Announcements

- Quiz 1 due this Thursday, testing mainly lin reg.
- HW0 done, grading in progress. Coding grades looked good.
- HW1 in progress.

- Coming soon:
  - Project format announcements
  - Mid-term preparation materials

CIS 4190/5190: Lec 05 Mon Sep 16, 2024

First few mins: Linear Regression Wrap-Up

Recap of Linear Regression

- Lec 01: lin reg model class, loss function, feature maps
  - Asides (broader than lin reg): performance evaluation & metrics, function approx. view of ML, model capacity & overfitting
- Lec 02: Regularized lin reg loss function
  - Asides (broader than lin reg): train and test error, bias and variance, the concept of regularization, hyperparameter tuning & validation data
- Lec 03: Ways to optimize lin reg loss function
  - Both involve measuring the gradient of loss w.r.t parameters.
  - Option 1: set analytical expression of gradient to 0, and solve. **Closed-form solution.**
  - Option 2: iteratively move in the direction of gradient. **Gradient descent.**
- **Today: wrap-up. Feature standardization and ℓ_1 regularization**

Indexing: from 0 or 1?

- Our slides mostly follow the math tradition of indexing from 1.

$$L(\beta;Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda\sum_{j=1}^{d}\beta_j^2$$

$$L(\beta;Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda\sum_{j=1}^{d}|\beta_j|$$

- With polynomial functions, our index starts from 0, so that β_j is the coefficients for the j-th order term:

$$f_{\boldsymbol{\beta}}(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \cdots \beta_{10} x^{10}$$

Indexing: from 0 or 1?

- With intercept term ($\phi(x) = [1 \quad x_1 \quad \cdots \quad x_d]^\top$), no penalty on the weight for the intercept term (which is β_1 here):

- $L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \sum_{j=2}^{d} \beta_j^2$

Features in Linear Regression

- Feature Standardization

- Automatic Feature Selection with L1 Regularization

Feature Standardization

- Unregularized linear regression is invariant to feature scaling
  - Suppose we scale $x_{ij} \leftarrow 2x_{ij}$ for all examples x_i
  - Without regularization, simply use $\beta_j \leftarrow \beta_j/2$ to obtain equivalent solution
  - In particular, $\frac{\beta_j}{2} \cdot 2x_{ij} = \beta_j \cdot x_{ij}$

- Not true for regularized regression!
  - Penalty $\left(\beta_j/2\right)^2$ is scaled by 1/4 (not cancelled out!)

- $L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda\left(\beta_2^2 + \cdots + \beta_j^2 + \cdots + \beta_d^2\right)$
- $L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \frac{\beta}{2}^\top 2x_i\right)^2 + \lambda\left(\frac{\beta_2^2}{4} + \cdots + \frac{\beta_j^2}{4} + \cdots + \frac{\beta_d^2}{4}\right)$

Feature Standardization

- Rescale features to zero mean and unit variance

- $x_{i,j} \leftarrow \frac{x_{i,j}-\mu_j}{\sigma_j}$ $\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$ $\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x_{i,j} - \mu_j\right)^2$
  - Note: When using intercept term, do not rescale $x_1 = 1$
  - Can be sensitive to outliers (fix by dropping outliers)
- Makes it easier to estimate coefficients
- Often better encodes real variations in data

- **Common Rookie Error: Must use same transformation during training & prediction**
  - Please always compute μ_j and σ_j on training data, and use the same values when standardizing test data

Automatic Feature Set Selection with L1 Regularization

$\boldsymbol{L_0}$ Regularization → $\boldsymbol{L_1}$ Regularization

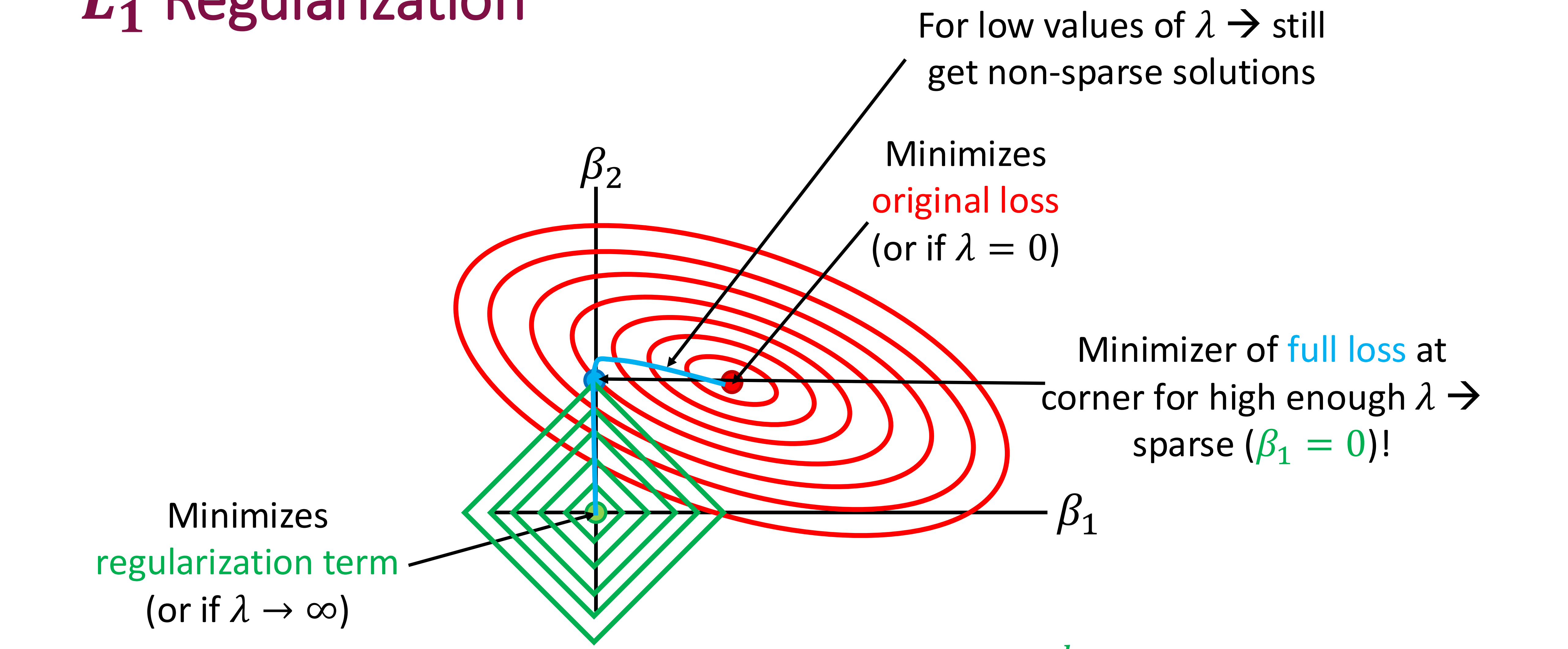
- Sparsity: Can we minimize $||\beta||_0 = \left|\{\, j \mid \beta_j \neq 0 \,\}\right|$, the number of non-zero components? (This is called $\boldsymbol{L_0}$ regularization)
  - Automatic feature selection!
  - Improves interpretability.

