Announcements

- **Homework 1 updated**
  - Due next Wednesday at 8pm
  - Covers linear regression
  - **Make sure you use the Spring 2025 version, you will not receive credit if you submit the Fall 2024 version (or early)**

- Office hours have started, see course website for details

Lecture 5: Logistic Regression (Part 1)

CIS 4190/5190

Spring 2025

Supervised Learning

Data $Z = \{(x_i, y_i)\}_{i=1}^{n}$

$\hat{\beta}(Z) = \arg\min_\beta L(\beta; Z)$

L encodes $y_i \approx f_\beta(x_i)$

Model $f_{\hat{\beta}(Z)}$

Classification

Data $Z = \{(x_i, y_i)\}_{i=1}^{n}$

$\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$

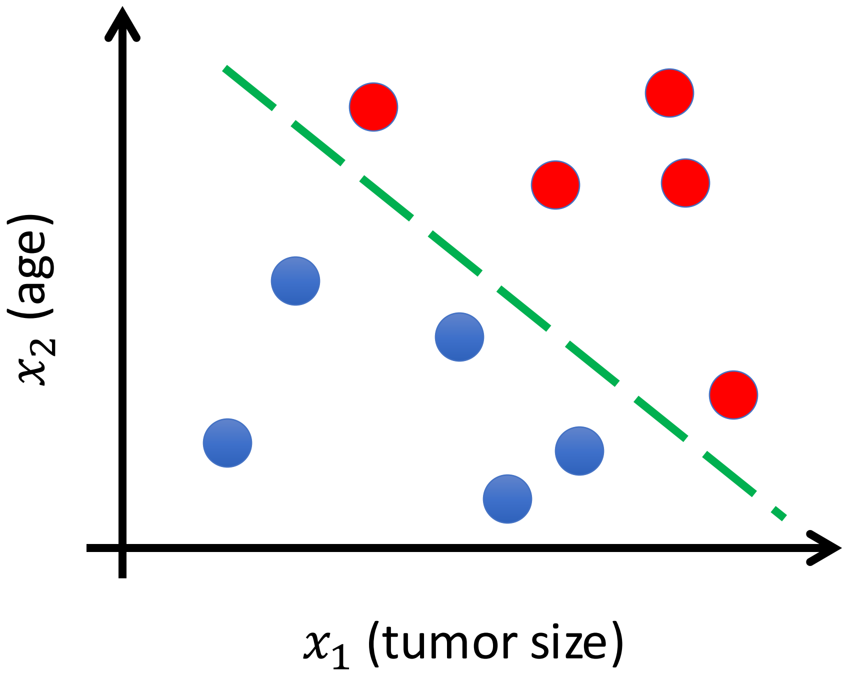
L encodes $y_i \approx f_{\beta}(x_i)$

Model $f_{\hat{\beta}(Z)}$

Label is a **discrete value** $y_i \in \mathcal{Y} = \{1, \ldots, k\}$

(Binary) Classification

- **Input:** Dataset $Z = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
- **Output:** Model $y_i \approx f_\beta(x_i)$



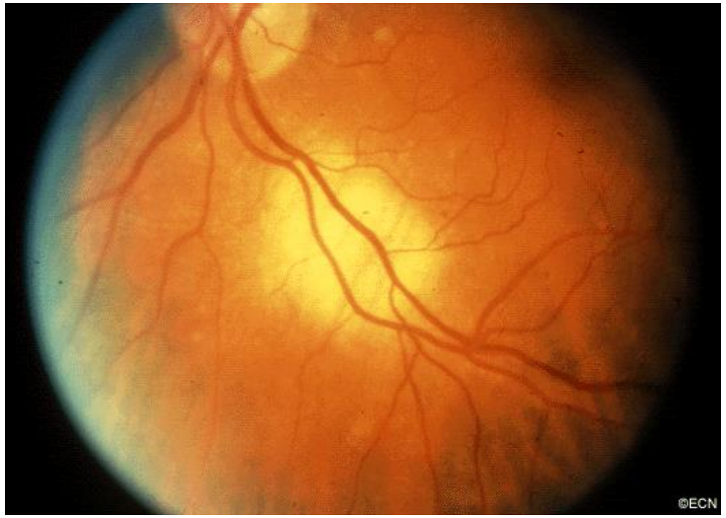


Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/

Example: Malignant vs. Benign Ocular Tumor

Loss Minimization View of ML

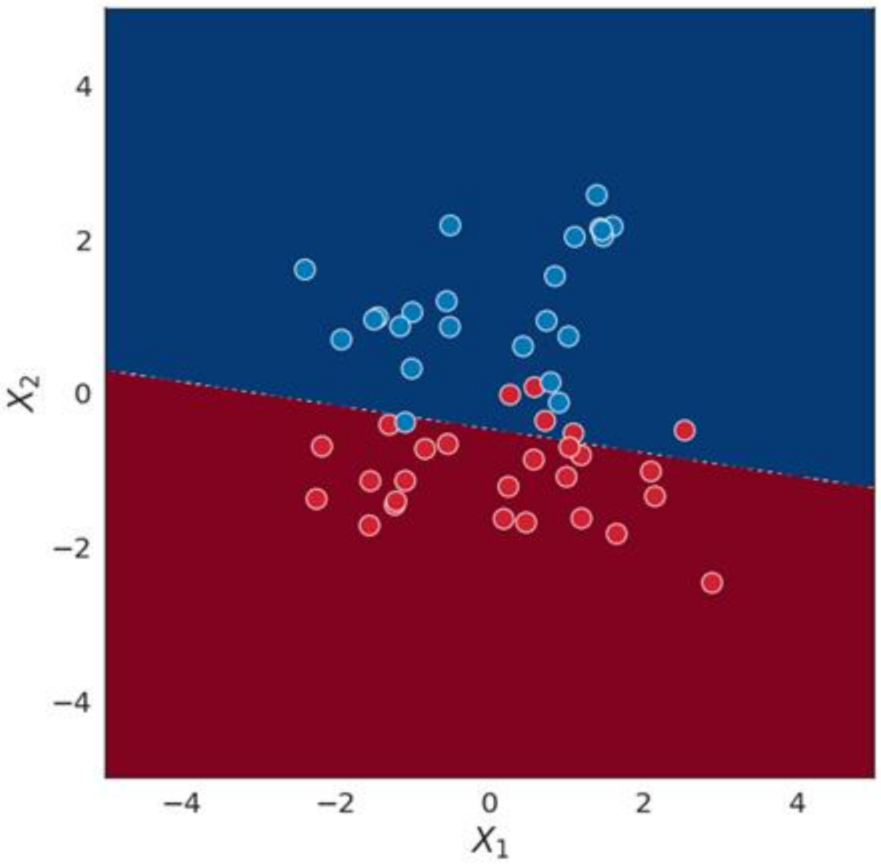
- **Three design decisions**
  - **Model family:** What are the candidate models f? (E.g., linear functions)
  - **Loss function:** How to define “approximating”? (E.g., MSE loss)
  - **Optimizer:** How do we optimize the loss? (E.g., gradient descent)

- How do we adapt to classification?

