Classification with Linear Models

Losses for linear classification, logistic regression, multiclass classification

Programming and Statistical Analysis, 2025

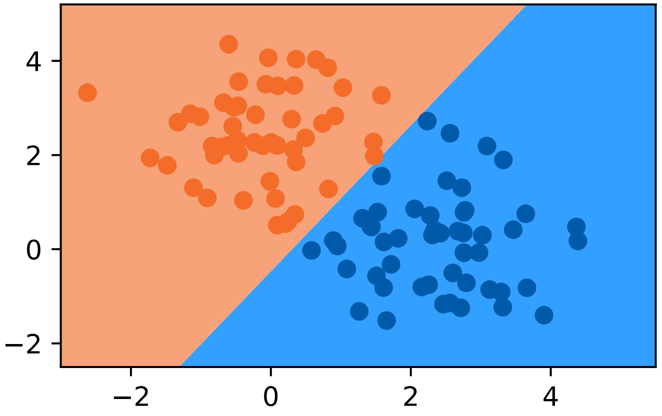
Majid Sohrabi
National Research University Higher School of Economics

Can’t we just use linear regression for classification?

Classification with linear regression

Classification:

$$\hat{f}(x) = \text{sign}[\theta^{\mathrm{T}} x]$$

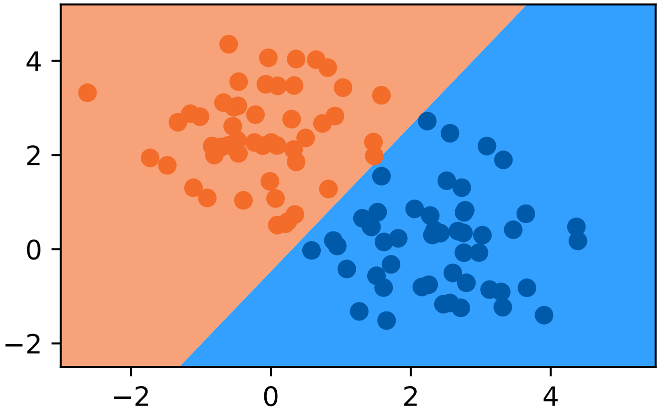


- For binary classification task, assign:
  - $y = +1$ for **positive** class
  - $y = -1$ for **negative** class

Classification with linear regression

Classification:

$$\hat{f}(x) = \text{sign}[\theta^{\text{T}} x]$$

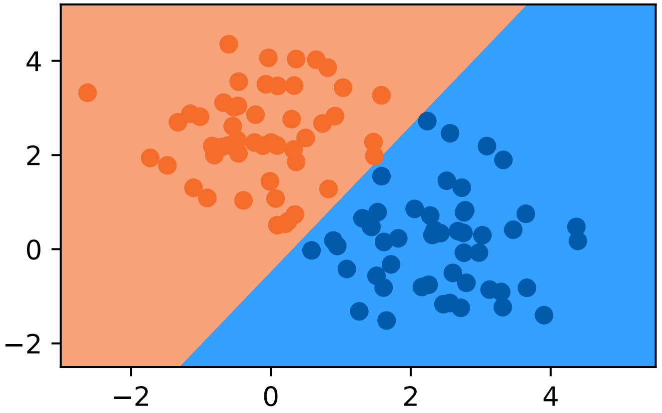


- For binary classification task, assign:
  - $y = +1$ for **positive** class
  - $y = -1$ for **negative** class
- Solve linear regression for $\hat{y} = \theta^T x$ with MSE loss

Classification with linear regression

Classification:

$$\hat{f}(x) = \text{sign}[\theta^{\mathrm{T}} x]$$

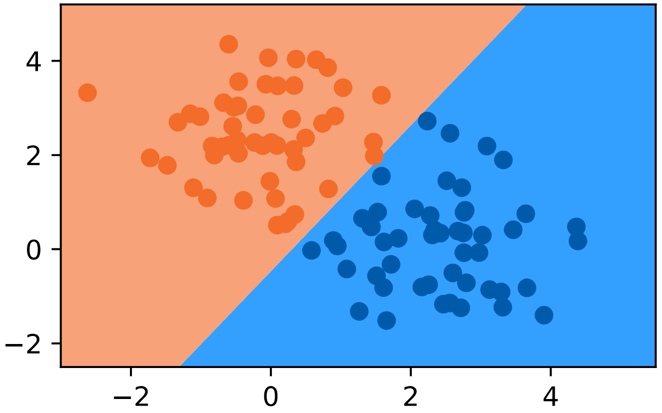


- For binary classification task, assign:
  - $y = +1$ for **positive** class
  - $y = -1$ for **negative** class
- Solve linear regression for $\hat{y} = \theta^T x$ with MSE loss
- Classify with $\text{sign}[\hat{y}]$

