Upcoming Deadlines

- **Project Team Formation due tonight**
- **Quiz 1 due tomorrow**
- HW 2 due in one week

Lecture 6: Logistic Regression

CIS 4190/5190

Spring 2023

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

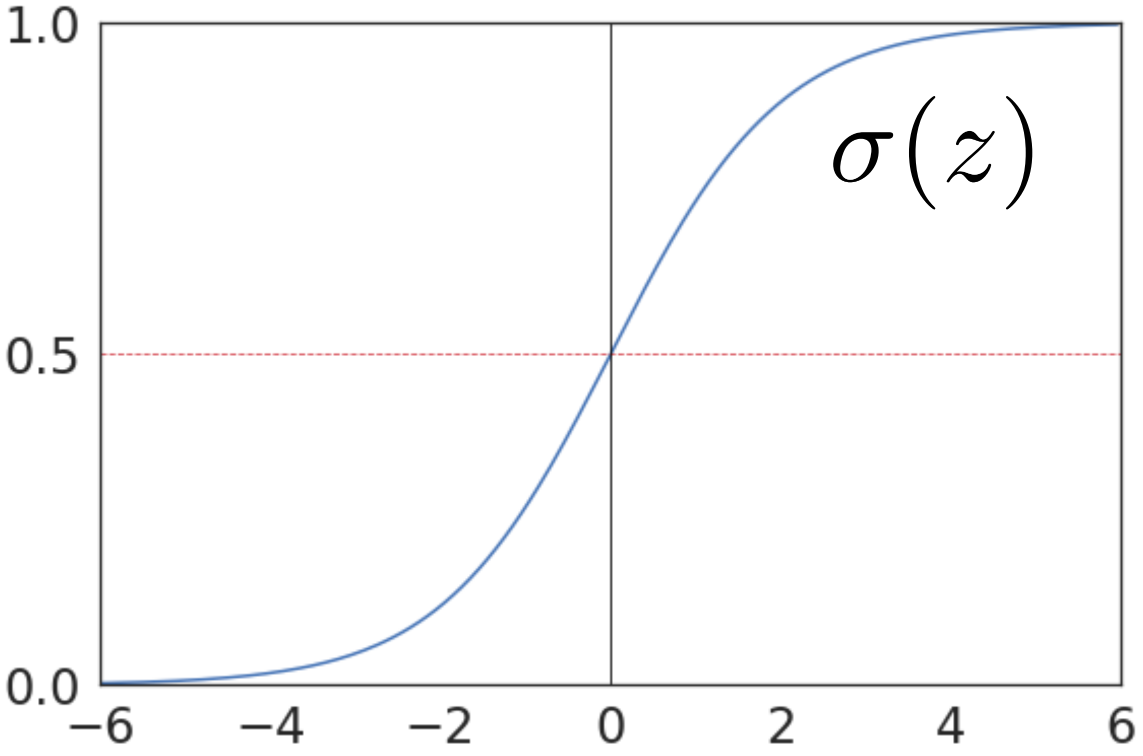
- Very powerful framework for designing cutting edge ML algorithms
  - Write down the “right” likelihood, form tractable approximation if needed
  - Especially useful for thinking about non-i.i.d. data

Recap: Logistic Regression

- Consider the following choice:

