CIS 4190/5190: Lec 06 Wed Sep 18, 2024

Logistic Regression Part 2

Announcements

- Quiz 2 should be out today. Will be announced on Ed.

Recap: Logistic Regression so far

- Model class: $p(y|x) = \sigma(\beta^T x) = \frac{1}{1+e^{-\beta^T x}}$
  - The "raw" scores $\beta^T x$ are sometimes called "logits",
  - $\sigma(\cdot)$ is called the "logistic function" or "sigmoid function"

- "Negative Log Likelihood" (NLL) loss function:

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

prefers model parameters β that assign high probabilities to the true labels $y_i \in \{0,1\}$

Optimizing the Logistic Regression Objective

Optimization for Logistic Regression

$$-\sum_{i=1}^{N}\left[\, y_i \log \sigma(\beta^T \boldsymbol{x}_i) + (1 - y_i)\log(1 - \sigma(\beta^T \boldsymbol{x}_i))\,\right]$$

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

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

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

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

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

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

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

Q: What is the dimensionality of this RHS?

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 $\sigma(\cdot)$
  - Gradient of loss for i^{th} sample (x_i, y_i) w.r.t. parameter $\beta_j \propto$ error on that sample $\times$ j^{th} element of x_i.

- 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 $j = 2 \ldots D$

Understanding Regularization Better

Regularized Logistic Regression

- We can add ℓ_1 or ℓ_2 regularization to the NLL loss, e.g.:

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

- PS: Again, do not regularize the intercept term in the parameter vector if there is one. (You will usually add this)
- The NLL objective has a probabilistic interpretation as the likelihood of the dataset under the model.
- How should we understand the role of regularizers in this context?

Expressing Preferences over Parameters

- Recall that the maximum likelihood objective selected parameters purely based on the data fit:

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

- What if we expressed a preference over parameters "a priori" before ever having seen the data?

$$\underbrace{\max_{\beta} l_{\mathcal{D}}(\boldsymbol{\beta}) = \prod_{i=1}^{N} p(\, y_i \mid \boldsymbol{x_i} \,; \boldsymbol{\beta}) \underbrace{p(\beta)}_{\text{Prior}}}_{\text{Maximum "a posteriori" (MAP) objective}}$$

Regularization as a Prior

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

Plugging in Gaussian prior

$$L(\beta; Z) = p_{Y|X,\beta}(Y \mid X, \beta) \cdot N(\beta; 0, \sigma^2 I)$$

$$= \left(\prod_{i=1}^{n} p_{\beta}(y_i \mid x_i)\right) \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|\beta\|_2^2}{2\sigma^2}}$$

Taking logarithms and adding negative sign, the "loss function" is:

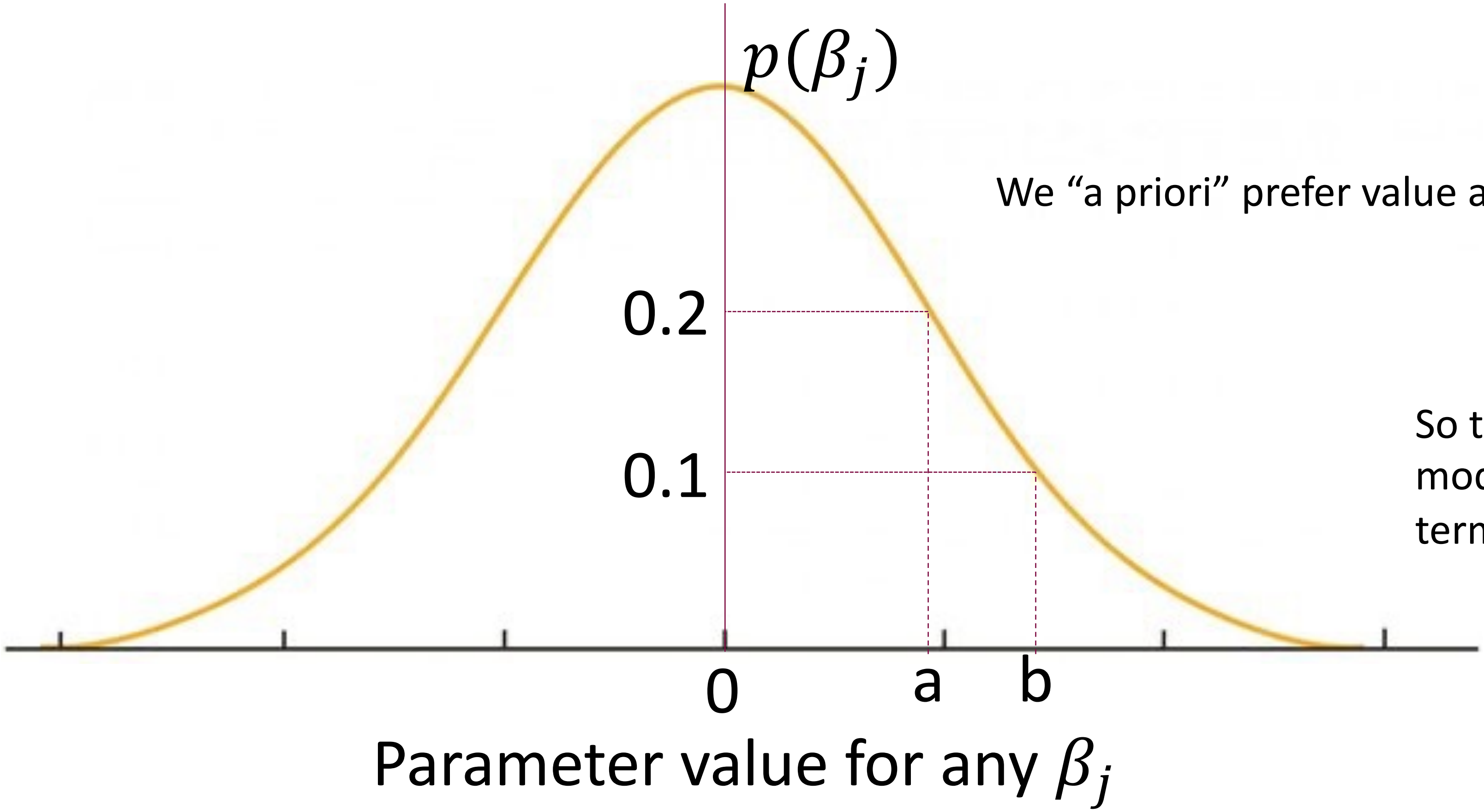
$$\ell(\beta; Z) = -\sum_{i=1}^{n} \log p_{\beta}(y_i \mid x_i) + \underbrace{\log \sigma\sqrt{2\pi}}_{\text{Constant, can remove}} + \underbrace{\frac{\|\beta\|_2^2}{2\sigma^2}}_{\text{regularization!}}$$