Classification with linear regression

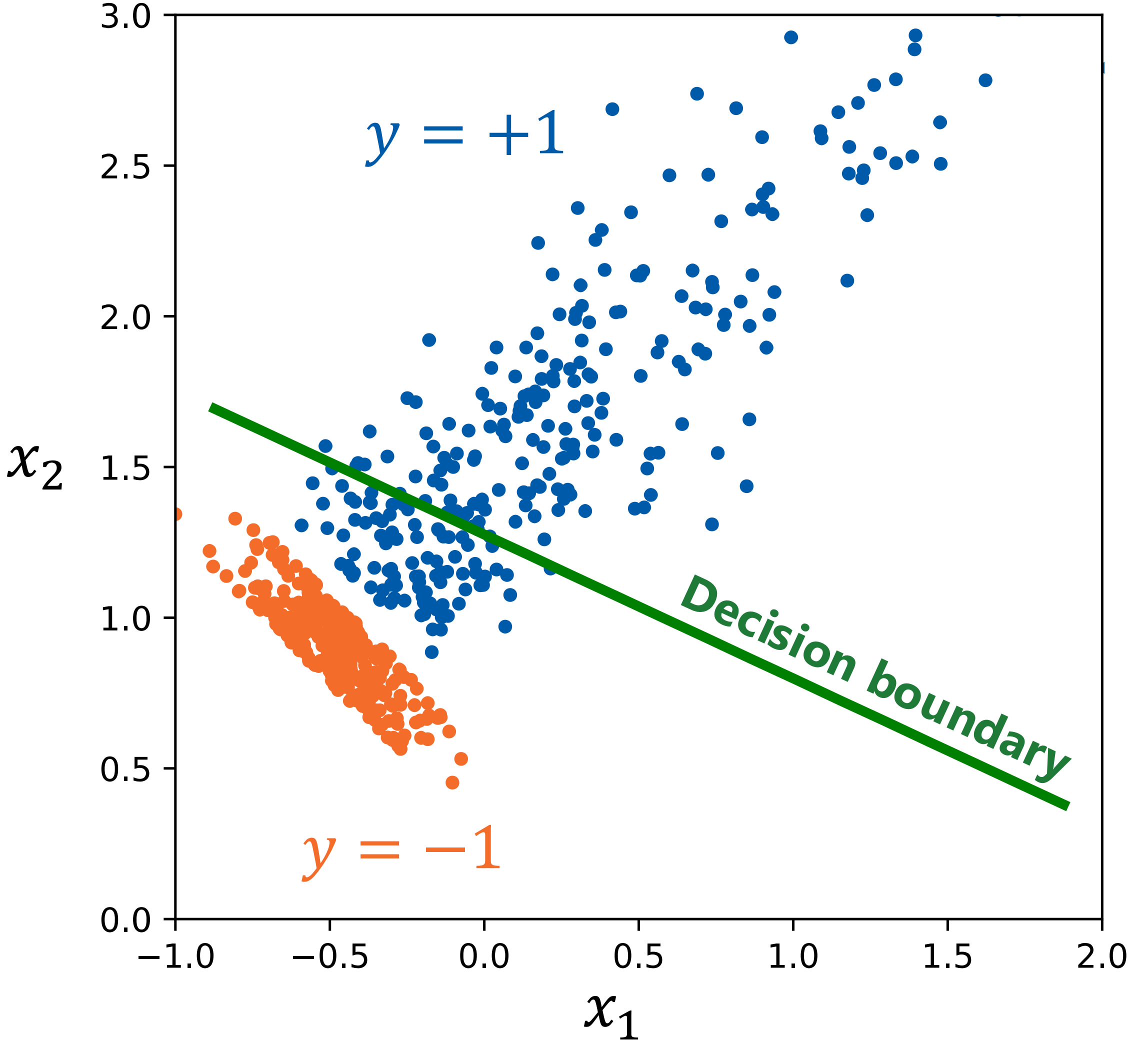
Classification:

$$\hat{f}(x) = \mathrm{sign}[\theta^{\mathrm{T}} x]$$



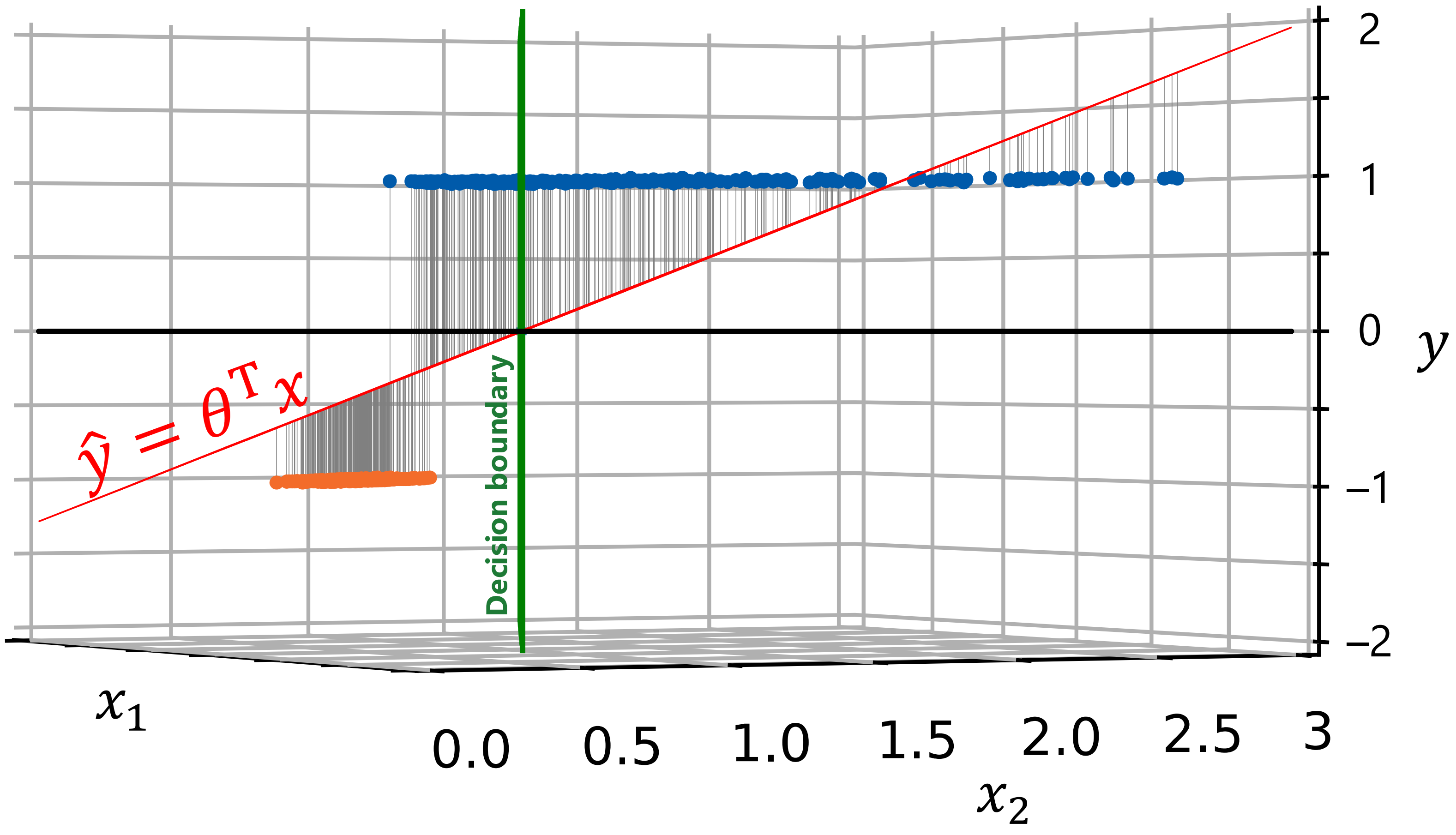
- For binary classification task, assign:
  - $y = +1$ for **positive** class
  - $y = -1$ for **negative** class
- Solve linear regression for $\hat{y} = \theta^T x$ with MSE loss
- Classify with $\mathrm{sign}[\hat{y}]$
- Any problems with this approach?