$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda||\beta||_0$$

- Challenge: $||\beta||_0$ is not differentiable, making it hard to optimize

- Solution: $\boldsymbol{L_1}$ Regularization
  - We can instead use an L_1 norm $||\beta||_1$ as the regularizer!
  - Still harder to optimize than L_2 norm, but at least it is convex

L_1 Regularization



$$L(\beta; Z) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta^\top x_i)^2 + \lambda \sum_{j=1}^{d}|\beta_j|$$

L_1 Regularization for Feature Selection

- Step 1: Construct a lot of features and add to feature map

- Step 2: Use L_1 regularized regression to "select" subset of features
  - I.e., coefficient $\beta_j \neq 0$ → feature j is selected)

- Optional: Remove unselected features from the feature map and run vanilla linear regression (a.k.a. ordinary least squares)

Optimizing L_1 Regularized Linear Regression?

- Gradient descent still works!

- Specialized algorithms work better in practice
  - Simple one: Gradient descent + soft thresholding
  - Basically, if $|\beta_{t,j}| \leq \lambda$, just set it to zero
  - Good theoretical properties

What About Classification Problems?

CIS 4190/5190: Lec 05 Mon Sep 16, 2024

Logistic Regression: Linear Models for Classification

Robot Image Credit: Viktoriya Sukhanova © 123RF.com

Recall: Supervised Learning

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

PS: sometimes denoted $\mathcal{D}$

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

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

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

Recall: Regression

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 **real value** $y_i \in \mathbb{R}$

Recall: Classification

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

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

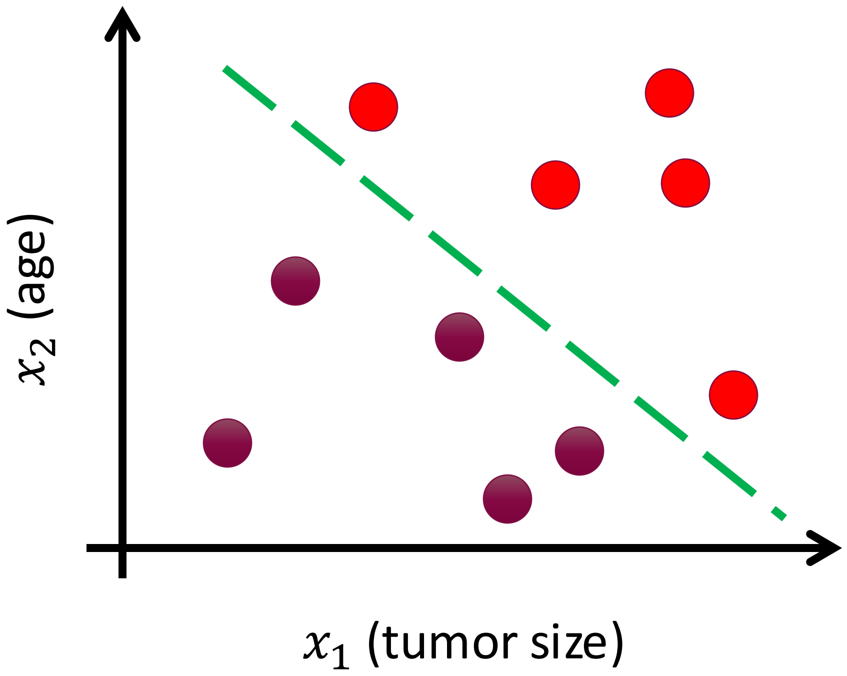
L encodes $y_i \approx f_\beta(x_i)$

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

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

(Binary) Classification

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



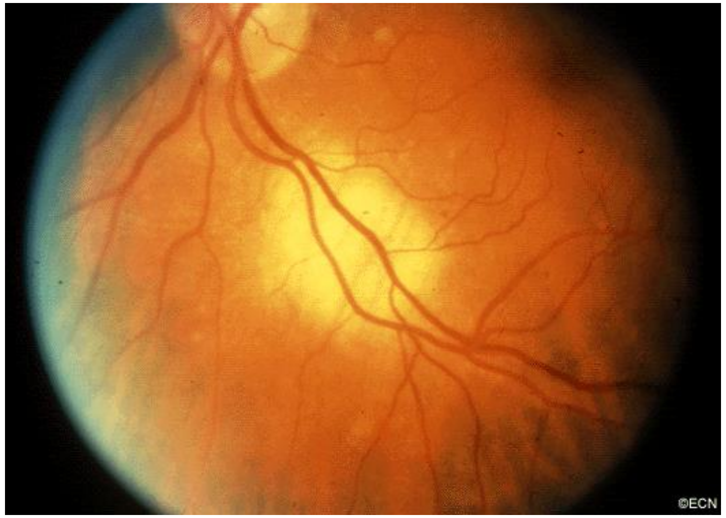


Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/

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?

Trying to Come up With A Model Class For Logistic Regression

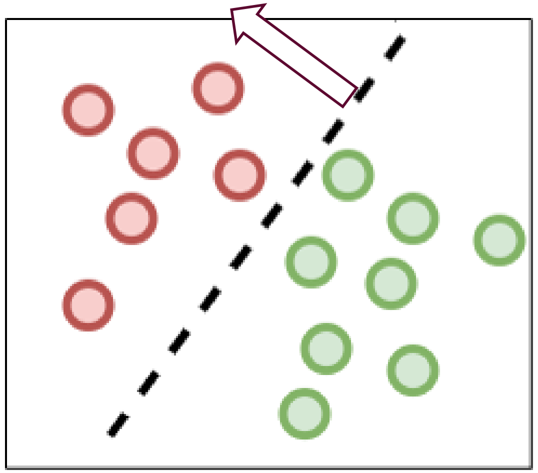
Repurpose Linear Regression For Classification?

Given $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_N}, y_N)\}$ where $\boldsymbol{x_i} \in \mathbb{R}^D, y_i \in \{0, 1\}$

Predict $y_i = \boldsymbol{\beta}^T \boldsymbol{x}_i$

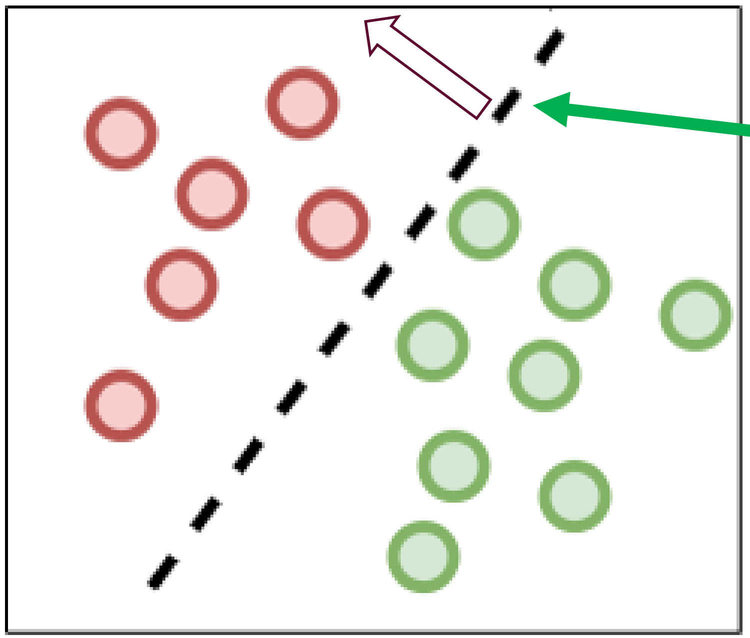
Predict y_i = class 1 if $\boldsymbol{\beta}^T \boldsymbol{x}_i \geq 0$

Predict y_i = class 0 if $\boldsymbol{\beta}^T \boldsymbol{x}_i < 0$



Repurpose Linear Regression For Classification?