Constant, can remove

regularization! With $\lambda = \frac{1}{2\sigma^2}$

Recall: ℓ_2 Regularization: Gaussian Priors

- L2 regularization amounts to preferring smaller weights according to a Gaussian "prior" probability density function.

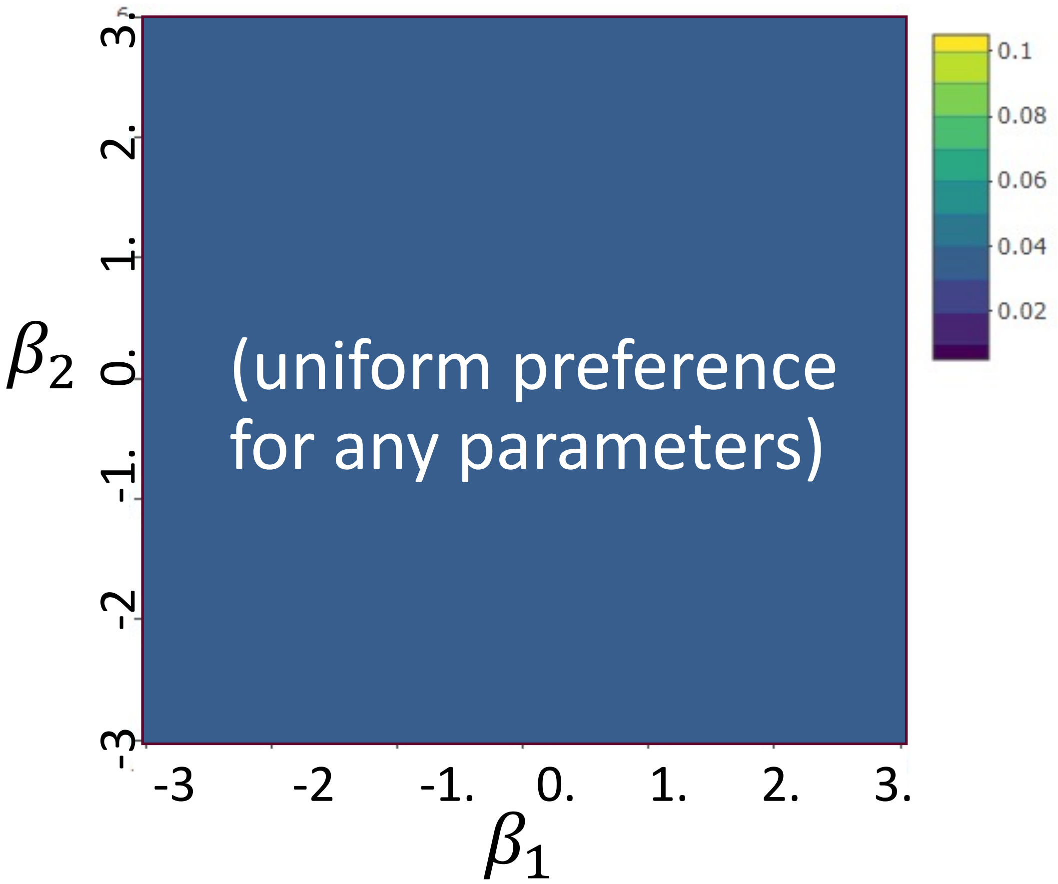


We "a priori" prefer value a for β_j twice as much as value b.

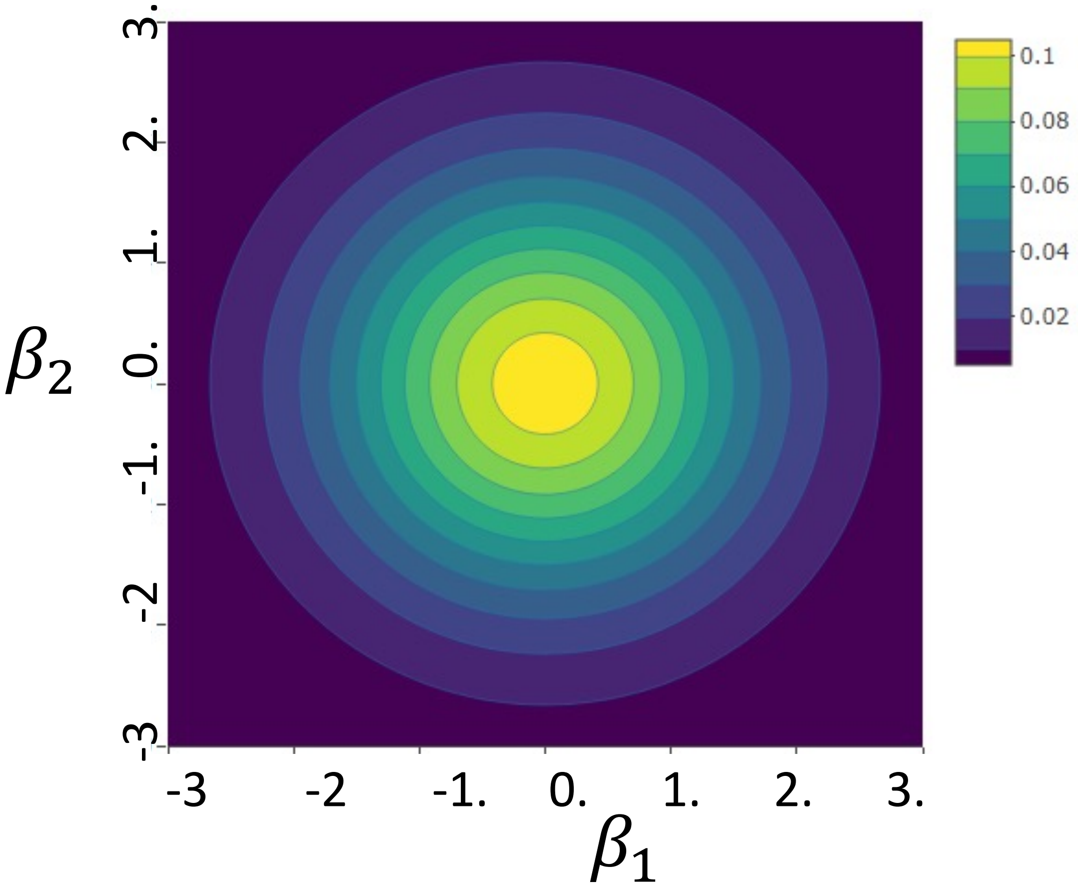
So the larger value is only selected for the model if it is *much* better for the data fit term (MSE)