Classification with linear regression



- May face problems when classes are unbalanced or have different spread

Classification with linear regression



MSE loss makes the model **avoid high residuals**

at a price of **pushing the decision boundary** towards the class with higher spread

Can we find a better loss function?

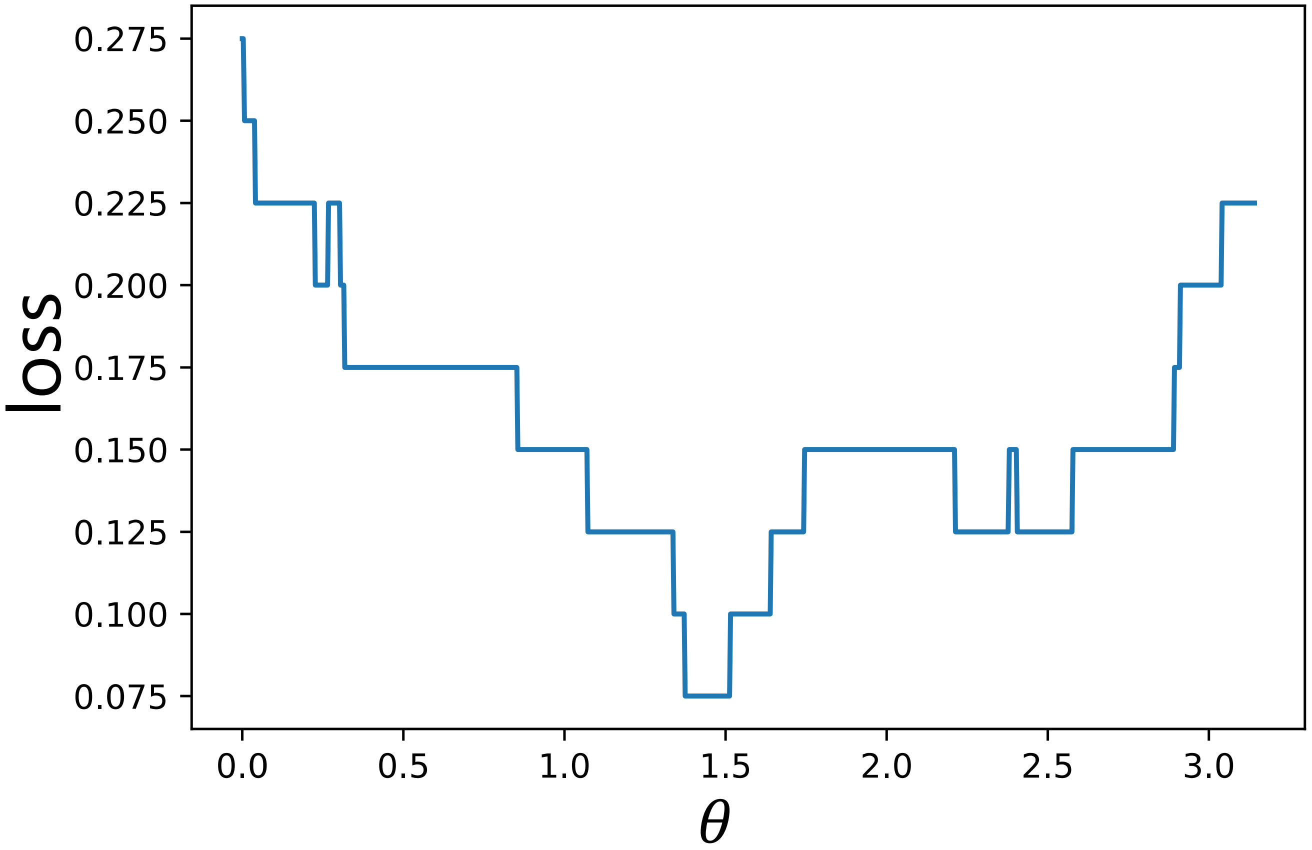
Classification loss functions

0-1 Loss

- Probability of an error

$$\mathcal{L}_{0-1} = \frac{1}{N} \sum_{i=1\ldots N} \mathbb{I}\left(\theta^{\mathrm{T}} x_i \cdot y_i < 0\right)$$