Predict y_i = class 1 if $\boldsymbol{\beta}^T \boldsymbol{x}_i \geq 0$
Predict y_i = class 0 if $\boldsymbol{\beta}^T \boldsymbol{x}_i < 0$



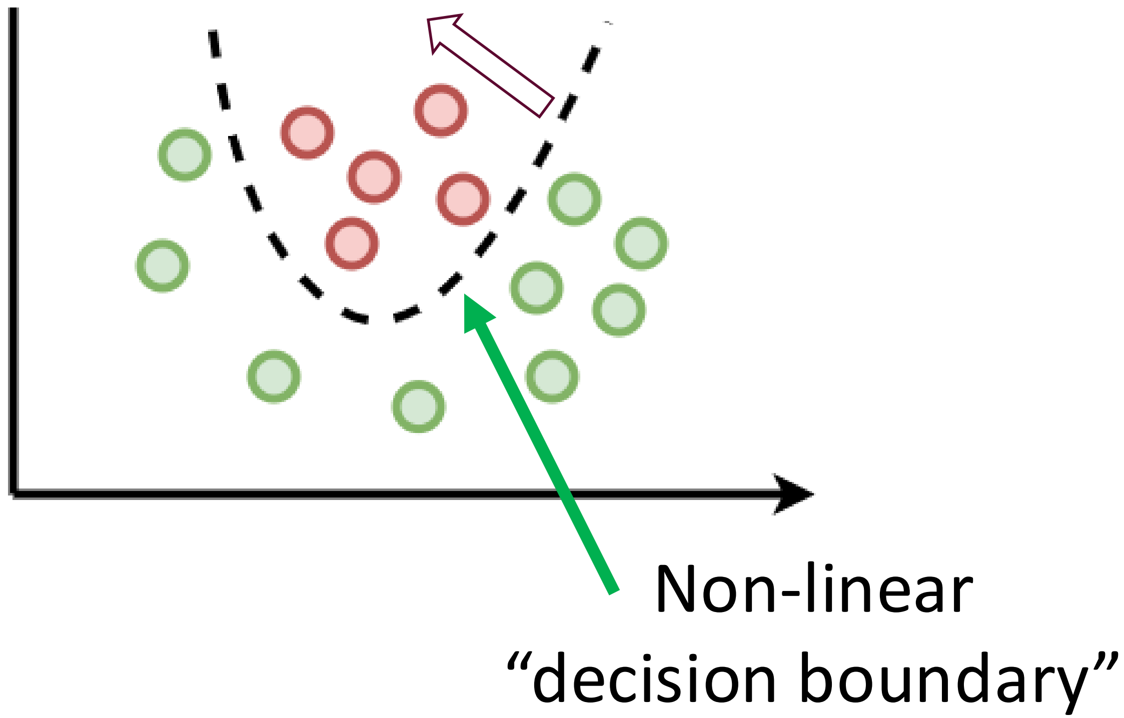
Linear
"decision boundary"

What if the data requires a non-linear decision boundary?

Non-Linear Decision Boundaries Thru Feature Expansion

Can apply basis expansion to features, same as with linear regression

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ \vdots \end{bmatrix}$$



Looks like we have a reasonable model class to start from …

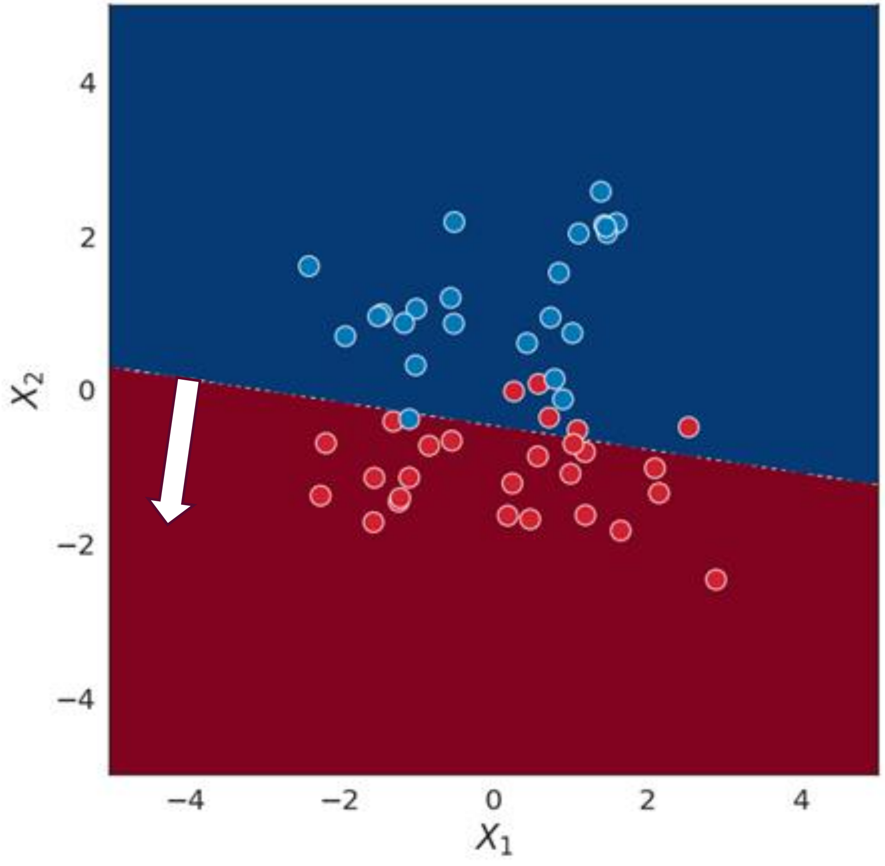
Can We Come Up With A Loss Function?

Loss Function

- **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!**

Any ideas?