Intuition on ℓ_2 Regularization: Gaussian Priors

Before regularization



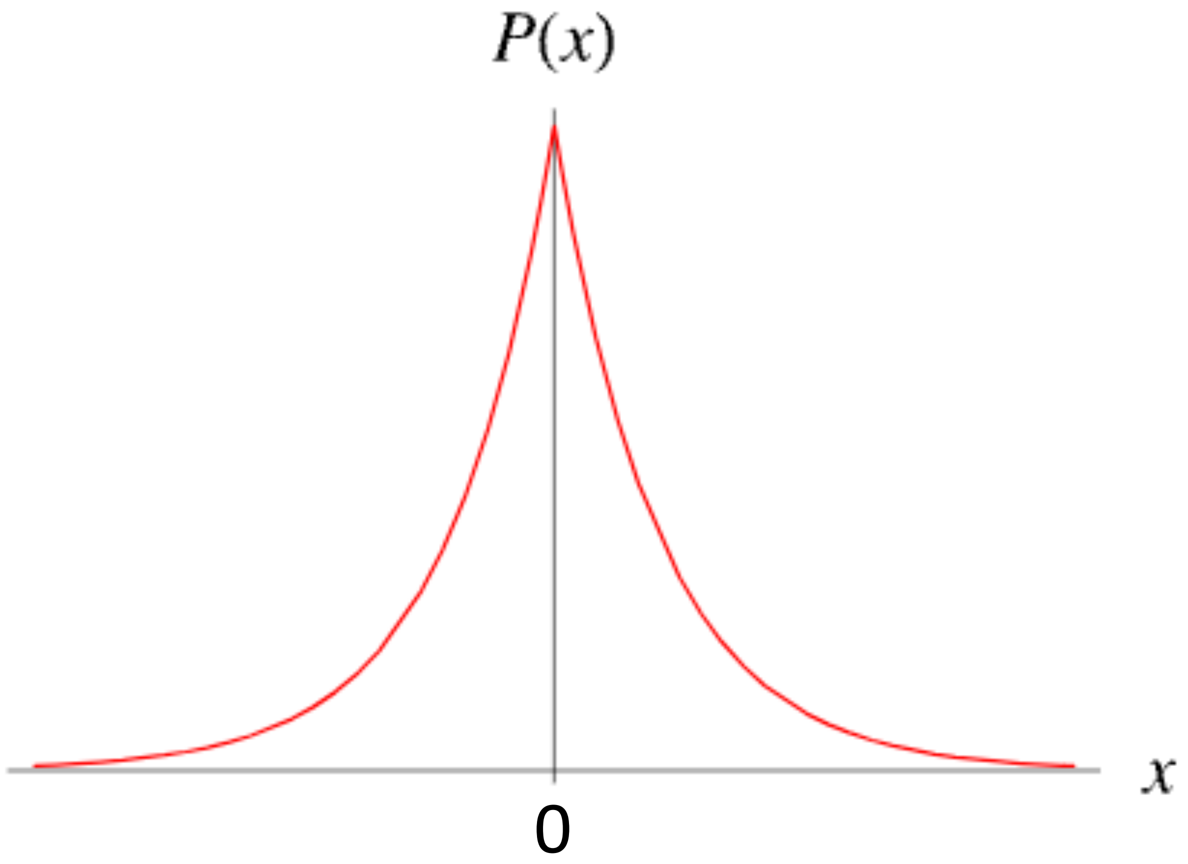
With L2 regularization



Intuition on ℓ_1 Regularization: Laplacian Priors

Similarly, ℓ_1 regularization corresponds to a Laplacian prior

$\beta_i \sim \text{Laplace}(0, \sigma^2)$ for each i

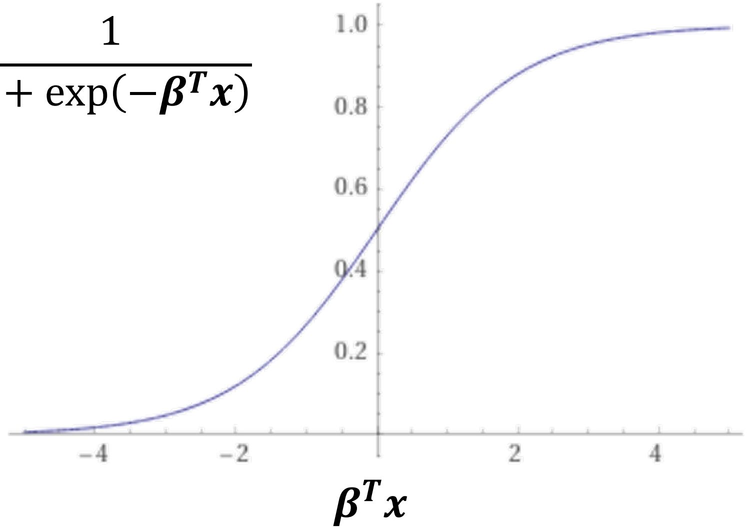


What Kinds of Functions Do Models With Small Weights Express?