Linear Functions for (Binary) Classification

- **Input:** Dataset $Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

- **Classification:**
  - Labels $y_i \in \{0, 1\}$
  - Predict $y_i \approx 1(\beta^\top x_i \geq 0)$
  - $1(C)$ equals 1 if C is true and 0 if C is false
  - How to learn β? **Need a loss function!**

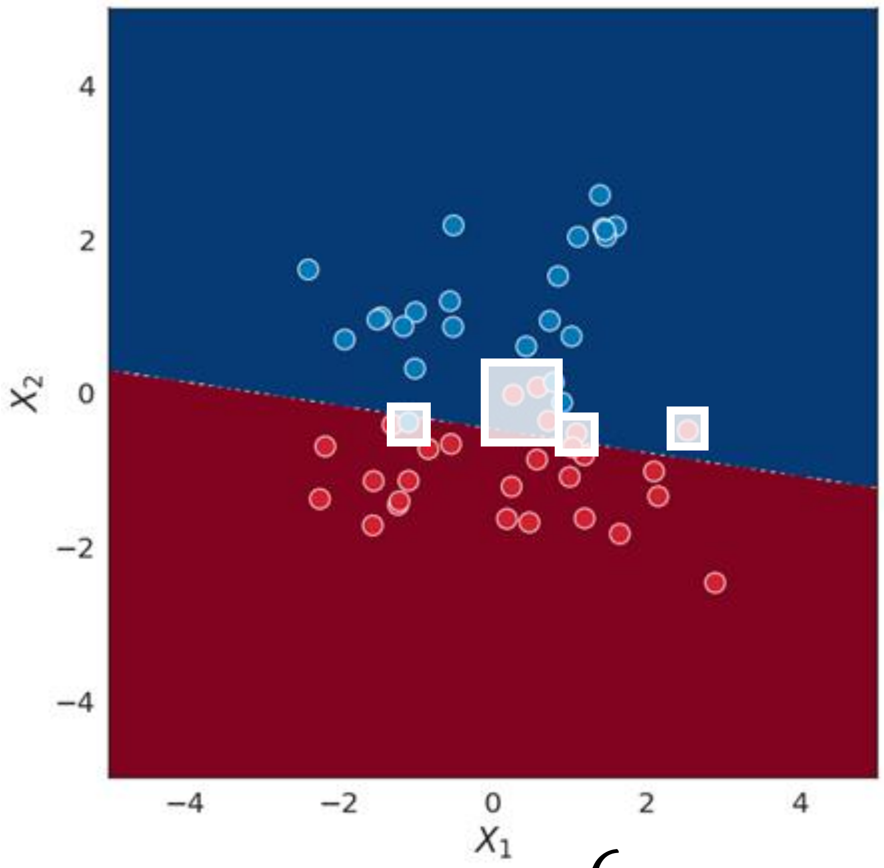


Loss Functions for Linear Classifiers

- **(In)accuracy:**

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n} 1\left(y_i \neq f_\beta(x_i)\right)$$

- Computationally intractable
- Often, but not always the "true" loss (e.g., imbalanced data)



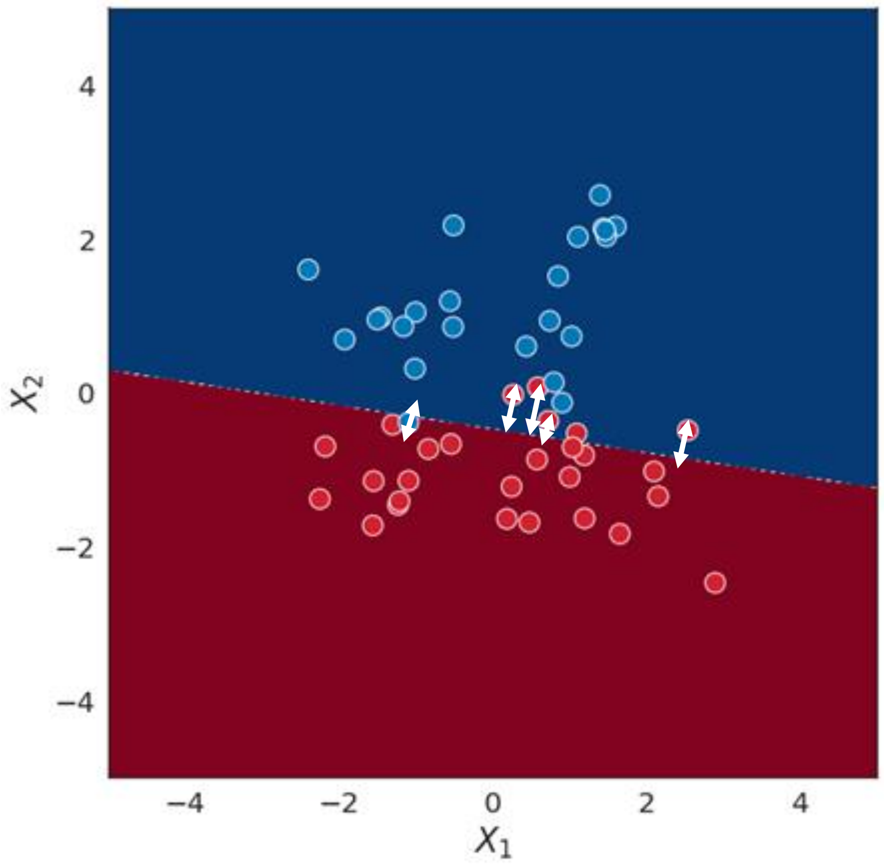
$$L(\beta; Z) = \frac{6}{50}$$

Loss Functions for Linear Classifiers

- **Distance:**

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{dist}(x_i, f_\beta) \cdot 1\big(f_\beta(x_i) \neq y_i\big)$$

- If $L(\beta; Z) = 0$, then 100% accuracy
- Variant of this loss results in SVM
- We consider a more general strategy



$L(\beta; Z) = 1.2$

Maximum Likelihood Estimation

- A **probabilistic** viewpoint on learning (from statistics)

- Given x_i, **suppose** y_i is drawn i.i.d. from distribution $p_{Y|X}(\, Y = y \mid x; \beta \,)$ with parameters β (or density, if y_i is continuous):

$$y_i \sim p_{Y|X}(\,\cdot \mid x_i; \beta\,)$$

Y is random variable, not vector

- Typically write $p_\beta(\, Y = y \mid x \,)$ or just $p_\beta(\, y \mid x \,)$
  - Called a **model** (and $\{p_\beta\}_\beta$ is the **model family**)
  - Will show up convert p_β to f_β later

Maximum Likelihood Estimation

- **Compare to loss function minimization:**
  - **Before:** $y_i \approx f_\beta(x_i)$
  - **Now:** $y_i \sim p_\beta(\cdot | \, x_i; \beta \,)$

- **Intuition the difference:**
  - $f_\beta(x_i)$ just provides a point that y_i should be close to
  - $p_\beta(\cdot | \, x_i; \beta \,)$ provides a score for each possible y_i

- Maximum likelihood estimation combines the **loss function** and **model family** design decisions