Training dataset overlay on decision boundary

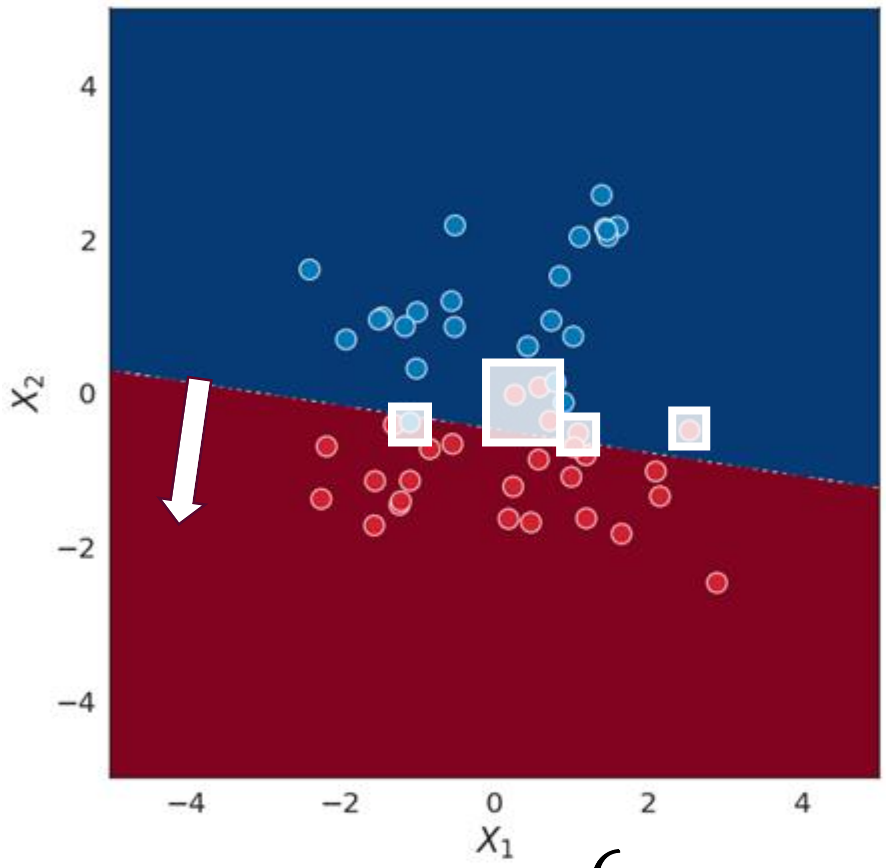


Candidate Classification Loss Function: Inaccuracy

- **(In)accuracy / Error Rate:**

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

- Indeed often captures what we care about in terms of classifier performance. Good *performance metric*.
- But bad loss function, because **computationally intractable to optimize**
  - Discontinuous measures are often hard to optimize. As an example, think about gradient descent …
- Need to “soften” this in some way … make more continuous



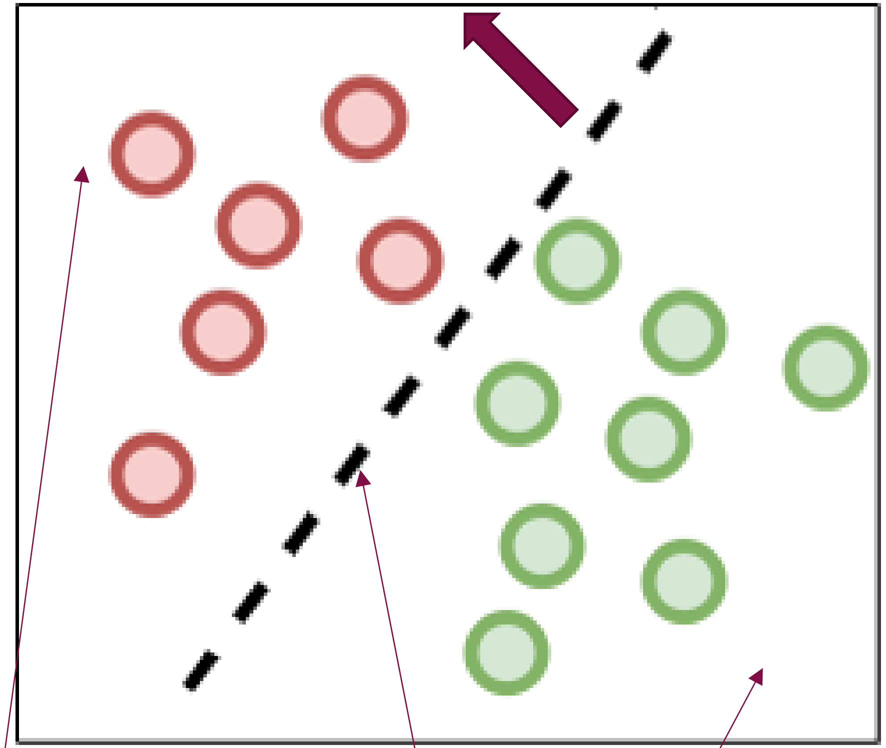
$$L(\beta; Z) = \frac{6}{50}$$

Revisiting the Model Class

Making Soft Decisions: Revisiting the Model Class

Predict y_i = class 1 if $\beta^\top x_i \geq 0$
Predict y_i = class 0 if $\beta^\top x_i < 0$

Predict $\mathrm{p}(y_i = 1|x_i, \beta)$ based on the value of $\beta^\top x_i$



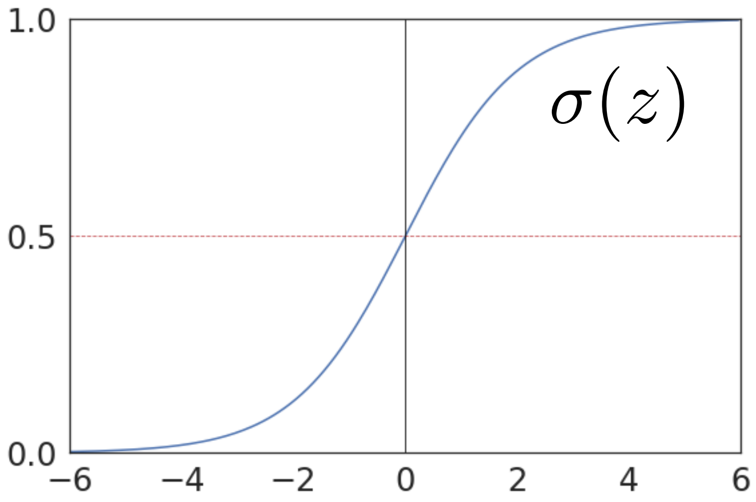
Intuition:

if $\beta^\top x_i$ has large positive value, then high $\mathrm{p}(y_i = 1|x_i, \beta) \rightarrow 1$
large negative value, then low $\mathrm{p}(y_i = 1|x_i, \beta) \rightarrow 0$
zero, then $\mathrm{p}(y_i = 1|x_i, \beta) \approx 0.5$

Logistic Regression

How to convert from $\beta^\top x_i$ which lies in $(-\infty, \infty)$ to a meaningful probability?