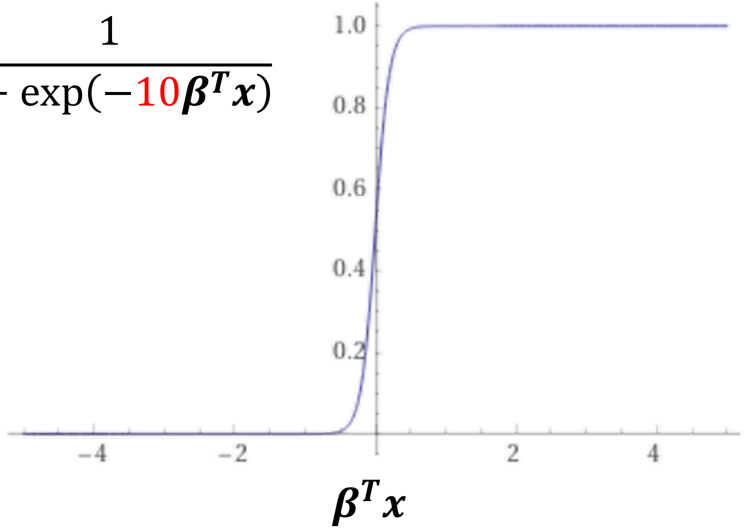
Scaling the Logistic Regression Parameter Vector

- Recall: $p_\beta(y=1|x) = \frac{1}{1+e^{-\boldsymbol{\beta}^T\boldsymbol{x}}}$
- The decision boundary is at $\boldsymbol{\beta}^T\boldsymbol{x}=0$
- If you replace $\boldsymbol{\beta}$ by $k\boldsymbol{\beta}$, where $k \gg 1$, what happens to:
  - The decision boundary?
  - The probability scores $p_\beta(y=1|x)$?

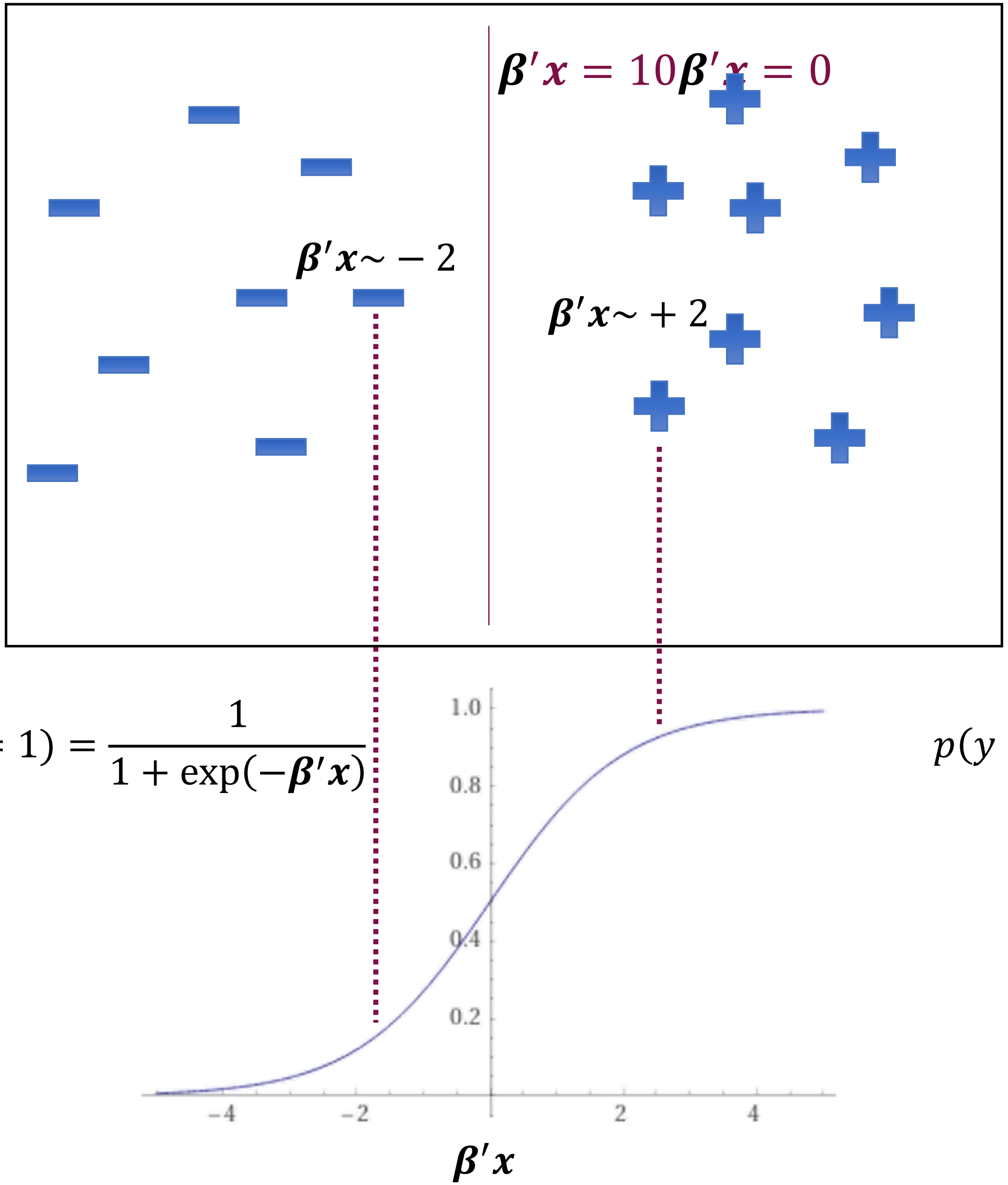
$$p(y=1) = \frac{1}{1+\exp(-\boldsymbol{\beta}^T\boldsymbol{x})}$$