Maximum Likelihood Estimation

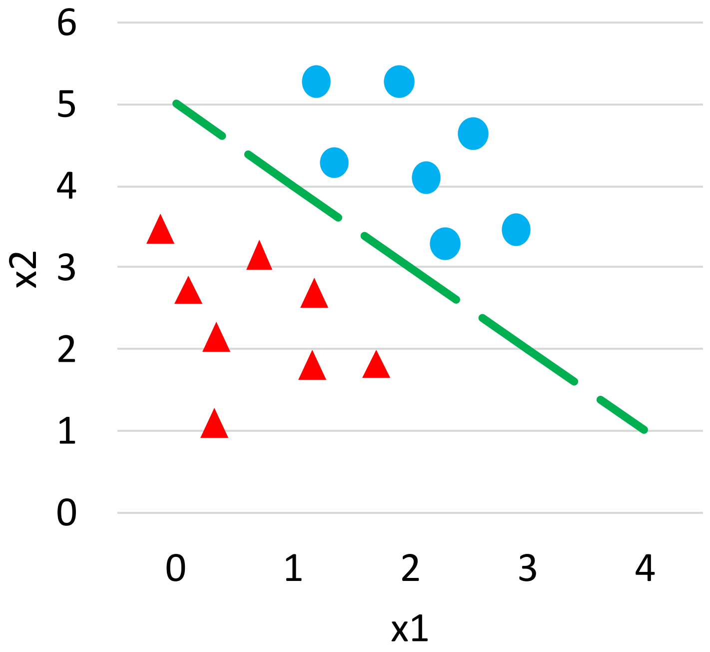
- **Likelihood:** Given model p_β, the probability of dataset Z (replaces loss function in loss minimization view):

$$L(\beta; Z) = p_\beta(Y \mid X) = \prod_{i=1}^{n} p_\beta(y_i \mid x_i)$$

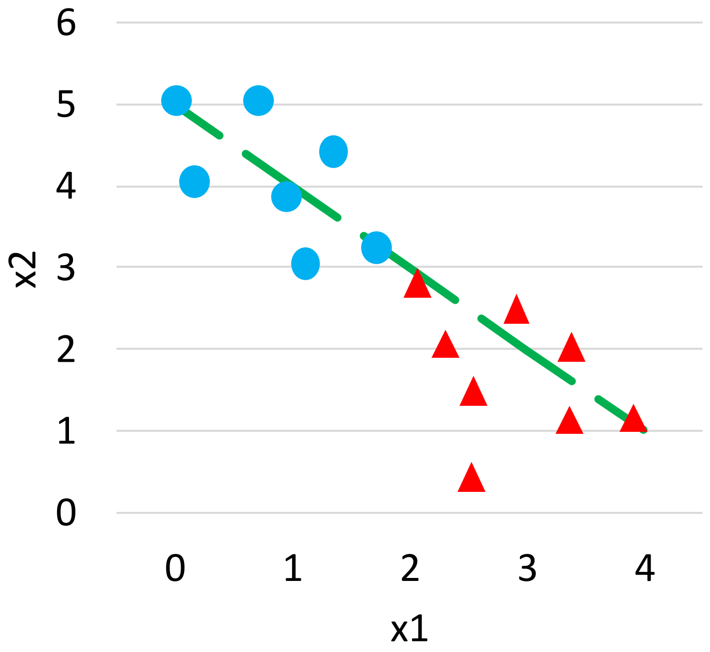
- **Negative Log-likelihood (NLL):** Computationally better behaved form:

$$\ell(\beta; Z) = -\log L(\beta; Z) = -\sum_{i=1}^{n} \log p_\beta(y_i \mid x_i)$$

Intuition on the Likelihood



High likelihood
(Low NLL)



Low likelihood
(High NLL)

Example: Linear Regression

- Assume that the conditional density is

$$p_\beta(y_i \mid x_i) = N(y_i; \beta^\top x_i, 1) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(\beta^\top x_i - y_i)^2}{2}}$$

- $N(y; \mu, \sigma^2)$ is the density of the normal (a.k.a. Gaussian) distribution with mean μ and variance σ^2

Example: Linear Regression

- Then, the likelihood is

$$L(\beta; Z) = \prod_{i=1}^{n} p_\beta(y_i \mid x_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(\beta^\top x_i - y_i)^2}{2}}$$

- The NLL is

$$\ell(\beta; Z) = -\sum_{i=1}^{n} \log p_\beta(y_i \mid x_i) = \underbrace{\frac{n \log(2\pi)}{2}}_{\text{constant}} + \underbrace{\frac{1}{2}\sum_{i=1}^{n} (\beta^\top x_i - y_i)^2}_{\text{MSE!}}$$

Example: Linear Regression

- Loss minimization for maximum likelihood estimation:

$$\hat{\beta}(Z) = \arg\min_{\beta} \ell(\beta; Z)$$

- **Note:** Called maximum likelihood estimation since maximizing the likelihood equivalent to minimizing the NLL

Example: Linear Regression

- What about the model family?

$$f_\beta(x) = \arg\max_y p_\beta(y \mid x)$$

$$= \arg\max_y \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(\beta^\top x - y)_2^2}{2}}$$

$$= \beta^\top x$$

- **Recovers linear functions!**

Loss Minimization View of ML

- **Three design decisions**
  - **Model family:** What are the candidate models f? (E.g., linear functions)
  - **Loss function:** How to define "approximating"? (E.g., MSE loss)
  - **Optimizer:** How do we optimize the loss? (E.g., gradient descent)

Maximum Likelihood View of ML

- Two **design decisions**
  - **Likelihood:** Probability $p_\beta(y \mid x)$ of data (x, y) given parameters β
  - **Optimizer:** How do we optimize the NLL? (E.g., gradient descent)

- **Corresponding Loss Minimization View:**
  - **Model family:** Most likely label $f_\beta(x) = \arg\max_y p_\beta(y \mid x)$
  - **Loss function:** Negative log likelihood (NLL) $\ell(\beta; Z) = -\sum_{i=1}^{n} \log p_\beta(y_i \mid x_i)$

- Very powerful framework for designing cutting edge ML algorithms
  - Write down the "right" likelihood, form tractable approximation if needed
  - Especially useful for thinking about non-i.i.d. data

What about classification?

Compare to linear regression:

$$p_\beta(y \mid x_i) \propto e^{-\frac{(\beta^\top x_i - y)^2}{2}}$$

- Consider the following choice:

$$p_\beta(Y = 0 \mid x_i) \propto e^{-\frac{\beta^\top x_i}{2}} \text{ and } p_\beta(Y = 1 \mid x_i) \propto e^{\frac{\beta^\top x_i}{2}}$$

- Then, we have

Sigmoid function

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

$$p_\beta(Y = 1 \mid x_i) = \frac{e^{\frac{\beta^\top x_i}{2}}}{e^{\frac{\beta^\top x_i}{2}} + e^{-\frac{\beta^\top x_i}{2}}} = \frac{1}{1 + e^{-\beta^\top x_i}}$$

What about classification?

Compare to linear regression:

$$p_\beta(y \mid x_i) \propto e^{-\frac{(\beta^\top x_i - y)^2}{2}}$$

- Consider the following choice:

$$p_\beta(Y = 0 \mid x_i) \propto e^{-\frac{\beta^\top x_i}{2}} \text{ and } p_\beta(Y = 1 \mid x_i) \propto e^{\frac{\beta^\top x_i}{2}}$$