Logistic / Sigmoid Function



Logistic regression model:

$$p(y = 1 \mid \boldsymbol{x} \,; \beta) = \sigma(\beta^\top x),$$

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

$$p(\, y = 0 \mid \boldsymbol{x} \,; \beta) = 1 - p(\, y = 1 \mid \boldsymbol{x} \,; \beta)$$

Provides a score for each possible outcome $y = 0$ or $y = 1$

Example: Interpretation of Hypothesis Output

Example: Ocular tumor diagnosis from size

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ tumorSize \end{bmatrix}$$

$$p(y = 1 \mid \boldsymbol{x}\,; \beta) = \sigma(\beta^\top x),$$

→ Tumor has a 85% chance of being class 1: malignant

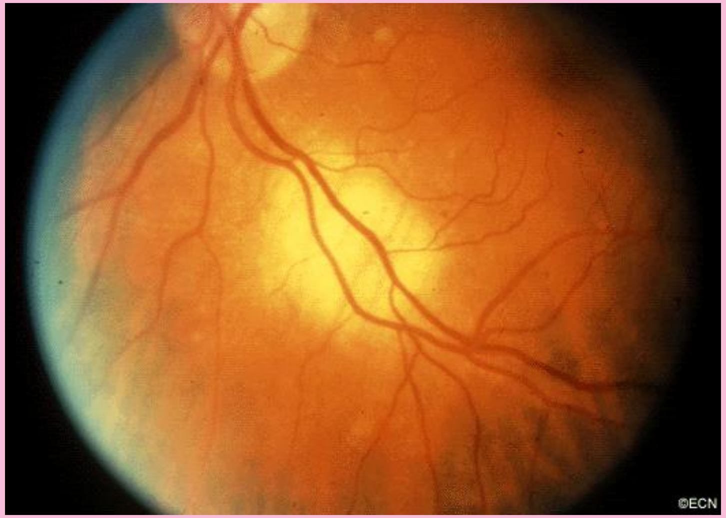


Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/

Decision Boundary?

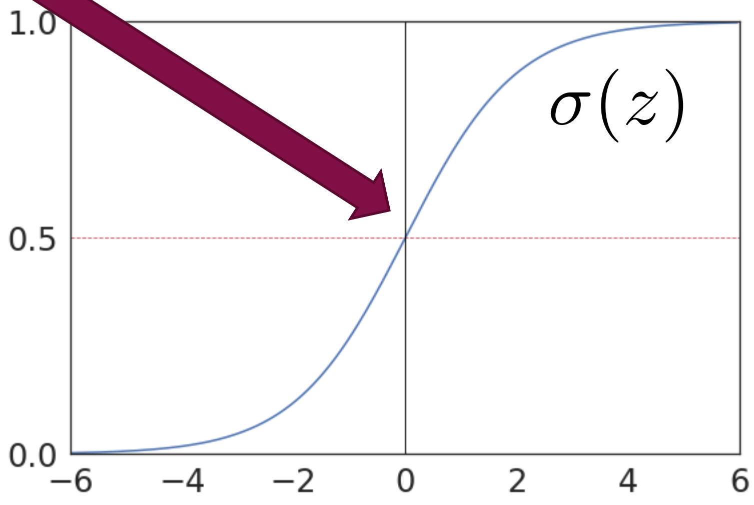
Exercise: What happens to $\boldsymbol{x}$ at infinite +/- distance from boundary?

$$p(y = 1 \mid \boldsymbol{x}\,; \beta) = \sigma(\beta^T \boldsymbol{x}) = 0.5$$

So, decision boundary is at:

$$\beta^T \boldsymbol{x} = 0$$

Logistic / Sigmoid Function



Consistent with:

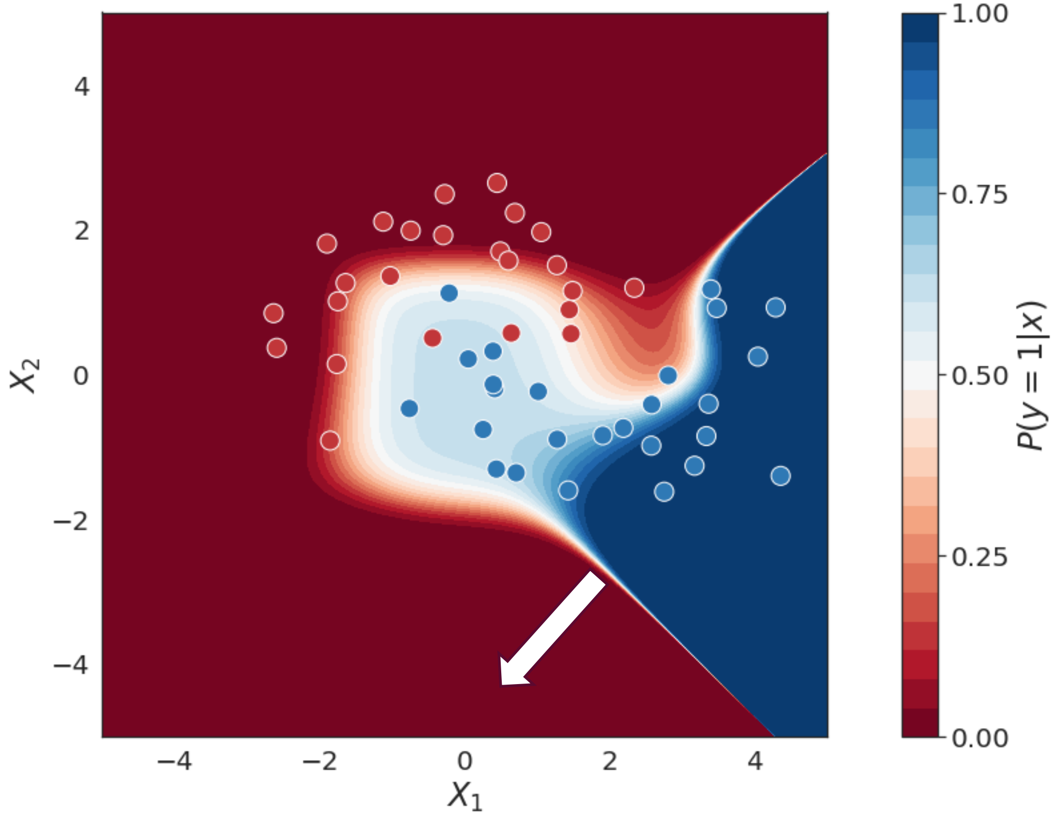
Predict y_i = class 1 if $\beta^T \boldsymbol{x}_i \geq 0$
Predict y_i = class 0 if $\beta^T \boldsymbol{x}_i < 0$

We now have a model class that can predict meaningful binary class probabilities!

Soft *Non-Linear* Decision Boundaries

Same feature expansion trick still works.

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ \vdots \end{bmatrix}$$



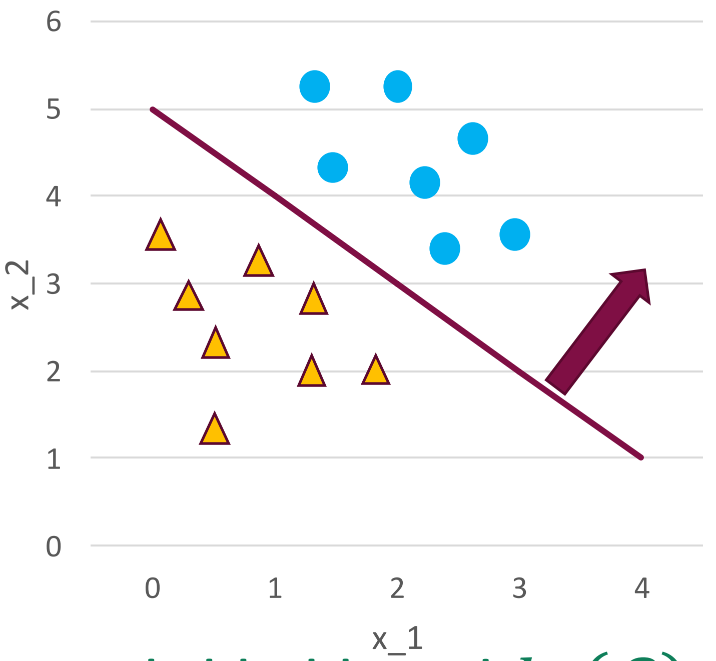
And Now, A Probability-Based Loss Function for Classification