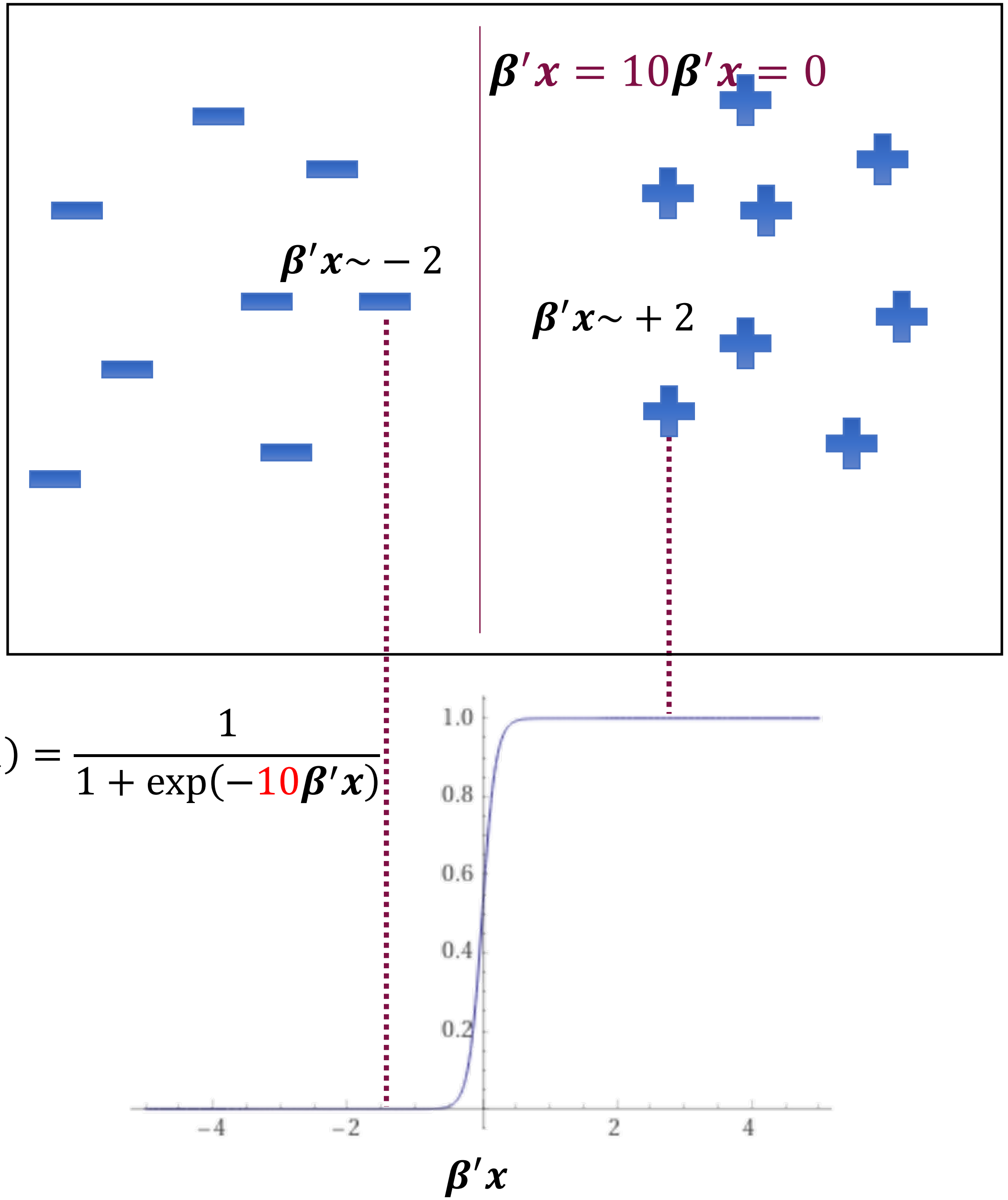
$$p(y=1) = \frac{1}{1+\exp(-10\boldsymbol{\beta}^T\boldsymbol{x})}$$



Smaller Parameters $\boldsymbol{\beta}$=> "Hedging Your Bets"



$$p(y=1) = \frac{1}{1+\exp(-\boldsymbol{\beta}'\boldsymbol{x})}$$



$$p(y=1) = \frac{1}{1+\exp(-10\boldsymbol{\beta}'\boldsymbol{x})}$$

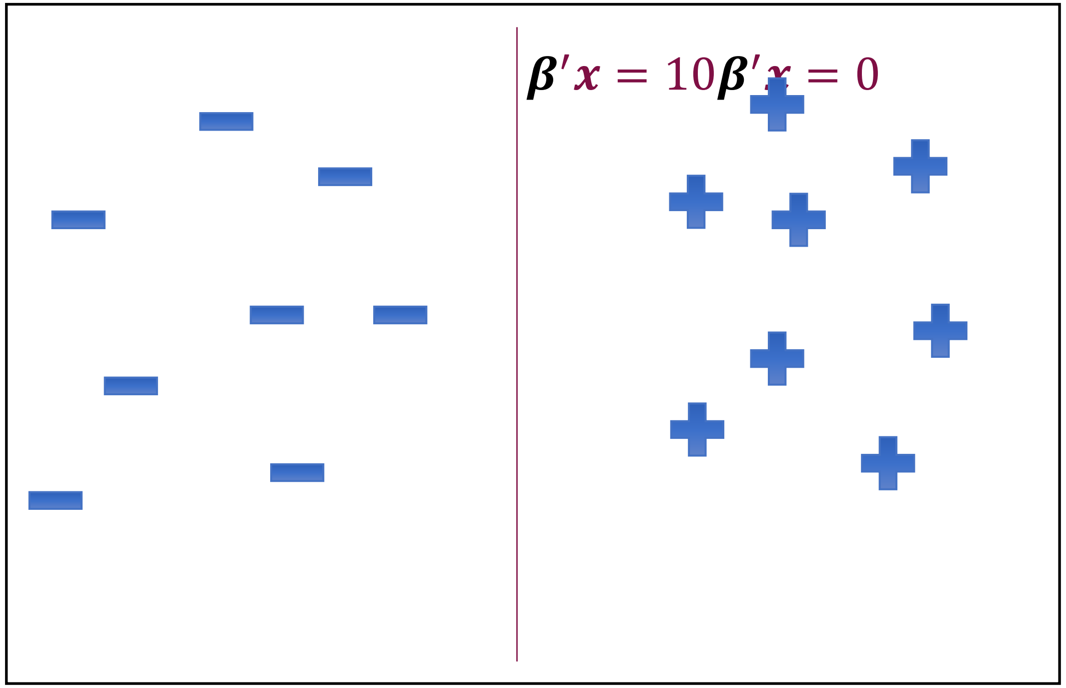
Regularization => Smaller $\boldsymbol{\beta}$=> "Hedging Your Bets"