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

Recap: Logistic Regression



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

Recap: Logistic Regression

- **Model family:** Linear classifiers $f_\beta(x) = 1(\beta^\top x \geq 0)$

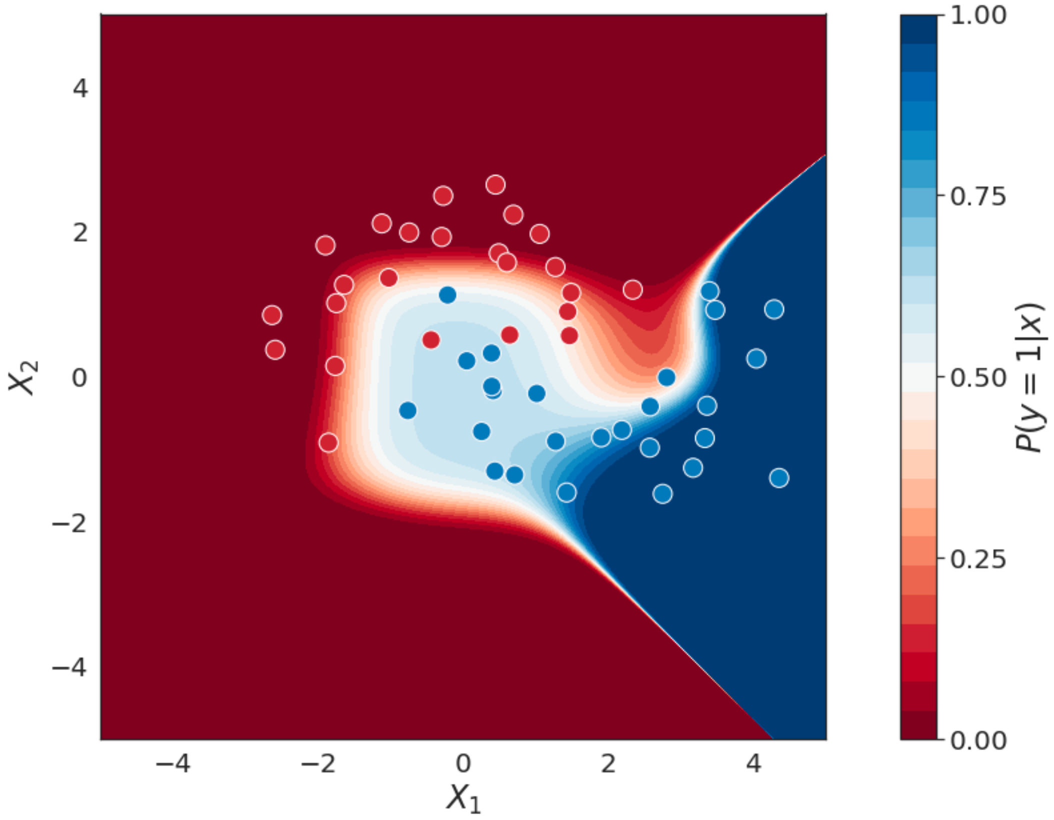
Loss function: Negative log likelihood

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

- **Optimizer:** Gradient descent

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.:

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

- 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)?

- Consider the modified likelihood

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

Regularization as a Prior

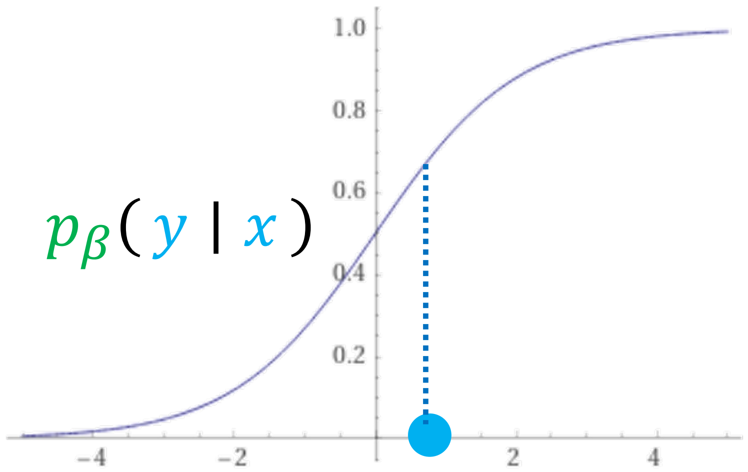
- So far, we have not assumed any distribution over the parameters β
  - What if we assume $\beta \sim N(0, \sigma^2 I)$ (the d dimensional normal distribution)?
- Consider the modified NLL

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

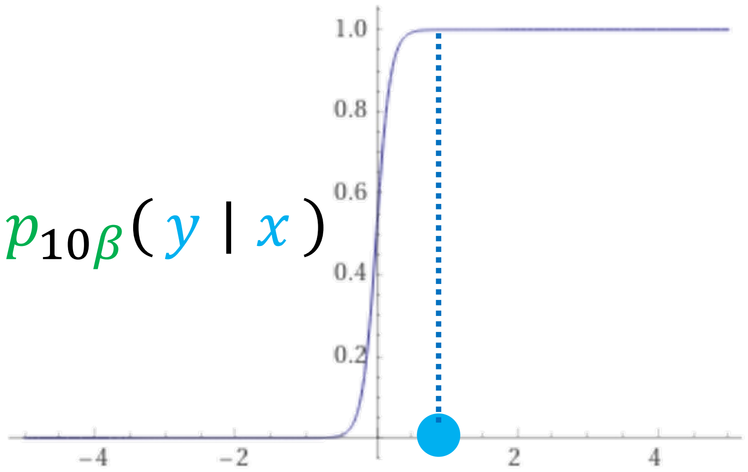
- Obtain L_2regularization on β!
  - With $\lambda = \frac{1}{2\sigma^2}$
  - If $\beta_i \sim \mathrm{Laplace}(0, \sigma^2)$ for each i, obtain L_1regularization