“Likelihood” of Data Under a Model

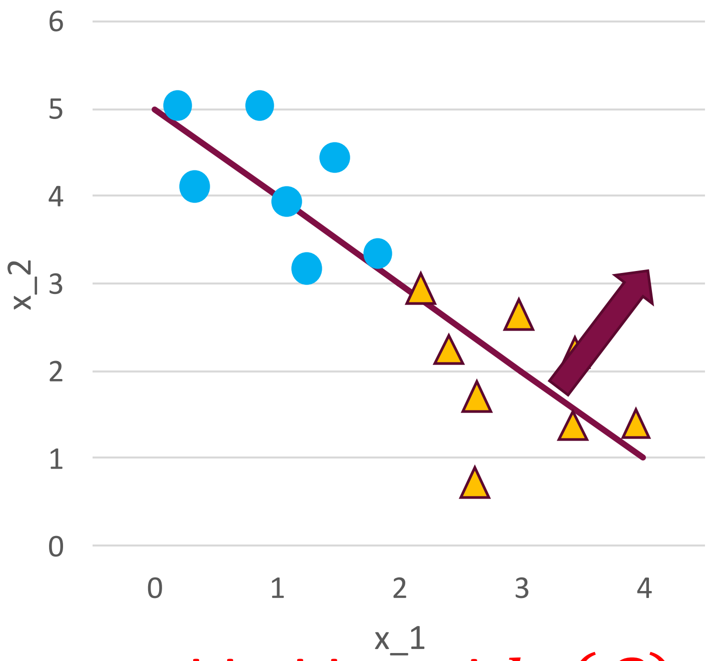
“Likelihood” $l_{\mathcal{D}}(\boldsymbol{\beta})$ of data $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ under some probabilistic model with parameters $\boldsymbol{\beta}$ describes, loosely:

<u>“if this model assigned labels to each $\boldsymbol{x}_i$ in the data, what is the probability that it would assign exactly the true labels y_i in $\mathcal{D}$”?</u>

Which of these datasets has high likelihood under this model?



High likelihood $l_{\mathcal{D}}(\boldsymbol{\beta})$

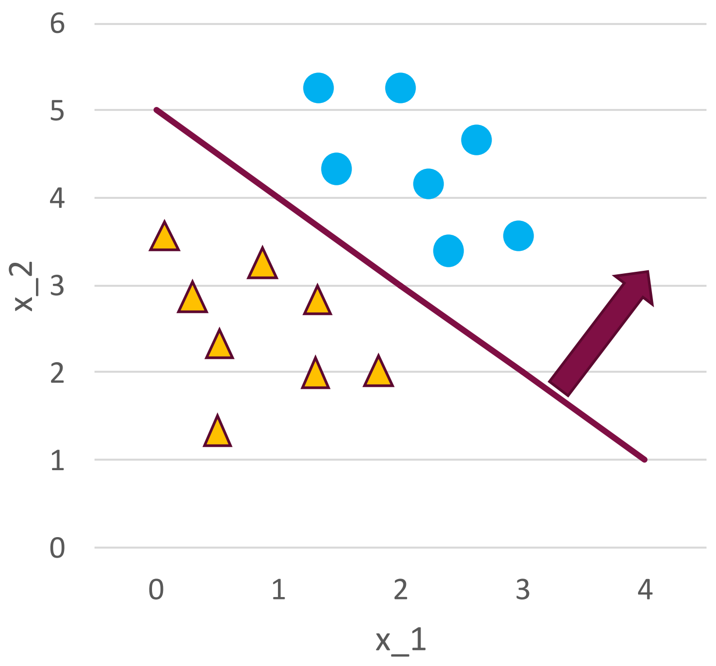


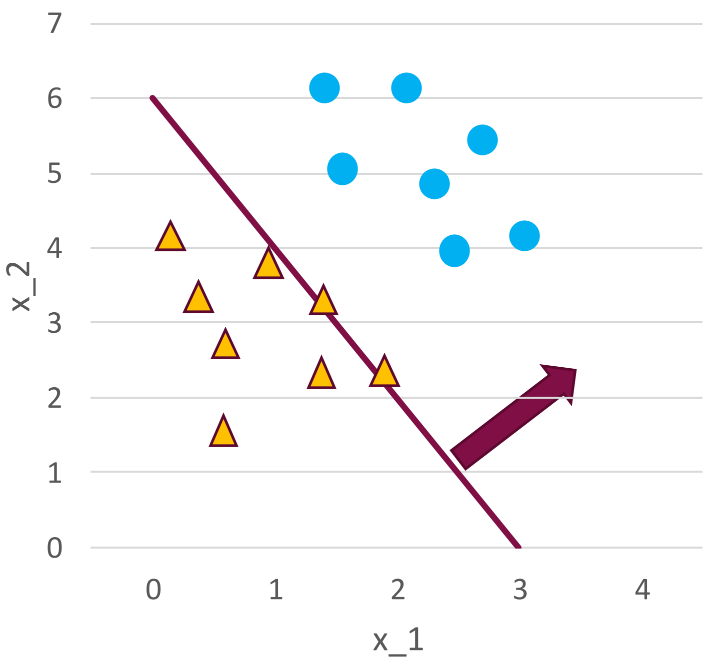
Low likelihood $l_{\mathcal{D}}(\boldsymbol{\beta})$

"Likelihood" of Data Under a Model

- In practice, the dataset is fixed, and we are looking to find good models.

Which of these models does the data have high likelihood under?





High likelihood $l_{\mathcal{D}}(\boldsymbol{\beta})$

Low likelihood $l_{\mathcal{D}}(\boldsymbol{\beta})$

"Likelihood" of Data Under a LogReg Model

"Likelihood" $l_{\mathcal{D}}(\boldsymbol{\beta})$:"What is the probability that the model with parameters β would assign labels y_i to the samples $\boldsymbol{x}_i$ for all $(\boldsymbol{x}_i, y_i)_{i=1}^{N}$ in the dataset $\mathcal{D}$?"

For a single sample dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1)\}$, this would be:

$$p(y_1|\boldsymbol{x}_1; \boldsymbol{\beta}) = p(y = y_1 \mid \boldsymbol{x} = \boldsymbol{x}_1 \,; \boldsymbol{\theta}) = \begin{cases} \sigma(\boldsymbol{\beta}^T \boldsymbol{x}_1) & \text{if } y_1 = 1 \\ 1 - \sigma(\boldsymbol{\beta}^T \boldsymbol{x}_1) & \text{if } y_1 = 0 \end{cases}$$

For a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ with N samples:

$$l_{\mathcal{D}}(\boldsymbol{\beta}) = \prod_{i=1}^{N} p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta}\,)$$

Because independent assignment of y_is to x_is
Recall: joint probability of two "*independent*" events = product of their probabilities.