- Then, we have

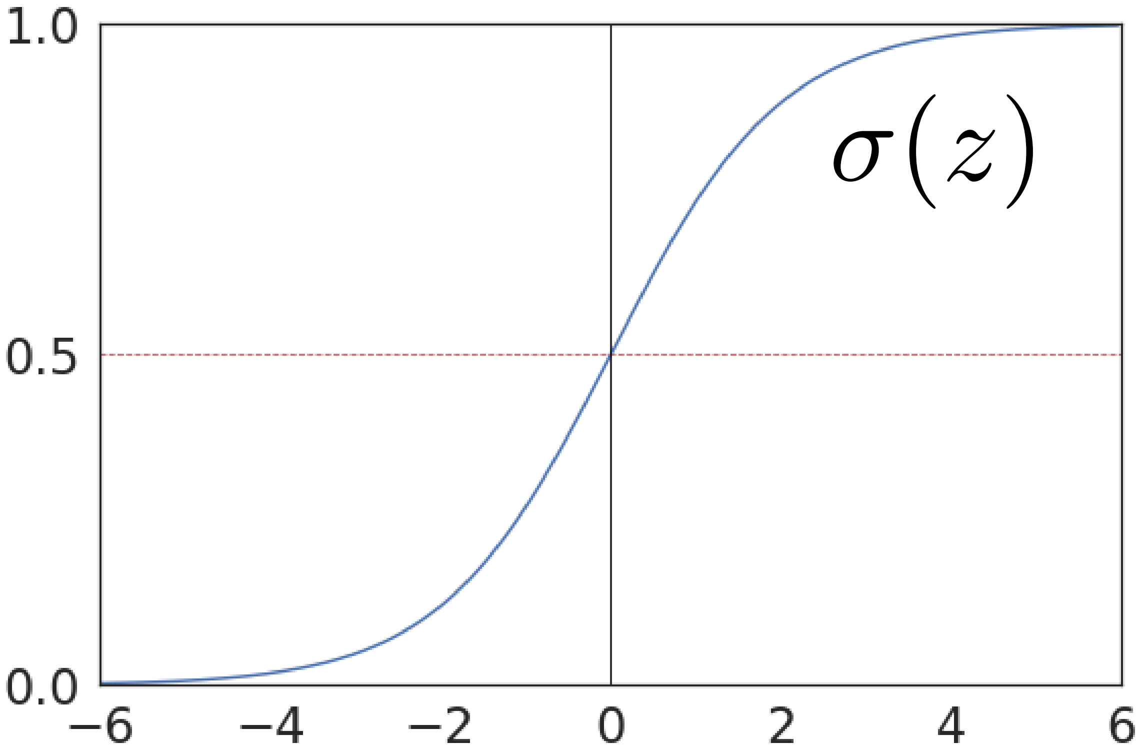
Sigmoid function

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

$$p_\beta(Y = 1 \mid x_i) = \frac{e^{\frac{\beta^\top x_i}{2}}}{e^{\frac{\beta^\top x_i}{2}} + e^{-\frac{\beta^\top x_i}{2}}} = \sigma(\beta^\top x_i)$$

- Furthermore, $p_\beta(Y = 0 \mid x_i) = 1 - \sigma(\beta^\top x_i)$

Logistic/Sigmoid Function



$$p_\beta(Y = 1 \mid x_i) = \sigma(\beta^\top x_i)$$

Logistic Regression Model Family

$$f_\beta(x) = \arg\max_y p_\beta(y \mid x)$$

$$= \arg\max_y \begin{cases} \sigma(\beta^\top x) & \text{if } y = 1 \\ 1 - \sigma(\beta^\top x) & \text{if } y = 0 \end{cases}$$

$$= \begin{cases} 1 & \text{if } \sigma(\beta^\top x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Logistic Regression Model Family

$$f_\beta(x) = \arg\max_y p_\beta(y \mid x)$$

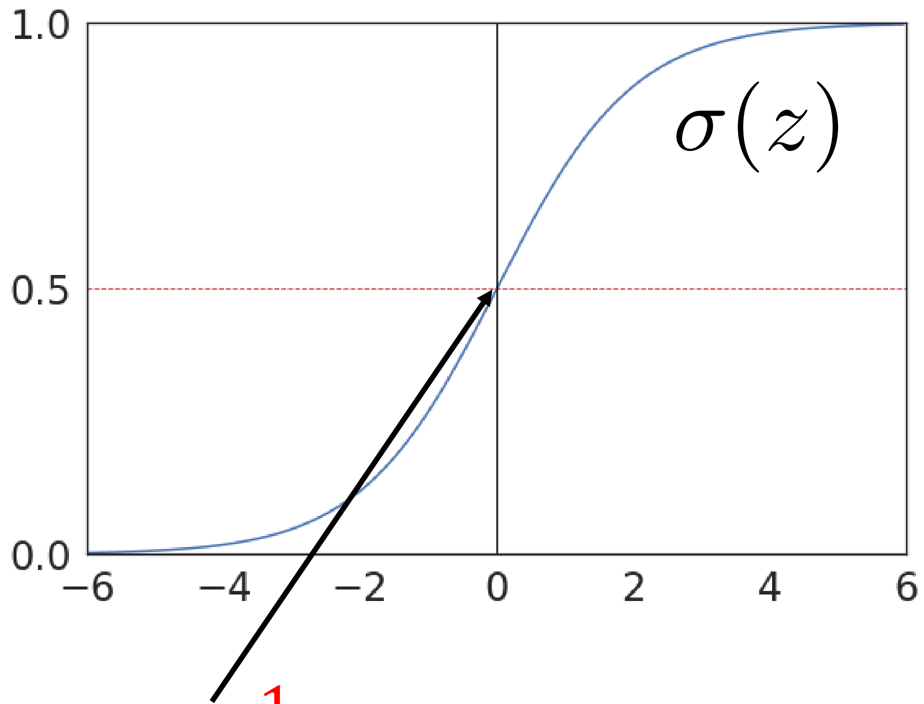
$$= \arg\max_y \begin{cases} \sigma(\beta^\top x) & \text{if } y = 1 \\ 1 - \sigma(\beta^\top x) & \text{if } y = 0 \end{cases}$$

$$= \begin{cases} 1 & \text{if } \sigma(\beta^\top x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} 1 & \text{if } \beta^\top x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$= 1(\beta^\top x \geq 0)$$

- **Recovers linear classifiers!**



$$\sigma(0) = \frac{1}{2}$$

Logistic Regression Algorithm

- Then, we have the following NLL loss:

$$\begin{aligned}\ell(\beta; Z) &= -\sum_{i=1}^{n} \log p_{\beta}(y_i \mid x_i) \\ &= -\sum_{i=1}^{n} 1(y_i = 1) \cdot \log\left(\sigma(\beta^{\top} x_i)\right) + 1(y_i = 0) \cdot \log\left(1 - \sigma(\beta^{\top} x_i)\right) \\ &= -\sum_{i=1}^{n} y_i \cdot \log\left(\sigma(\beta^{\top} x_i)\right) + (1 - y_i) \cdot \log\left(1 - \sigma(\beta^{\top} x_i)\right)\end{aligned}$$

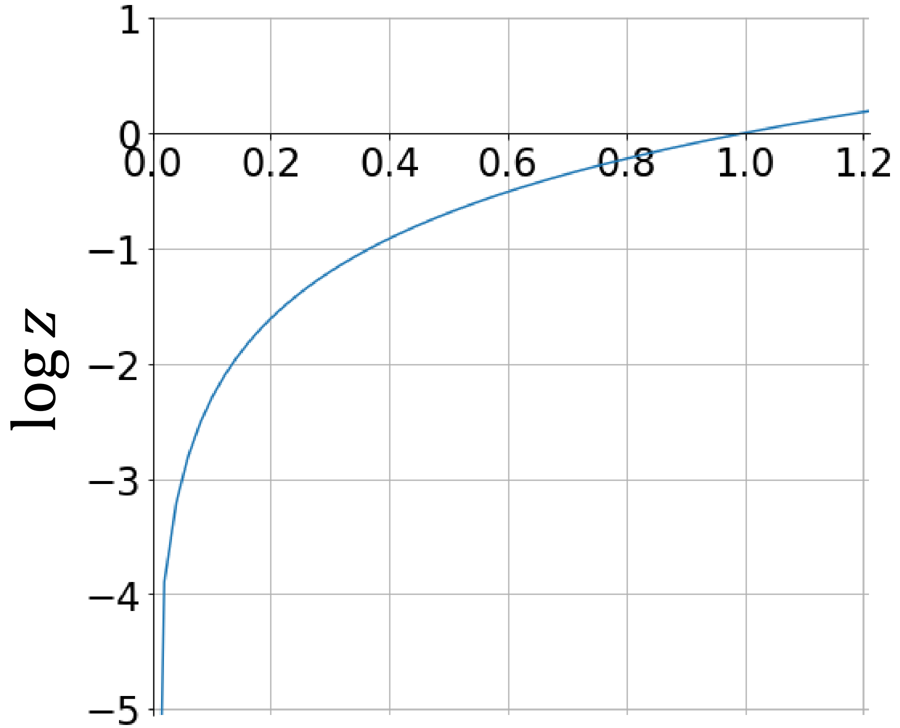
- Logistic regression minimizes this loss:

$$\hat{\beta}(Z) = \arg\min_{\beta} \ell(\beta; Z)$$

Intuition on the Objective

- Loss for example i is

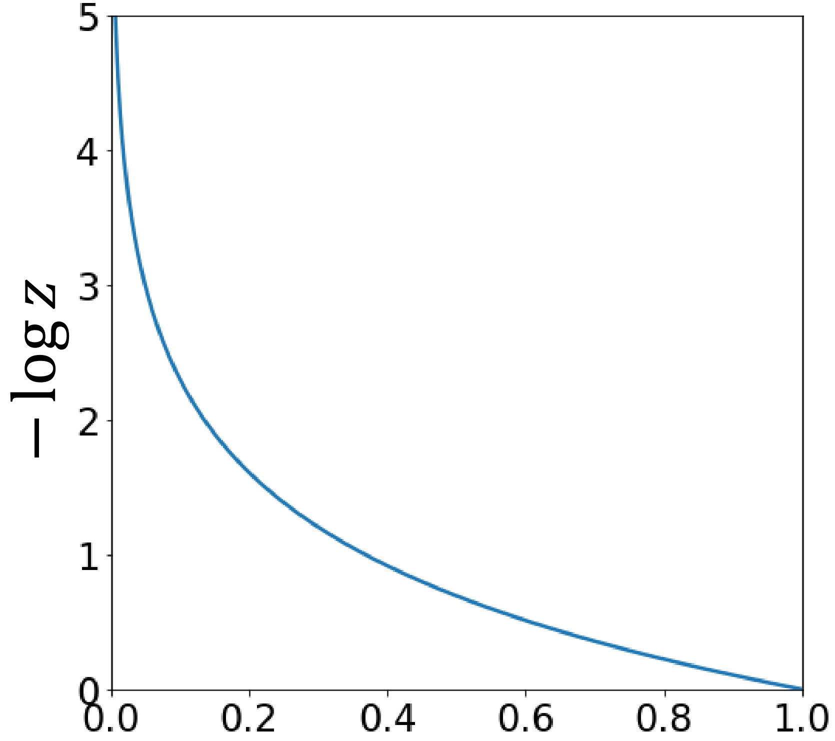
$$\begin{cases} -\log(\sigma(\beta^\top x_i)) & \text{if } y_i = 1 \\ -\log(1-\sigma(\beta^\top x_i)) & \text{if } y_i = 0 \end{cases}$$



Intuition on the Objective

- Loss for example i is

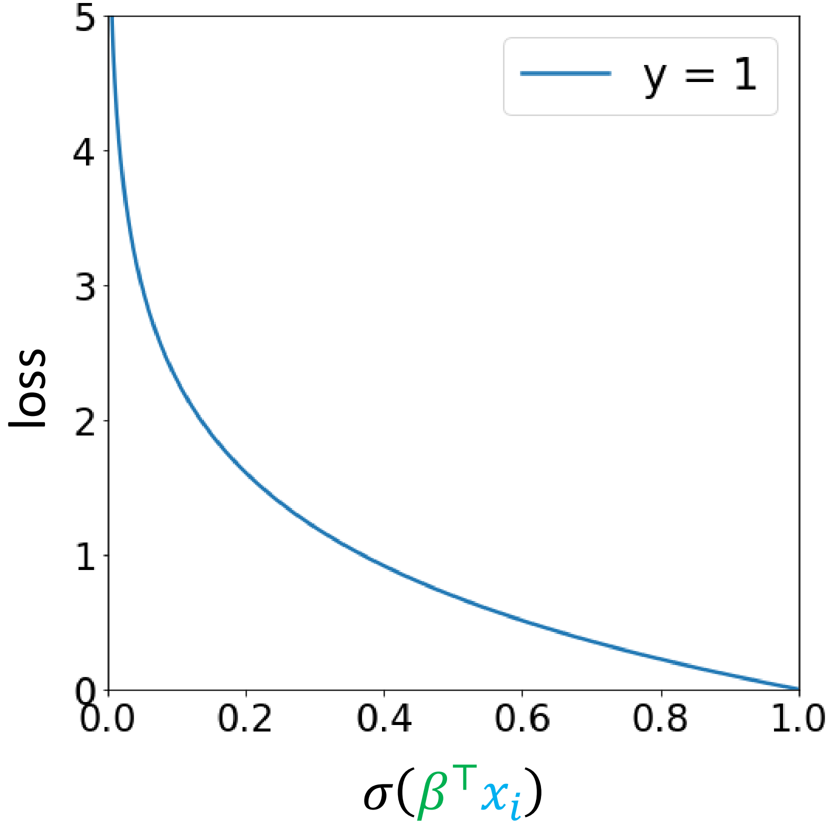
$$\begin{cases} -\log(\sigma(\beta^\top x_i)) & \text{if } y_i = 1 \\ -\log(1-\sigma(\beta^\top x_i)) & \text{if } y_i = 0 \end{cases}$$



Intuition on the Objective

- If $y_i = 1$:
  - If $\sigma(\beta^\top x_i) = 1$, then loss $= 0$
  - As $\sigma(\beta^\top x_i) \to 0$, loss $\to \infty$

- If $y_i = 0$
  - If $\sigma(\beta^\top x_i) = 0$, then loss $= 0$
  - As $\sigma(\beta^\top x_i) \to 1$, loss $\to \infty$

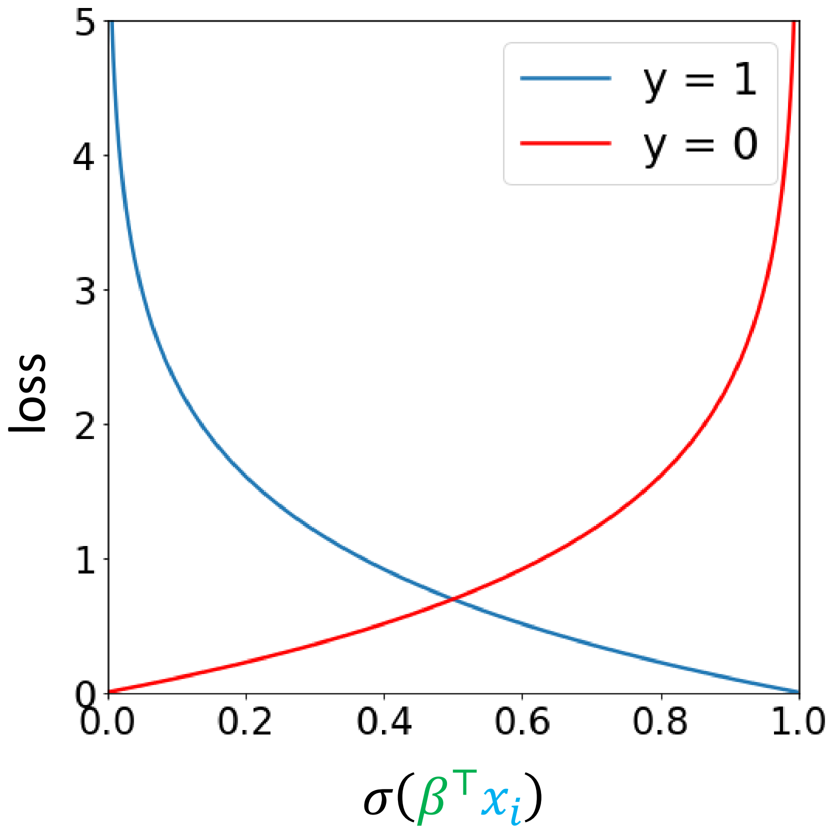


$$-y_i \cdot \boxed{\log\big(\sigma(\beta^\top x_i)\big)} - (1 - y_i) \cdot \log\big(1 - \sigma(\beta^\top x_i)\big)$$