Additional Role of Regularization

- In p_β, if we replace β with $c \cdot \beta$, 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



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



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

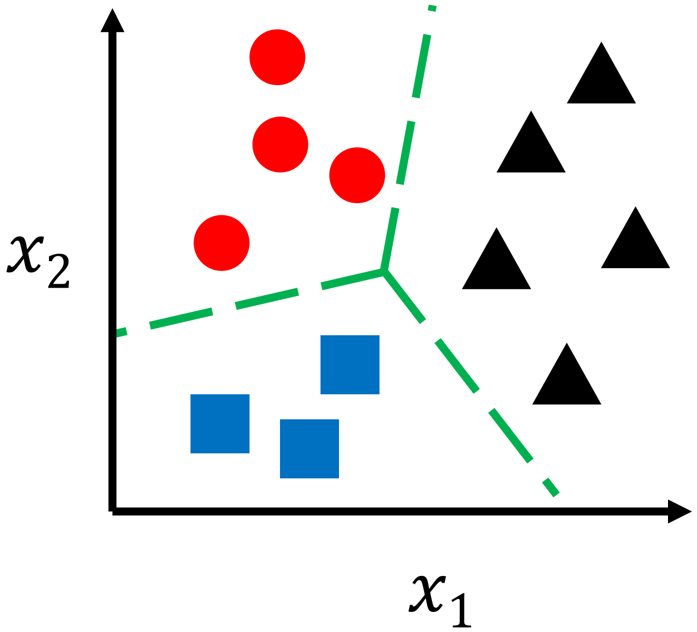
Additional Role of Regularization

- 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., $\beta \to \pm\infty$)