- When parameters are scaled up by a constant factor, category assignments remain unchanged, but they are made with much higher confidence
- This provides a new insight about the role of regularization:
  - ℓ_1 and ℓ_2 regularization both penalize large $\boldsymbol{\beta}$, thus expressing the preference for less overconfident classification decisions on training data.
  - The resulting classifiers hedge their bets and perform better on test data, especially if the training dataset is small or noisy.

Which classifier would logistic regression learn *without any regularization* when trained on the dataset shown on the last slide?

Regularization is sometimes necessary!

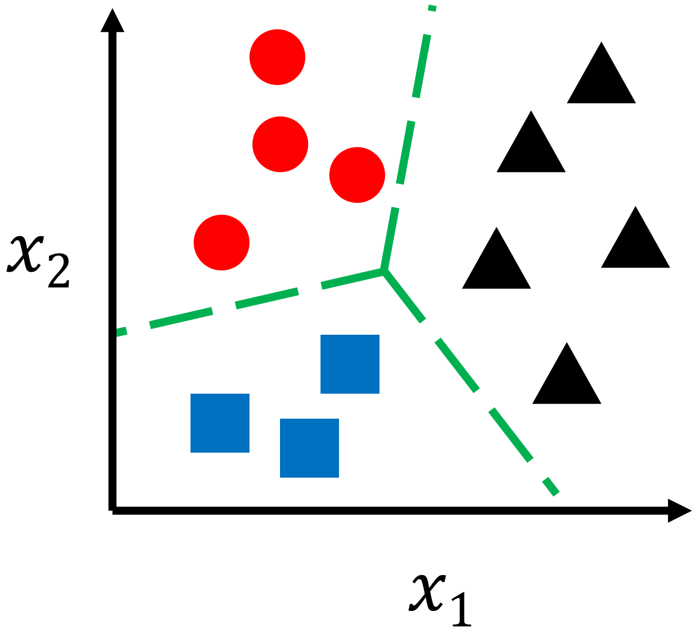
- If your data is linearly separable, then gradient descent on the logistic regression “diverges”.
  - Q: Why, and how does regularization stop this?



Multi-class logistic regression

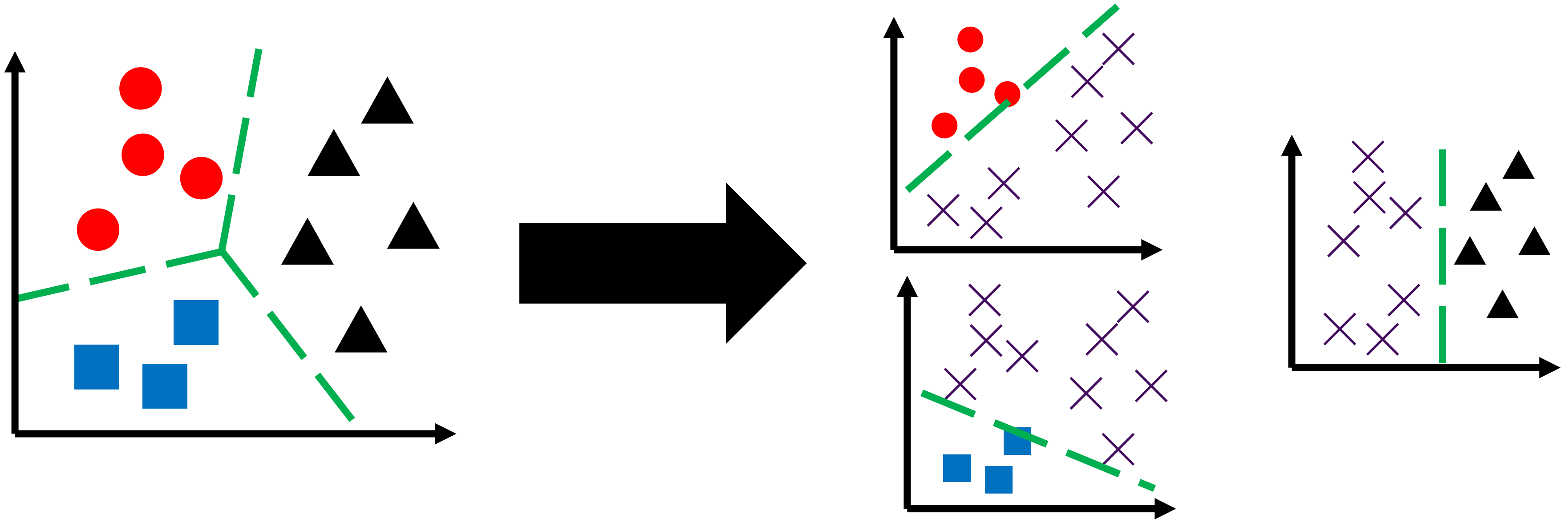
Multi-Class Classification

- 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 $y = \arg\max_{y'} p_{\beta_{y'}}(Y = 1 \mid x)$



Better Multi-Class Logistic Regression: Softmax

- **Strategy:** Include separate β_y for each label $y \in \mathcal{Y} = \{1, \ldots, k\}$
- 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 $\text{softmax}(z_1, \ldots, z_k) = \left[\frac{e^{z_1}}{\sum_{i=1}^k e^{z_i}} \quad \cdots \quad \frac{e^{z_k}}{\sum_{i=1}^k e^{z_i}}\right]$
- Then, $p_\beta(y \mid x) = \text{softmax}\left(\beta_1^\top x, \ldots, \beta_k^\top x\right)_y$
  - Thus, sometimes called **softmax regression**

