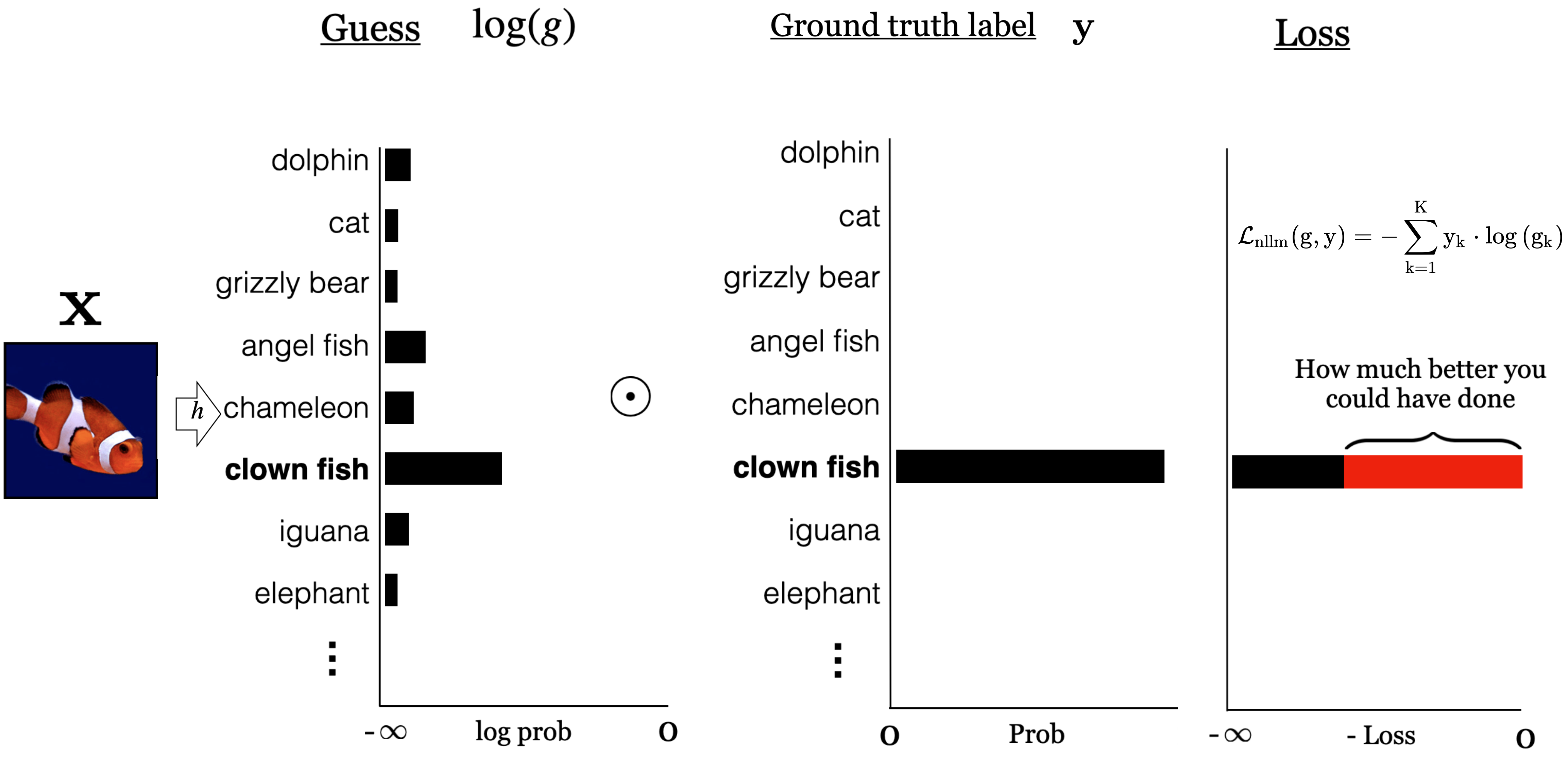
K-by-1

Generalize NLL to NLL multi-class (NLLM, or just cross-entropy)

Every data point incur a scalar loss:

$$\mathcal{L}_{\text{nll}}(\mathrm{g}, \mathrm{y}) = -\left(y \log g + (1 - y) \log (1 - g)\right)$$

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


Guess $\log(g)$
Ground truth label $\mathbf{y}$
Loss
$\mathbf{x}$
h
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮
$\odot$
- ∞
log prob
0
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
⋮
0
Prob
$\mathcal{L}_{\mathrm{nllm}}(\mathrm{g}, \mathrm{y}) = -\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{y}_{\mathrm{k}} \cdot \log(\mathrm{g}_{\mathrm{k}})$
How much better you could have done
- ∞
- Loss
0

Summary

- Classification is 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 d-1 dimensional hyperplane that cuts the d-dimensional input space into two half-spaces. (This is hard conceptually!)
- 0-1 loss is a natural loss function for classification, BUT, hard to optimize. (Non-smooth; zero-gradient)
- NLL is smoother and has nice probabilistic motivations. We can optimize using gradient descent!
- Regularization is still important.
- Generalizes to multi-class.

We'd love it for you to share some lecture feedback.

Thanks!