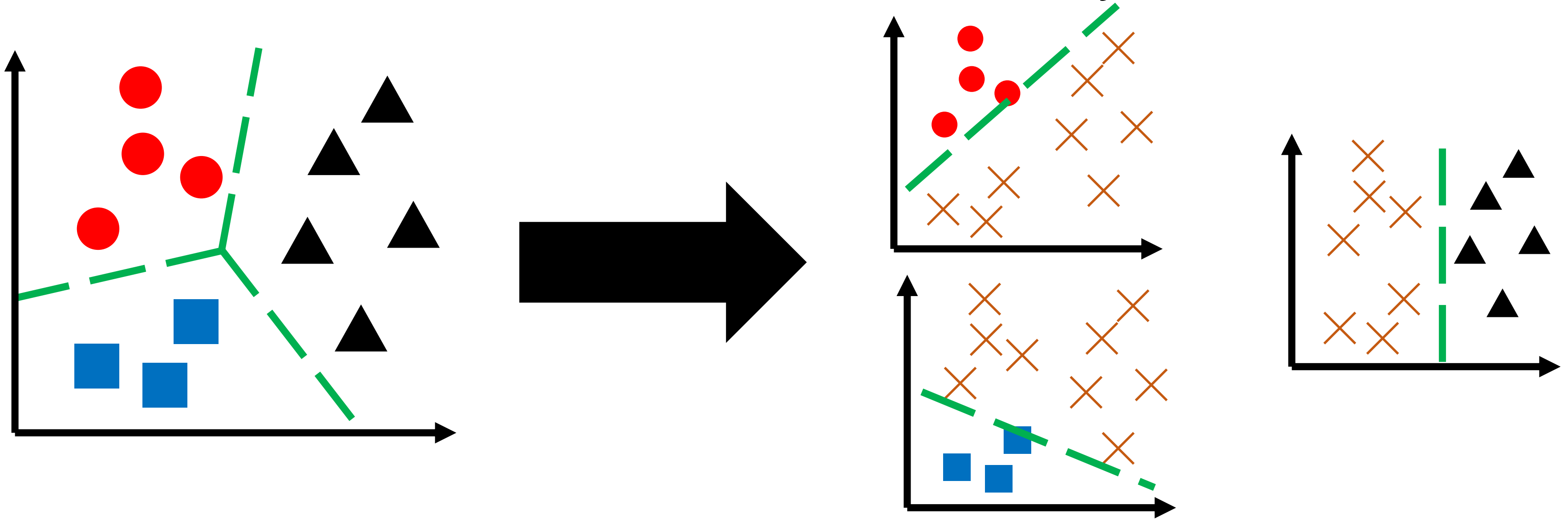
Multi-Class Classification

- 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 $y \in \mathcal{Y} = \{1, \ldots, k\}$
- 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 $\mathrm{softmax}(z_1, \ldots, z_k) = \left[\frac{e^{z_1}}{\sum_{i=1}^k e^{z_i}} \quad \cdots \quad \frac{e^{z_k}}{\sum_{i=1}^k e^{z_i}}\right]$
- Then, $p_\beta(y \mid x) = \mathrm{softmax}\left(\beta_1^\top x, \ldots, \beta_k^\top x\right)_y$
  - 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	sensitivity $= 3/7$
	No	6 FP	37 TN	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

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	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

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

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

Precision & Recall

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

$\text{recall} = 3/7$

$\text{precision} = 3/9$

Classification Metrics

- **How to obtain a single metric?**
  - Combination, e.g., F_1 score $= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ is the harmonic mean
  - More on this later
- **How to choose the "right" metric?**
  - No generally correct answer
  - Depends on the goals for the specific problem/domain

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:** Optimize prediction threshold threshold
  - **Strategy 2:** Upweight positive (or negative) examples