"Maximum Likelihood Estimation"

"Likelihood" of a dataset $\mathcal{D}$ with N samples under model with parameters $\boldsymbol{\beta}$:

$$l_{\mathcal{D}}(\boldsymbol{\beta}) = \prod_{i=1}^{N} p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta})$$

We are looking for the $\boldsymbol{\beta}$ that maximizes the likelihood of the training data, so the optimal $\boldsymbol{\beta}$ is the "maximum likelihood estimate" (MLE):

$$\boldsymbol{\beta}_{MLE} = \arg\max_{\boldsymbol{\beta}} l_{\mathcal{D}}(\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{N} p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta})$$

Note: Since each probability is in [0,1], this product is a very small number.
What happens if you multiply 0.1 by itself 10,000 times in a computer? Bad things!

"Log Likelihood" Objective

Need to solve $\boldsymbol{\beta}_{MLE} = \arg\max_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{N} p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta})$

Since the logarithm is always higher for higher numbers, we can take the log without changing the optimal $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{MLE} = \arg\max_{\boldsymbol{\beta}} l(\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\beta}} \prod_{i=1}^{N} p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta})$$

$$= \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{N} \log p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta}\,)$$

This is called the log likelihood

Sum avoids underflow

"Negative Log-Likelihood Loss"

$$h_\beta(x) = p(y = 1|\boldsymbol{x}; \boldsymbol{\beta}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \boldsymbol{x}}}$$

$$\boldsymbol{\beta}_{MLE} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{N} \log p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta})$$

$$= \arg\min_{\boldsymbol{\beta}} - \sum_{i=1}^{N} \boxed{\log p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta}\,)}$$

Taking the negative turns a maximization problem into a minimization problem

$$\boxed{= \begin{cases} \log h_{\boldsymbol{\beta}}(\boldsymbol{x}_i) & \text{if } y_i = 1 \\ \log(1 - h_{\boldsymbol{\beta}}(\boldsymbol{x}_i)) & \text{if } y_i = 0 \end{cases}}$$

Just to avoid writing this expression on two lines, let's write this as:

$$\log p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta}\,) = [y_i \log h_{\boldsymbol{\beta}}(\boldsymbol{x}_i) + (1 - y_i) \log(1 - h_{\boldsymbol{\beta}}(\boldsymbol{x}_i))]$$

Summing up the Logistic Regression Loss Function

$$\boldsymbol{\beta}_{MLE} = \arg\min_{\boldsymbol{\beta}} -\sum_{i=1}^{N} \log p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta} \,)$$

$$\log p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta} \,) = [y_i \log h_{\boldsymbol{\beta}}(\boldsymbol{x}_i) + (1 - y_i) \log(1 - h_{\boldsymbol{\beta}}(\boldsymbol{x}_i))]$$

Logistic regression maximum likelihood loss function:

$$\min_{\theta} -\sum_{i=1}^{N} [\, y_i \log h_{\boldsymbol{\beta}}(\boldsymbol{x}_i) + (1 \, - y_i) \log(1 \, - h_{\boldsymbol{\beta}}(\boldsymbol{x}_i)) \,]$$

Thought Exercise: Maximum Likelihood More Broadly

- Maximum likelihood estimation is a general framework for thinking about objective function design for ML problems.
- In fact, the linear regression objective (MSE) can also be viewed as the negative log-likelihood of the training dataset under the model.

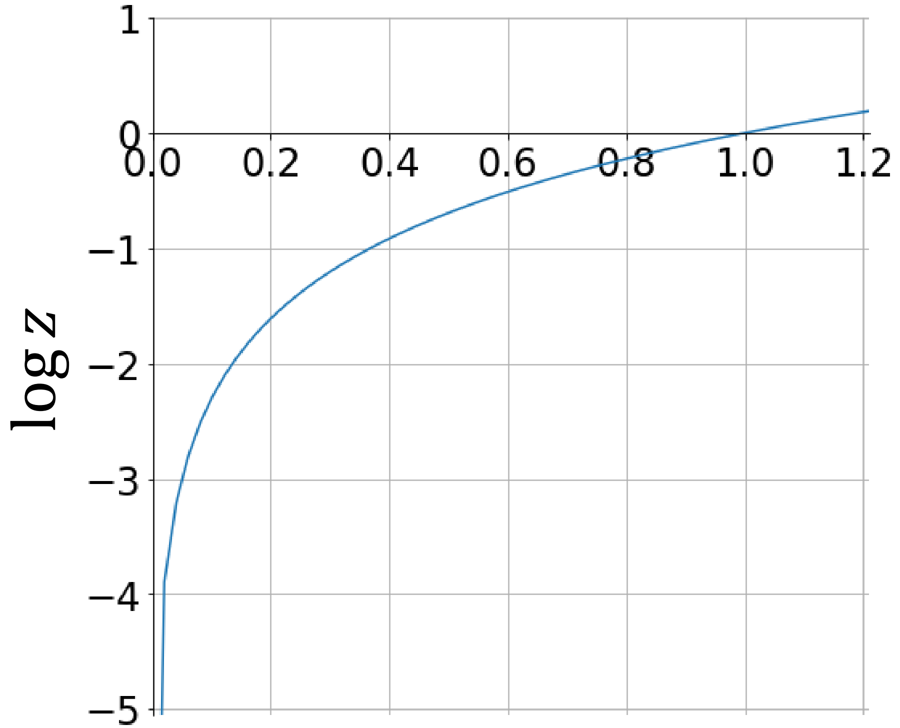
- Try to think through how: In particular, what form should $p(y|x;\beta) = \beta^T x$ take so that log likelihood$(\beta) \approx -\frac{1}{N}\sum_{\mathcal{D}}(\beta^T x_i - y_i)^2$

Intuition on the Logistic Regression Max-Likelihood Objective

Intuition on the Objective

- Loss for example i is

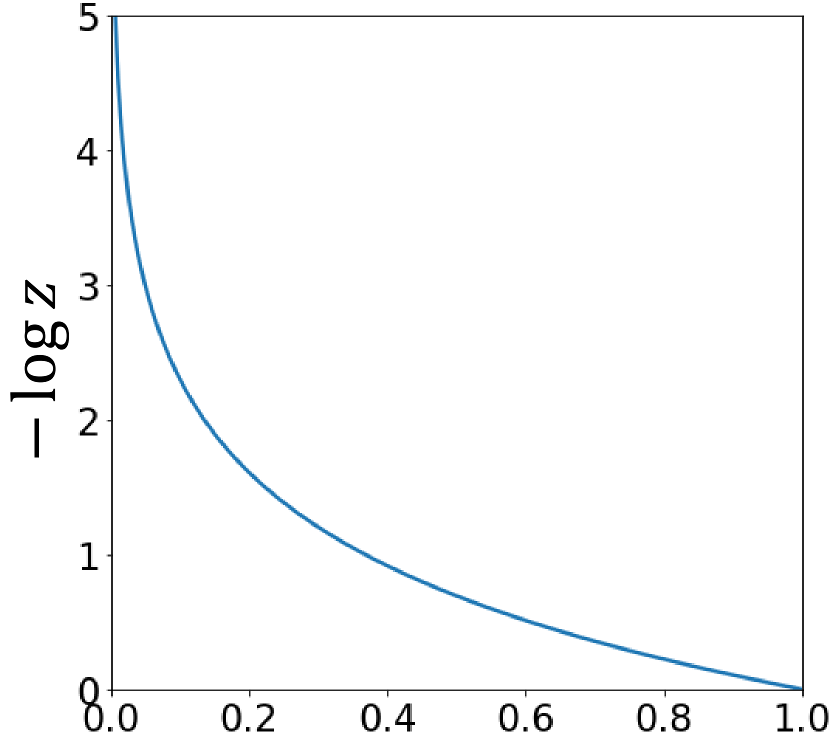
$$\begin{cases} -\log(\sigma(\beta^\top x_i)) & \text{if } y_i = 1 \\ -\log(1-\sigma(\beta^\top x_i)) & \text{if } y_i = 0 \end{cases}$$