$$y_i \in \{-1, +1\}$$

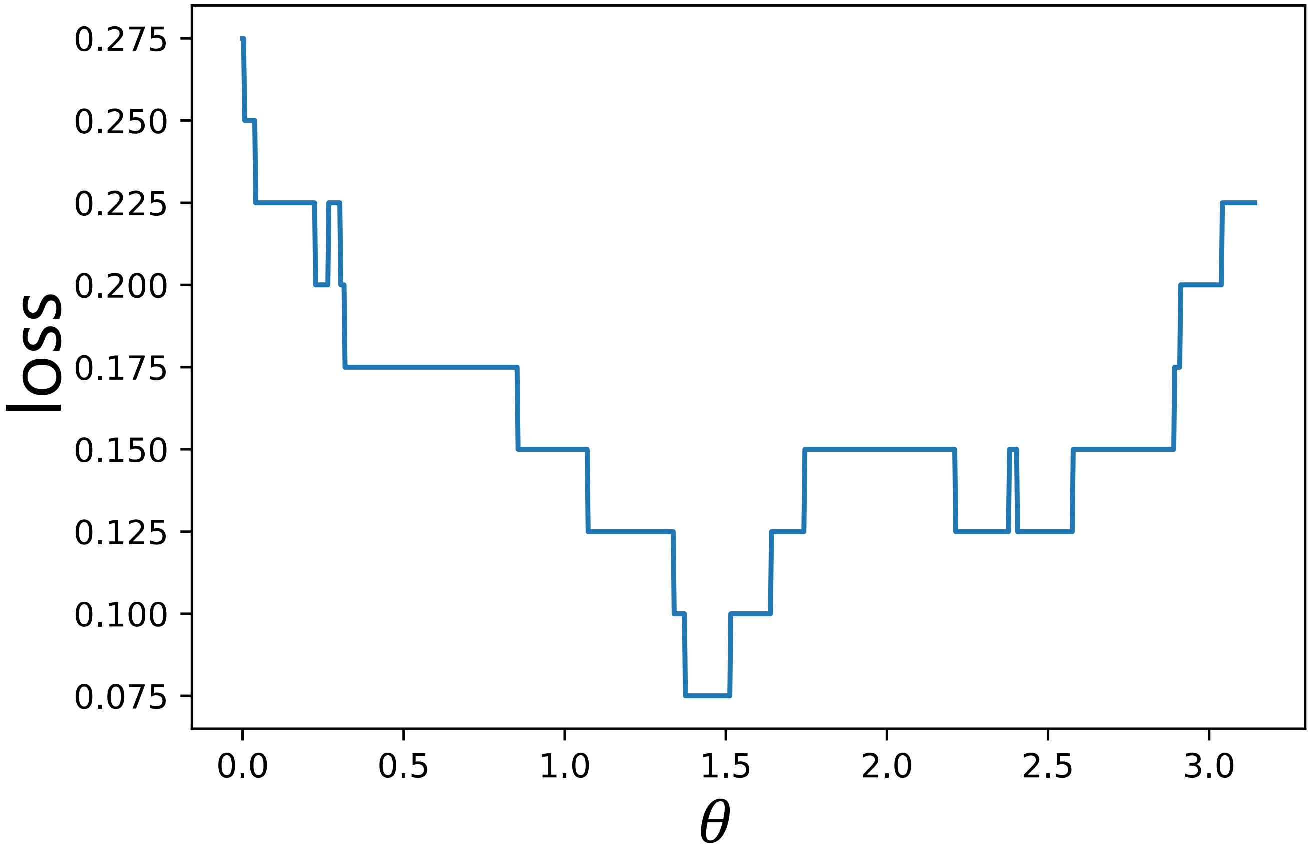


0-1 Loss

- Probability of an error

$$\mathcal{L}_{0-1} = \frac{1}{N} \sum_{i=1 \dots N} \mathbb{I}\left(\theta^{\mathrm{T}} x_i \cdot y_i < 0\right)$$

$$y_i \in \{-1, +1\}$$



- Can't optimize **piecewise constant** function with gradient-based methods*

*other techniques exist (still quite limited)