Better Multi-Class Logistic Regression: Softmax

- **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 $\{\beta_y\}_{y \in \mathcal{Y}}$

CIS 4190/5190: Lec 06 Part 2 Wed Sep 18, 2024

Measuring Classification Performance

Classification Metrics

- While we minimize the NLL, we often evaluate using **accuracy = fraction of samples that are correctly predicted**
- However, even accuracy isn't necessarily the "right" metric.
  - Imbalanced data: If 99% of labels are negative (i.e., $y_i = 0$), accuracy of always predicting $f_\beta(x) = 0$ is 99%!
    - For instance, very few patients test positive for most diseases
    - "Imbalanced data"
  - Not all mistakes are the same:
    - e.g. "better that ten guilty persons go free than that one innocent person be convicted"
- What are alternative metrics for these settings? We will mostly discuss metrics for *binary* classification

Confusion Matrix

- **All test examples fall into one of the following buckets:**
  - **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 (↓)

		Predicted Class	
		Yes	No
Actual Class	Yes	TP(↑)	FN (↓)
	No	FP (↓)	TN(↑)

Q: How to extend this to multi-class?

Confusion Matrix

Actual Class		Predicted Class: Yes	Predicted Class: No
	Yes	TP(↑)	FN (↓)
	No	FP (↓)	TN(↑)

Classification Metrics In Terms of TP, TN, FP, FN

- Many metrics expressed in terms of these elements of the confusion matrix; for example:

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

Here n is the number of samples you tested on in total = TP+TN+FP+FN

Confusion Matrix

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

Accuracy = 0.8 [Q: Is this good?]

Classification Metrics

- For imbalanced datasets, we roughly want to disentangle:
  - Accuracy on “positive examples” (↑)
  - Accuracy on “negative examples” (↑)

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

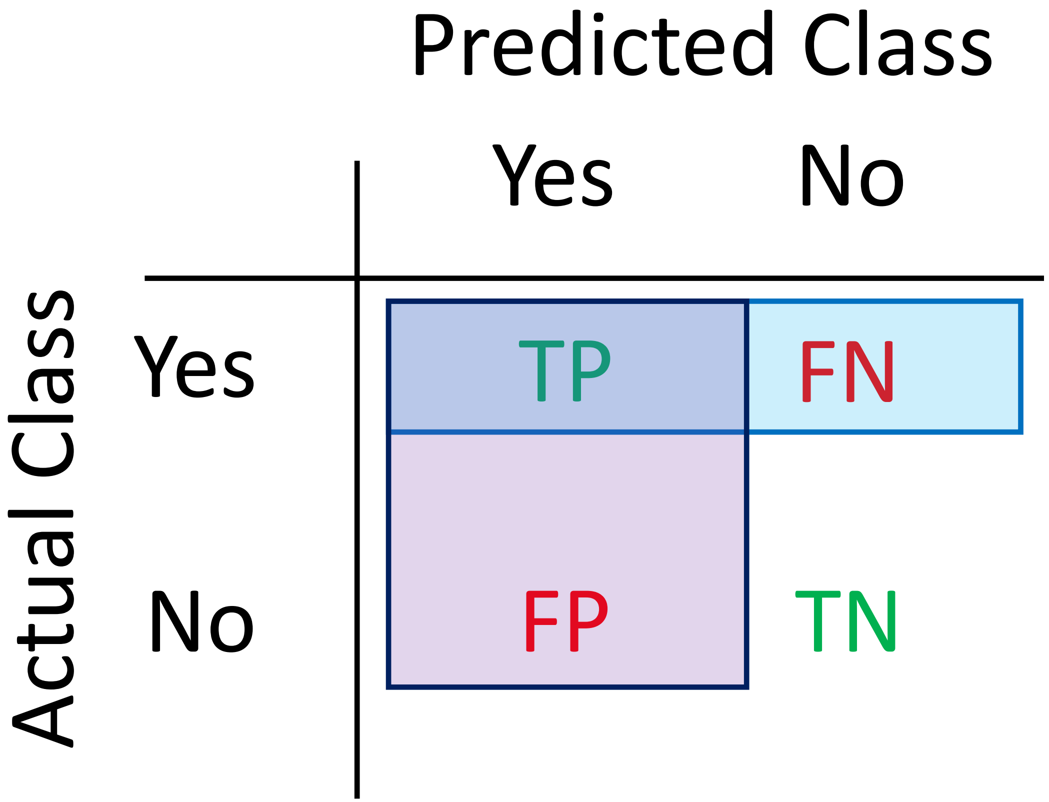
Precision & Recall

- **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)
  - Emphasized when it is important to avoid false negatives

- **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**
  - Emphasized when it is important to avoid false positives e.g. criminal law: “It is better that ten guilty persons escape than that one innocent suffer”

- Used in information retrieval, NLP

Precision & Recall



$$\text{Recall}(\uparrow) = \frac{TP}{TP + FN}$$

$$\text{Precision}(\uparrow) = \frac{TP}{TP + FP}$$

Precision & Recall

Predicted Class

Actual Class	Yes	No
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

$\text{recall} = 3/7$

$\text{precision} = 3/9$

True Positive Rate & False Positive Rate

- True Positive Rate / TPR (↑): fraction of **actual positives** that are **predicted positives**
- False Positive Rate / FPR (↓): fraction of **actual negatives** that are **predicted positives**

Sensitivity & Specificity

- **Sensitivity**(↑)**:** What fraction of **actual positives** are **predicted positive**?
  - **Good sensitivity:** If you have the disease, the test correctly detects it
  - Same as true positive rate, recall, etc.
  - "accuracy on positive samples"

- **Specificity**(↑)**:** What fraction of **actual negatives** are **predicted negative**?
  - **Good specificity:** If you do not have the disease, the test says so
  - Same as **true negative rate**
  - "accuracy on negative samples"

- Commonly used in medicine
- **Natural extension to multi-class:**
  - **Per-class accuracies** (↑) **:** for each class k, what fraction of class k samples are predicted as class k?