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

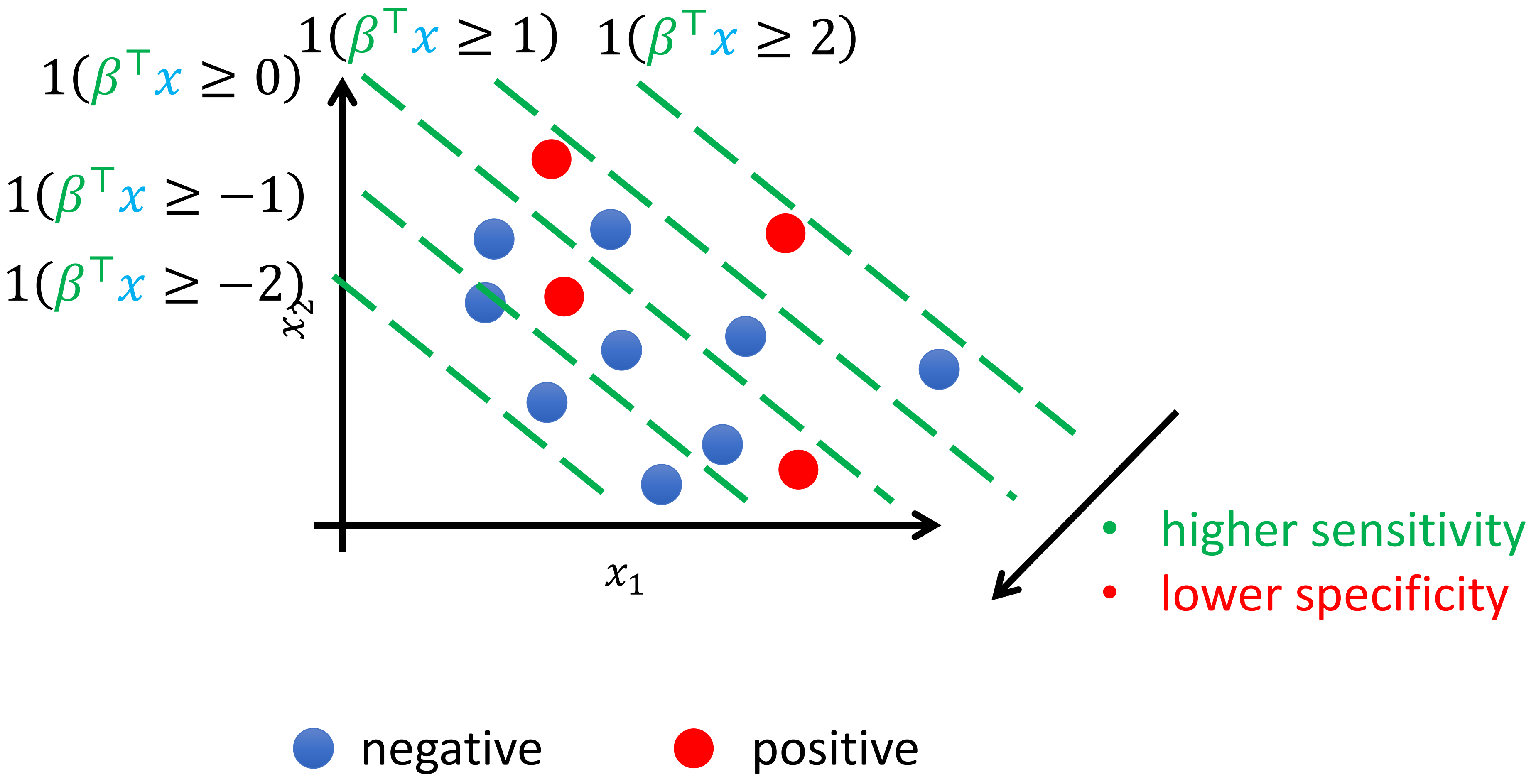
$$f_\beta(x) = 1(\beta^\top x \geq 0)$$

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

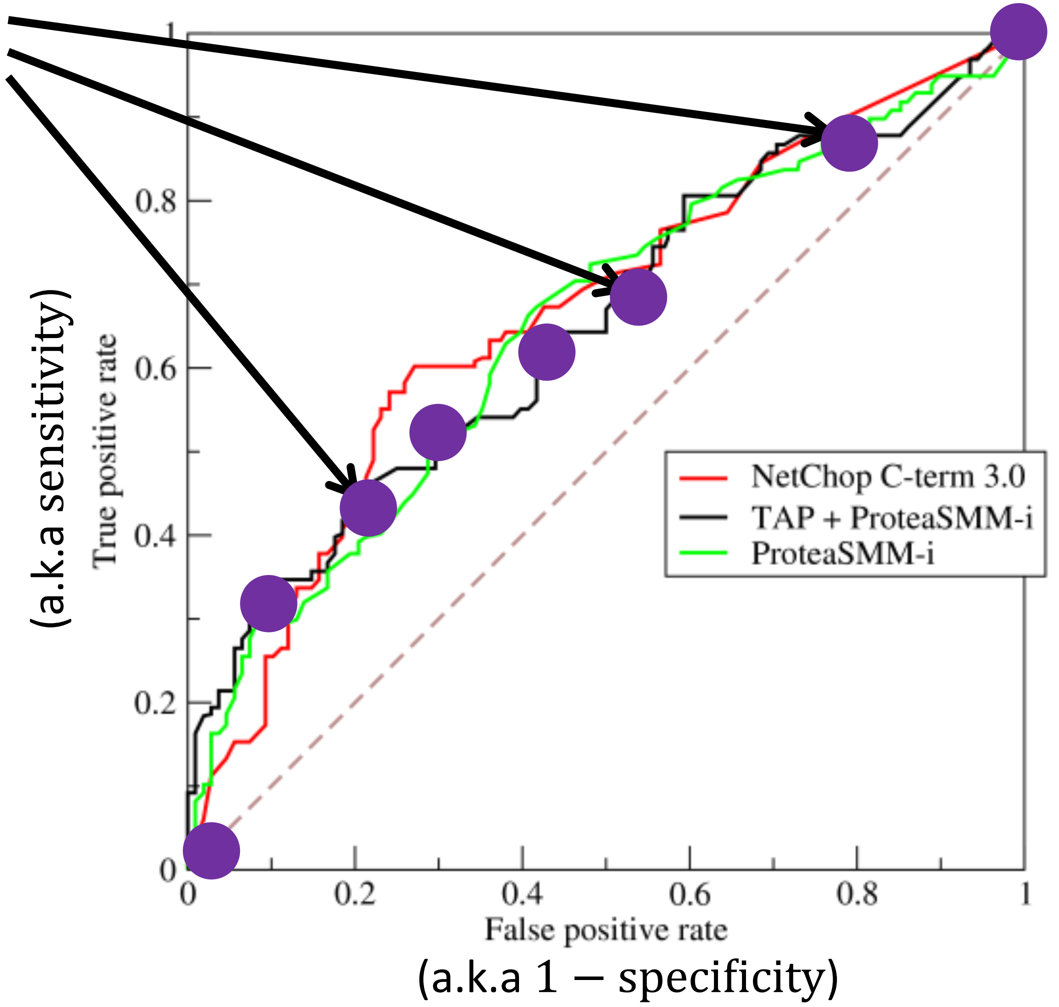
$$f_\beta(x) = 1(\beta^\top x \geq \tau)$$