Margin

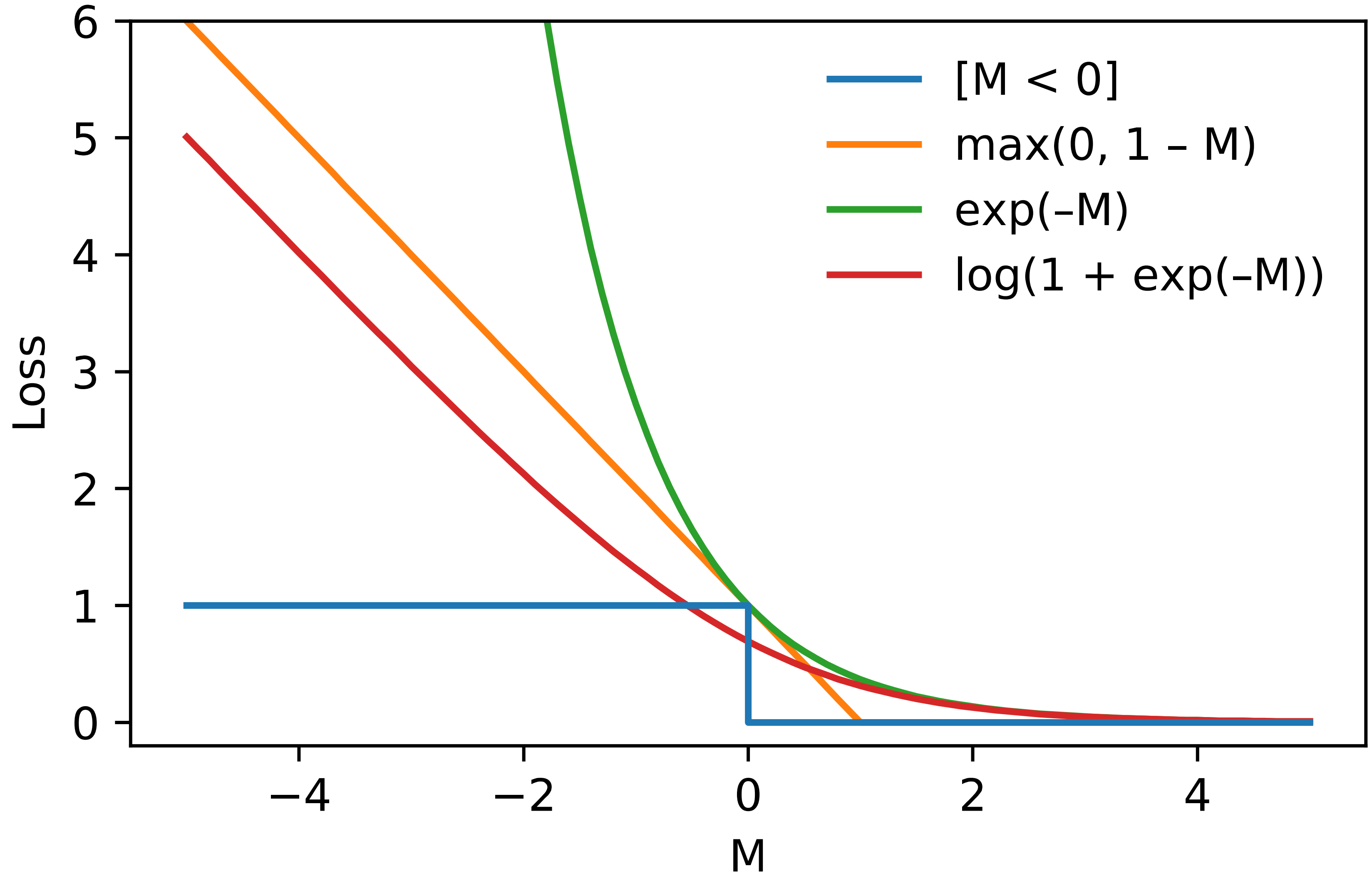
$$M = \theta^{\mathrm{T}} x \cdot y$$

$$\mathcal{L}_{0-1} = \frac{1}{N} \sum_{i=1...N} \mathbb{I}(\underbrace{\theta^{\mathrm{T}} x_i \cdot y_i}_{\text{margin}} < 0)$$