Sensitivity & Specificity

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

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

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

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

$\text{sensitivity} = 3/7$

$\text{specificity} = 37/43$

Optimizing something other than the NLL?

Classification Metrics

- **Obtaining a single metric from these various pairs of metrics?**
  - e.g., $F_1 \text{ score}(\uparrow) = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ is the harmonic mean
  - Mean per-class accuracy $(\uparrow)$: In binary classification, this is the mean of sensitivity (TPR) and specificity (TNR)
    - $\frac{TPR+TNR}{2}$
  - *Weighted* mean of per-class accuracy $(\uparrow)$: Set weights for each class w_P, w_N to indicate how much you care about accuracy on that class.
    - $\frac{w_P \times TPR + w_N \times TNR}{w_P + w_N}$
  - More advanced: "Area under precision-recall curve"/ "Area under receiver operating characteristic"

What is the “right” metric?

- No generally correct answer. Depends on the goals for the specific problem/domain
- Whatever metric you choose, to know whether you are doing anything at all useful, always a good idea to compare to a trivial baseline. e.g. always predicting 1 or always 0.

Q: Can you think of a “trivial baseline” for regression?

https://en.wikipedia.org/wiki/Confusion_matrix

Sources: [23][24][25][26][27][28][29][30] view · talk · edit

		Predicted condition			
	Total population $= P + N$	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	**True positive** (TP), hit	**False negative** (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
Actual condition	Negative (N)	**False positive** (FP), type I error, false alarm, overestimation	**True negative** (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P+N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN}$ $= 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

Optimizing a Classification Metric

- We are training a model to minimize NLL, but we have a different "true" metric that we actually want to optimize

- Two strategies (can be used together):
  - **Strategy 1: (After training)** Optimize prediction threshold threshold
  - **Strategy 2: (Before training)** Upweight positive (or negative) examples

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

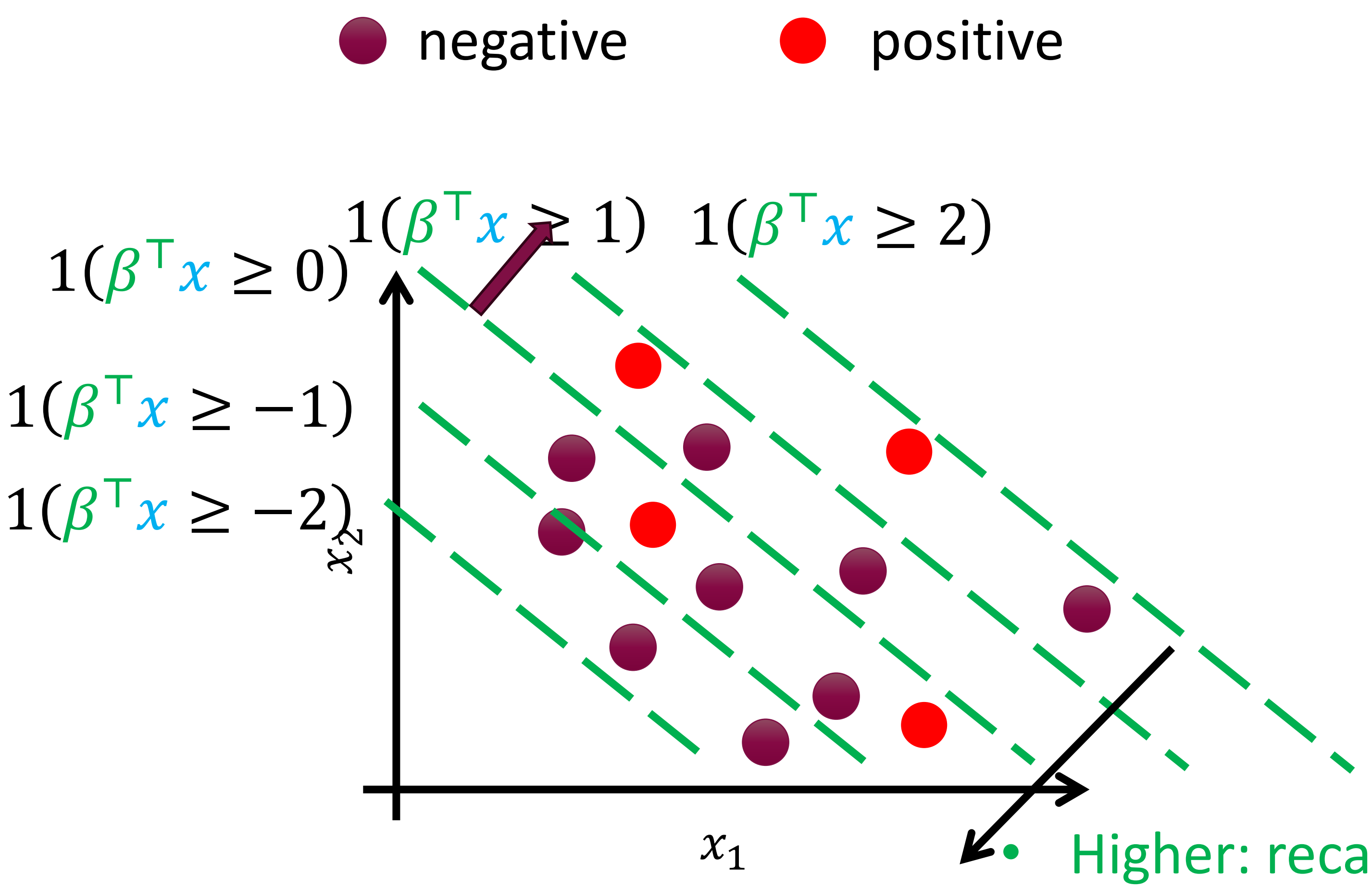
$$f_\beta(x) = 1(\beta^\top x \geq 0)$$

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

$$f_\beta(x) = 1(\beta^\top x \geq \tau)$$

Optimizing Prediction Threshold



- Higher: recall, TPR, sensitivity
- Lower: precision, FPR, specificity

No free lunch. Your new classifier is not automatically objectively better, but possible to be better than original NLL-optimal classifier on your "true" metric.