Intuition on the Objective

- If $y_i = 1$:
  - If $\sigma(\beta^\top x_i) = 1$, then loss $= 0$
  - As $\sigma(\beta^\top x_i) \to 0$, loss $\to \infty$

- If $y_i = 0$
  - If $\sigma(\beta^\top x_i) = 0$, then loss $= 0$
  - As $\sigma(\beta^\top x_i) \to 1$, loss $\to \infty$



$$-y_i \cdot \boxed{\log\left(\sigma(\beta^\top x_i)\right)} - (1 - y_i) \cdot \boxed{\log\left(1 - \sigma(\beta^\top x_i)\right)}$$

Optimization for Logistic Regression

- To optimize the NLL loss, we need its gradient:

$$\nabla_\beta \ell(\beta; Z) = -\sum_{i=1}^n y_i \cdot \nabla_\beta \log\big(\sigma(\beta^\top x_i)\big) + (1 - y_i) \cdot \nabla_\beta \log\big(1 - \sigma(\beta^\top x_i)\big)$$

$$= -\sum_{i=1}^n y_i \cdot \frac{\nabla_\beta \sigma(\beta^\top x_i)}{\sigma(\beta^\top x_i)} - (1 - y_i) \cdot \frac{\nabla_\beta \sigma(\beta^\top x_i)}{1 - \sigma(\beta^\top x_i)}$$

$\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big)$ →

$$= -\sum_{i=1}^n y_i \cdot \frac{\sigma(\beta^\top x_i)\big(1 - \sigma(\beta^\top x_i)\big) \cdot x_i}{\sigma(\beta^\top x_i)} - (1 - y_i) \cdot \frac{\sigma(\beta^\top x_i)\big(1 - \sigma(\beta^\top x_i)\big) \cdot x_i}{1 - \sigma(\beta^\top x_i)}$$

$$= -\sum_{i=1}^n y_i \cdot \big(1 - \sigma(\beta^\top x_i)\big) \cdot x_i - (1 - y_i) \cdot \sigma(\beta^\top x_i) \cdot x_i$$

$$= -\sum_{i=1}^n \big(y_i - \sigma(\beta^\top x_i)\big) \cdot x_i$$

Optimization for Logistic Regression

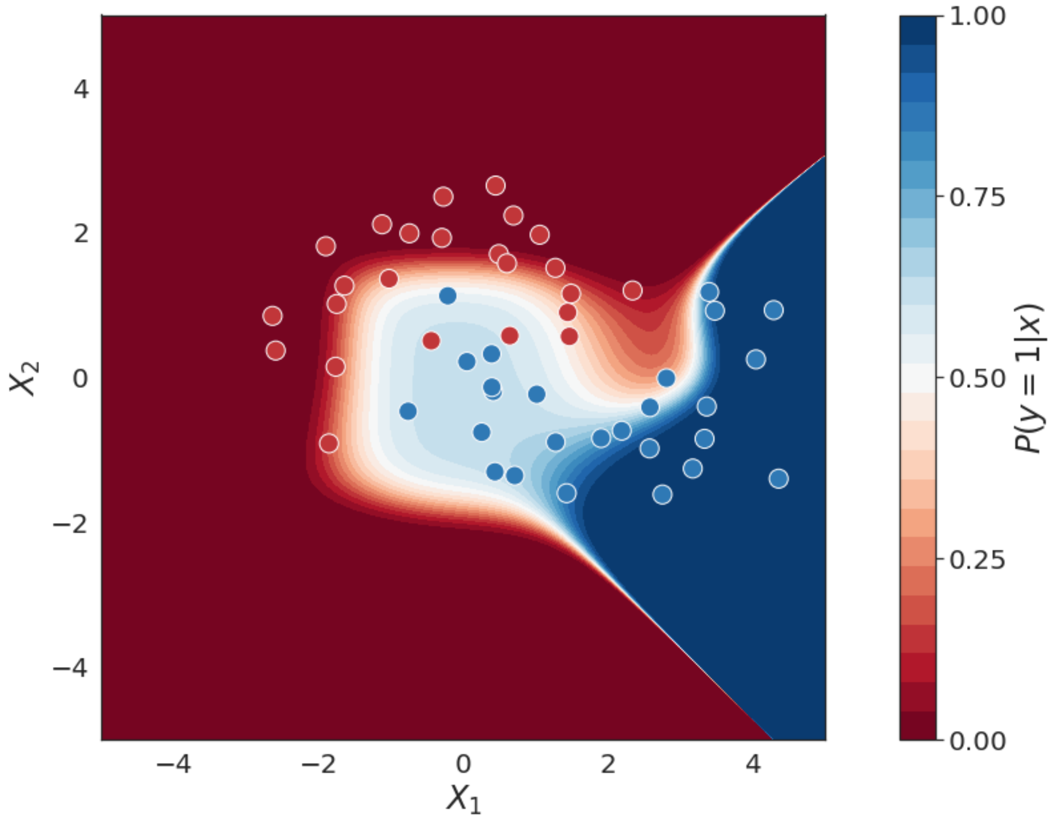
- Gradient of NLL:

$$\nabla_{\beta}\ell(\beta; Z) = \sum_{i=1}^{n} (\sigma(\beta^{\top} x_i) - y_i) \cdot x_i$$

- Surprisingly similar to the gradient for linear regression!
    - Only difference is the σ
- Gradient descent works as before
    - No closed-form solution for $\hat{\beta}(Z)$

Feature Maps

- Can use feature maps, just like linear regression



Regularized Logistic Regression

- We can add L_1 or L_2 regularization to the NLL loss, e.g.:

$$\ell(\beta; Z) = -\sum_{i=1}^{n} y_i \cdot \log\left(\sigma(\beta^\top x_i)\right) + (1 - y_i) \cdot \log\left(1 - \sigma(\beta^\top x_i)\right) + \lambda \cdot \|\beta\|_2^2$$

- Is there a more “natural” way to derive the regularized loss?

Regularization as a Prior

- So far, we have not assumed any distribution over the parameters β
  - What if we assume $\beta \sim N(0, \sigma^2 I)$ (the d dimensional normal distribution)?
  - (This σ is a hyperparameter, not the sigmoid function)

- Consider the modified likelihood

$$L(\beta; Z) = p_{Y,\beta \mid X}(Y, \beta \mid X)$$
$$= p_{Y \mid X,\beta}(Y \mid X, \beta) \cdot N(\beta; 0, \sigma^2 I)$$
$$= \left(\prod_{i=1}^{n} p_\beta(y_i \mid x_i)\right) \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|\beta\|_2^2}{2\sigma^2}}$$

Regularization as a Prior

- So far, we have not assumed any distribution over the parameters β
  - What if we assume $\beta \sim N(0, \sigma^2 I)$ (the d dimensional normal distribution)?