Optimizing Prediction Threshold



Visualization: ROC Curve

Each point on this curve corresponds to a choice of τ



Aside: Area under ROC curve is another metric people consider when evaluating $\hat{\beta}(Z)$

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

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

- Unlike most hyperparameters, we choose this one **after** we have already fit the model on the training data
  - Then, choose the value of τ that optimizes the desired metric
  - Fit using validation data (training data is OK if needed)

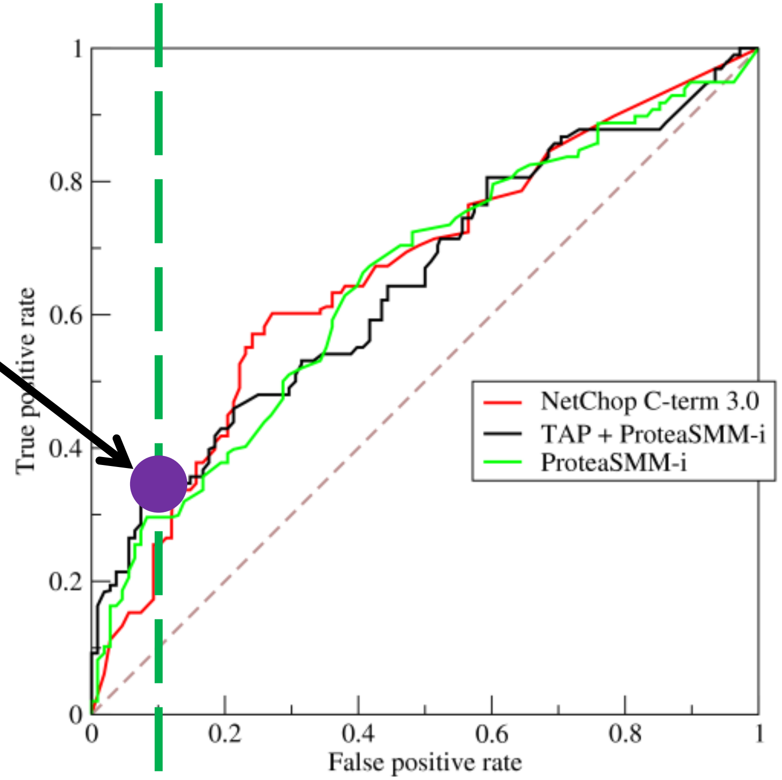
Optimizing Prediction Threshold

- **Step 1:** Compute the optimal parameters $\hat{\beta}(Z_{\text{train}})$
  - Using gradient descent on NLL loss over the training dataset
  - **Resulting model:** $f_{\hat{\beta}(Z_{\text{train}})}(x) = 1\left(\hat{\beta}(Z_{\text{train}})^\top x \geq 0\right)$
- **Step 2:** Modify threshold τ in model to optimize desired metric
  - Search over a fixed set of τ on the validation dataset
  - **Resulting model:** $f_{\hat{\beta}(Z_{\text{train}}),\hat{\tau}(Z_{\text{val}})}(x) = 1\left(\hat{\beta}(Z_{\text{train}})^\top x \geq \hat{\tau}(Z_{\text{val}})\right)$
- **Step 3:** Evaluate desired metric on test set

Choice of Metric Revisited

- **Common strategy:** Optimize one metric at fixed value of another

Choose τ corresponding to model at this point



$\text{specificity} = 0.9$

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:** Optimize prediction threshold threshold
  - **Strategy 2:** Upweight positive (or negative) examples

Class Re-Weighting

- **Weighted NLL:** Include a class-dependent weight w_y:

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

- **Intuition:** Tradeoff between accuracy on negative/positive examples
  - To improve sensitivity (true positive rate), upweight positive examples
  - To improve specificity (true negative rate), upweight negative examples

- Can use this strategy to learn β, and the first strategy to choose τ

Classification Metrics

- NLL isn’t usually the “true” metric
  - Instead, frequently used due to good computational properties
- Many choices with different meanings
- Typical strategy:
  - Learn β by minimizing the NLL loss
  - Choose class weights w_y and threshold τ to optimize desired metric