Intuition on the Objective

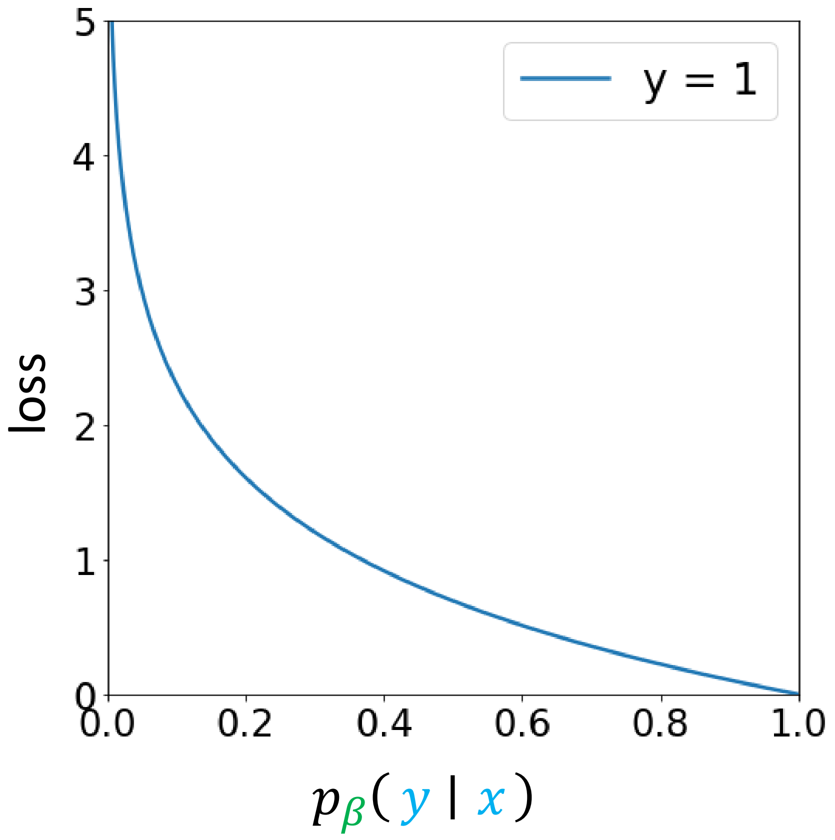
- Loss for example i is

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



Intuition on the Objective

- If $y_i = 1$:
  - If $p_\beta(\,Y = 1 \mid x_i\,) = 1$, then loss $= 0$
  - As $p_\beta(\,Y = 1 \mid x_i\,) \to 0$, loss $\to \infty$

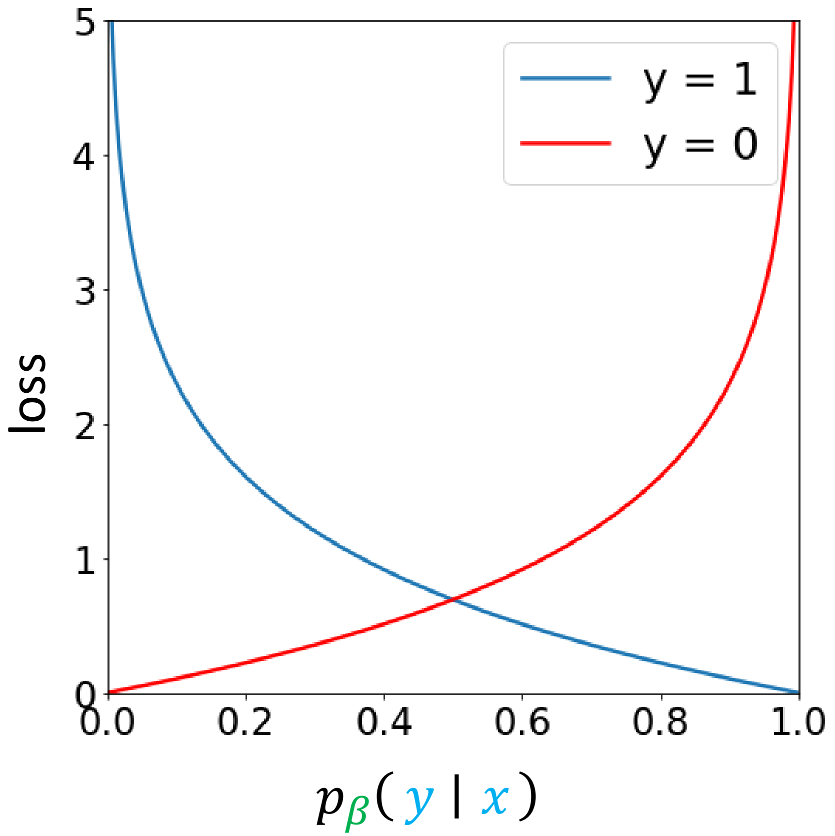


$$-y_i \cdot \boxed{\log\big(\sigma(\beta^\top x_i)\big)} - (1 - y_i) \cdot \log\big(1 - \sigma(\beta^\top x_i)\big)$$

Intuition on the Objective

- If $y_i = 1$:
  - If $p_\beta(Y = 1 \mid x_i) = 1$, then loss $= 0$
  - As $p_\beta(Y = 1 \mid x_i) \to 0$, loss $\to \infty$

- If $y_i = 0$
  - If $p_\beta(Y = 0 \mid x_i) = 1$, then loss $= 0$
  - As $p_\beta(Y = 0 \mid x_i) \to 0$, loss $\to \infty$



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

Optimizing the Logistic Regression Objective

Optimization for Logistic Regression

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

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

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

$$\xrightarrow{\sigma'(z) = \sigma(z)(1-\sigma(z))}$$

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

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

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

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 $\hat{\beta}(Z)$

Gradient Descent for Logistic Regression

- Initialize β
- Repeat until convergence

$$\beta_1 \leftarrow \beta_1 - \alpha \sum_{i=1}^{N} (\sigma(\beta^\top x_i) - y_i)$$

$$\beta_j \leftarrow \beta_j - \alpha \left[\sum_{i=1}^{N} (\sigma(\beta^\top x_i) - y_i) x_{ij} + \lambda \beta_j \right]$$

simultaneous update for j = 2 ... D