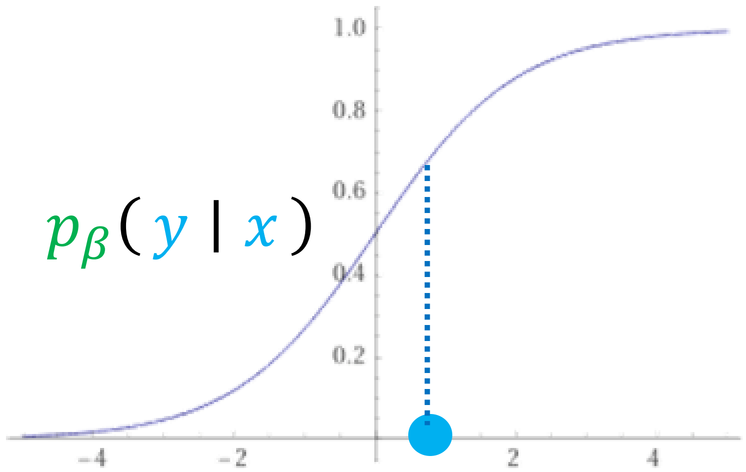
- Consider the modified NLL

$$\ell(\beta; Z) = -\sum_{i=1}^{n} \log p_\beta(y_i \mid x_i) + \underbrace{\log \sigma\sqrt{2\pi}}_{\text{constant}} + \underbrace{\frac{\|\beta\|_2^2}{2\sigma^2}}_{\text{regularization!}}$$

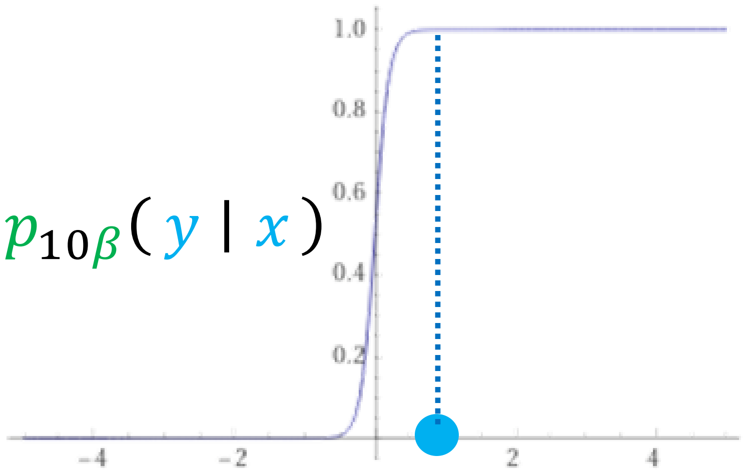
- Obtain L_2 regularization on β!
  - With $\lambda = \frac{1}{2\sigma^2}$
  - If $\beta_i \sim \text{Laplace}(0, \sigma^2)$ for each i, obtain L_1 regularization

Additional Role of Regularization

- In p_β, if we replace β with $c\beta$, where $c \gg 1$ (and $c \in \mathbb{R}$), then:
  - The decision boundary does not change
  - The probabilities $p_\beta(\, y \mid x \,)$ become more confident



$p_\beta(\, Y = 1 \mid x \,) \approx 0.6$



$p_{10\beta}(\, Y = 1 \mid x \,) \approx 1$

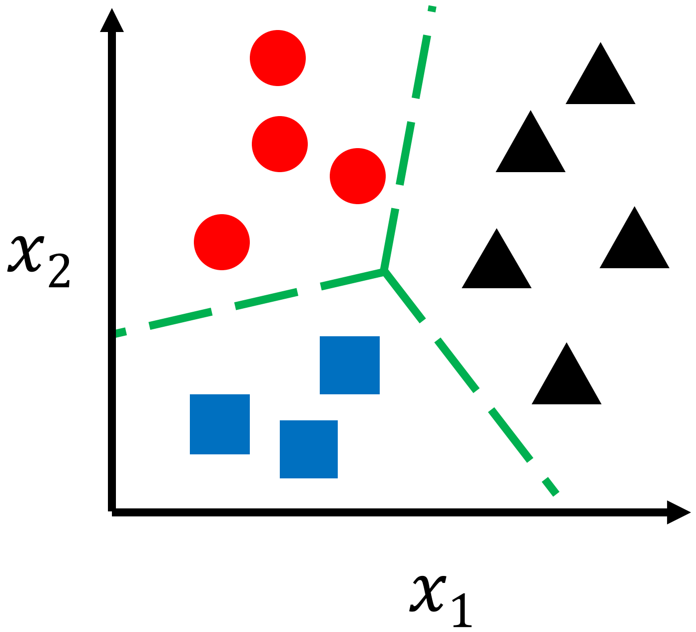
Additional Role of Regularization

- Regularization ensures that β does not become too large
  - Prevents overconfidence

- Regularization can also be **necessary**
  - Without regularization (i.e., $\lambda = 0$) and data is linearly separable, then gradient descent diverges (i.e., $\beta \to \pm\infty$)

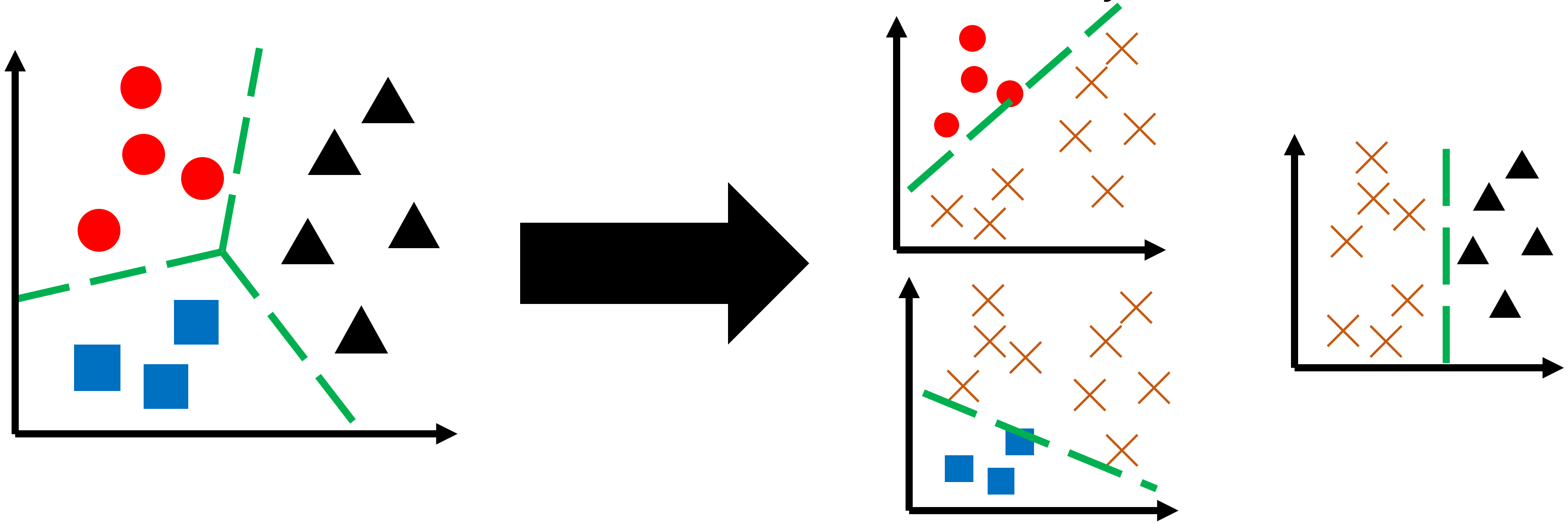
Multi-Class Classification

- What about more than two classes?
  - **Disease diagnosis:** healthy, cold, flu, pneumonia
  - **Object classification:** desk, chair, monitor, bookcase
  - In general, consider a finite space of labels $\mathcal{Y}$



Multi-Class Classification

- **Naïve Strategy:** One-vs-rest classification
  - **Step 1:** Train $|\mathcal{Y}|$ logistic regression models, where model $p_{\beta_y}(Y = 1 \mid x)$ is interpreted as the probability that the label for x is y
  - **Step 2:** Given a new input x, predict label $y = \arg\max_{y'} p_{\beta_{y'}}(Y = 1 \mid x)$



Multi-Class Logistic Regression

- **Strategy:** Include separate β_y for each label $y \in \mathcal{Y} = \{1, \dots, k\}$
- Let $p_\beta(y \mid x) \propto e^{\beta_y^\top x}$, i.e.

$$p_\beta(y \mid x) = \frac{e^{\beta_y^\top x}}{\sum_{y' \in \mathcal{Y}} e^{\beta_{y'}^\top x}}$$

- We define $\mathrm{softmax}(z_1, \dots, z_k) = \left[\frac{e^{z_1}}{\sum_{i=1}^k e^{z_i}} \quad \cdots \quad \frac{e^{z_k}}{\sum_{i=1}^k e^{z_i}}\right]$
- Then, $p_\beta(y \mid x) = \mathrm{softmax}\left(\beta_1^\top x, \dots, \beta_k^\top x\right)_y$
    - Thus, sometimes called **softmax regression**

Multi-Class Logistic Regression

- **Model family**
  - $f_\beta(x) = \arg\max_y p_\beta(y \mid x) = \arg\max_y \frac{e^{\beta_y^\top x}}{\sum_{y' \in \mathcal{Y}} e^{\beta_{y'}^\top x}} = \arg\max_y \beta_y^\top x$
- **Optimization**
  - Gradient descent on NLL
  - Simultaneously update all parameters $\{\beta_y\}_{y \in \mathcal{Y}}$

Classification Metrics

- While we minimize the NLL, we often evaluate using **accuracy**

- However, even accuracy isn't necessarily the "right" metric
  - If 99% of labels are negative (i.e., $y_i = 0$), accuracy of $f_\beta(x) = 0$ is 99%!
  - For instance, very few patients test positive for most diseases
  - "Imbalanced data"

- What are alternative metrics for these settings?

Classification Metrics

- **Classify test examples as follows:**
  - **True positive (TP):** Actually positive, predictive positive
  - **False negative (FN):** Actually positive, predicted negative
  - **True negative (TN):** Actually negative, predicted negative
  - **False positive (FP):** Actually negative, predicted positive

- Many metrics expressed in terms of these; for example:

$$\text{accuracy} = \frac{TP + TN}{n} \qquad \text{error} = 1 - \text{accuracy} = \frac{FP + FN}{n}$$

Confusion Matrix

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

Confusion Matrix

		Predicted Class	
		Yes	No
Actual Class	Yes	3 TP	4 FN
	No	6 FP	37 TN

Accuracy = 0.8

Classification Metrics

- For imbalanced metrics, we roughly want to disentangle:
    - Accuracy on “positive examples”
    - Accuracy on “negative examples”

- Different definitions are possible (and lead to different meanings)!

Sensitivity & Specificity

- **Sensitivity:** What fraction of **actual positives** are **predicted positive**?
  - **Good sensitivity:** If you have the disease, the test correctly detects it
  - Also called **true positive rate**
- **Specificity:** What fraction of **actual negatives** are **predicted negative**?
  - **Good specificity:** If you do not have the disease, the test says so
  - Also called **true negative rate**
- Commonly used in medicine

Sensitivity & Specificity

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

$$\text{sensitivity} = \frac{TP}{TP + FN}$$

$$\text{specificty} = \frac{TN}{TN + FP}$$

Sensitivity & Specificity

		Predicted Class	
		Yes	No
Actual Class	Yes	3 TP	4 FN
	No	6 FP	37 TN

$$\text{sensitivity} = \frac{TP}{TP + FN}$$

$$\text{specificity} = \frac{TN}{TN + FP}$$

Sensitivity & Specificity

		Predicted Class: Yes	Predicted Class: No	
Actual Class	Yes	3 TP	4 FN	$\text{sensitivity} = 3/7$
	No	6 FP	37 TN	$\text{specificity} = 37/43$

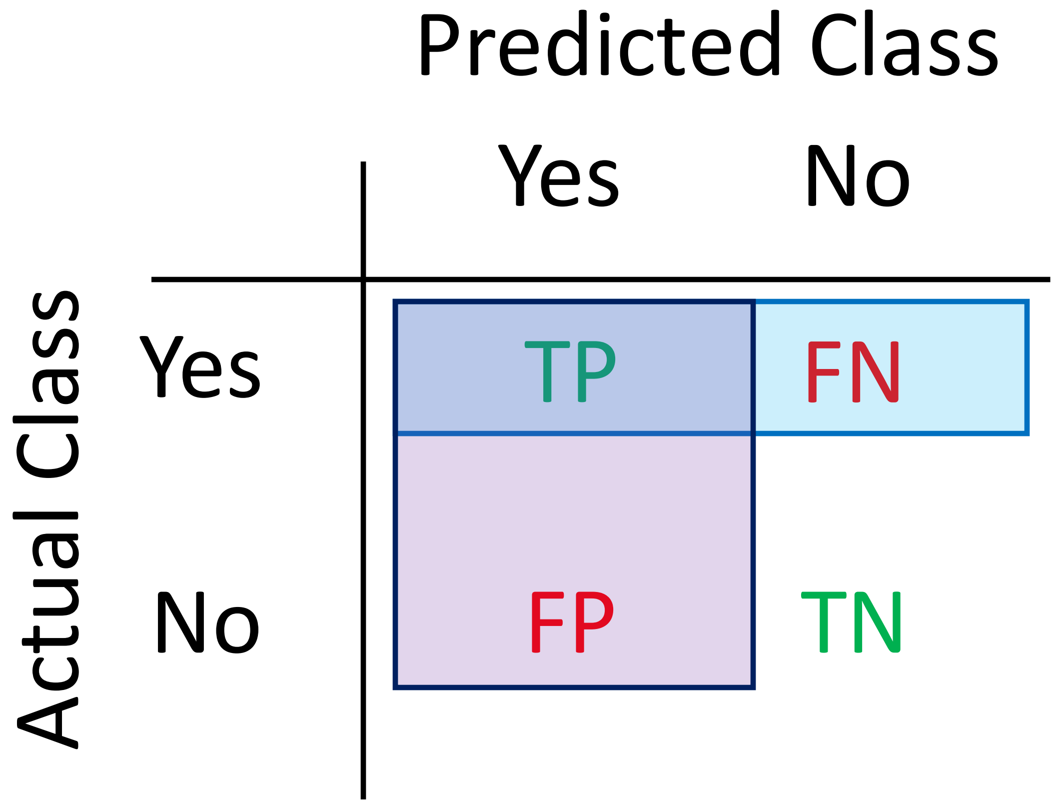
Precision & Recall

- **Recall:** What fraction of **actual positives** are **predicted positive**?
  - **Good recall:** If you have the disease, the test correctly detects it
  - Also called the **true positive rate** (and sensitivity)

- **Precision:** What fraction of **predicted positives** are **actual positives**?
  - **Good precision:** If the test says you have the disease, then you have it
  - Also called **positive predictive value**

- Used in information retrieval, NLP

Precision & Recall



$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

Precision & Recall

		Predicted Class	
		Yes	No
Actual Class	Yes	3 TP	4 FN
	No	6 FP	37 TN

$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

Precision & Recall

		Predicted Class	
		Yes	No
Actual Class	Yes	3 TP	4 FN
	No	6 FP	37 TN

$\text{recall} = 3/7$

$\text{precision} = 3/9$