$M > 0$ – correct classification

$M < 0$ – incorrect classification

Upper bounds on 0-1 loss



Instead of optimizing the 0-1 loss we can optimize a **differentiable upper bound**

Logistic Regression

Idea

- Let's model the **class probabilities**

$$P(y = +1|x) = \widehat{f_\theta}(x)$$
$$P(y = -1|x) = 1 - \widehat{f_\theta}(x)$$

Idea

- Let's model the **class probabilities**

$$P(y = +1|x) = \widehat{f_\theta}(x)$$
$$P(y = -1|x) = 1 - \widehat{f_\theta}(x)$$

- Fit with **maximum (log) likelihood**

Idea

$$\text{Likelihood} = \prod_{i=1\dots N} P(y_i|x_i)$$

- Let's model the **class probabilities**

$$P(y = +1|x) = \widehat{f_\theta}(x)$$
$$P(y = -1|x) = 1 - \widehat{f_\theta}(x)$$

- Fit with **maximum (log) likelihood**

Idea

- Let's model the **class probabilities**

$$P(y = +1|x) = \widehat{f_\theta}(x)$$
$$P(y = -1|x) = 1 - \widehat{f_\theta}(x)$$

$$\text{Likelihood} = \prod_{i=1\ldots N} P(y_i|x_i)$$
$$= \prod_{i=1\ldots N} \left[\mathbb{I}[y_i = +1] \cdot \widehat{f_\theta}(x_i) + \mathbb{I}[y_i = -1] \cdot \left(1 - \widehat{f_\theta}(x_i)\right)\right]$$

- Fit with **maximum (log) likelihood**

Idea

- Let's model the **class probabilities**

$$P(y=+1|x)=\widehat{f_\theta}(x)$$
$$P(y=-1|x)=1-\widehat{f_\theta}(x)$$

$$\text{Likelihood}=\prod_{i=1\ldots N}P(y_i|x_i)$$
$$=\prod_{i=1\ldots N}\left[\mathbb{I}[y_i=+1]\cdot\widehat{f_\theta}(x_i)\right.$$
$$\left.+\,\mathbb{I}[y_i=-1]\cdot\left(1-\widehat{f_\theta}(x_i)\right)\right]$$

- Fit with **maximum (log) likelihood**

$$\theta=\underset{\theta}{\operatorname{argmax}}\sum_{i=1\ldots N}\left[\mathbb{I}[y_i=+1]\cdot\log\widehat{f_\theta}(x_i)+\mathbb{I}[y_i=-1]\cdot\log\left(1-\widehat{f_\theta}(x_i)\right)\right]$$

Idea

- Let's model the **class probabilities**

$$P(y=+1|x)=\widehat{f_\theta}(x)$$
$$P(y=-1|x)=1-\widehat{f_\theta}(x)$$

$$\text{Likelihood}=\prod_{i=1\ldots N}P(y_i|x_i)$$
$$=\prod_{i=1\ldots N}\Big[\mathbb{I}[y_i=+1]\cdot\widehat{f_\theta}(x_i)$$
$$+\,\mathbb{I}[y_i=-1]\cdot\left(1-\widehat{f_\theta}(x_i)\right)\Big]$$

- Fit with **maximum (log) likelihood**

$$\theta=\underset{\theta}{\operatorname{argmax}}\sum_{i=1\ldots N}\left[\mathbb{I}[y_i=+1]\cdot\log\widehat{f_\theta}(x_i)+\mathbb{I}[y_i=-1]\cdot\log\left(1-\widehat{f_\theta}(x_i)\right)\right]$$

- Predict the class with **highest probability***

*more generally: find a probability threshold suitable for your problem

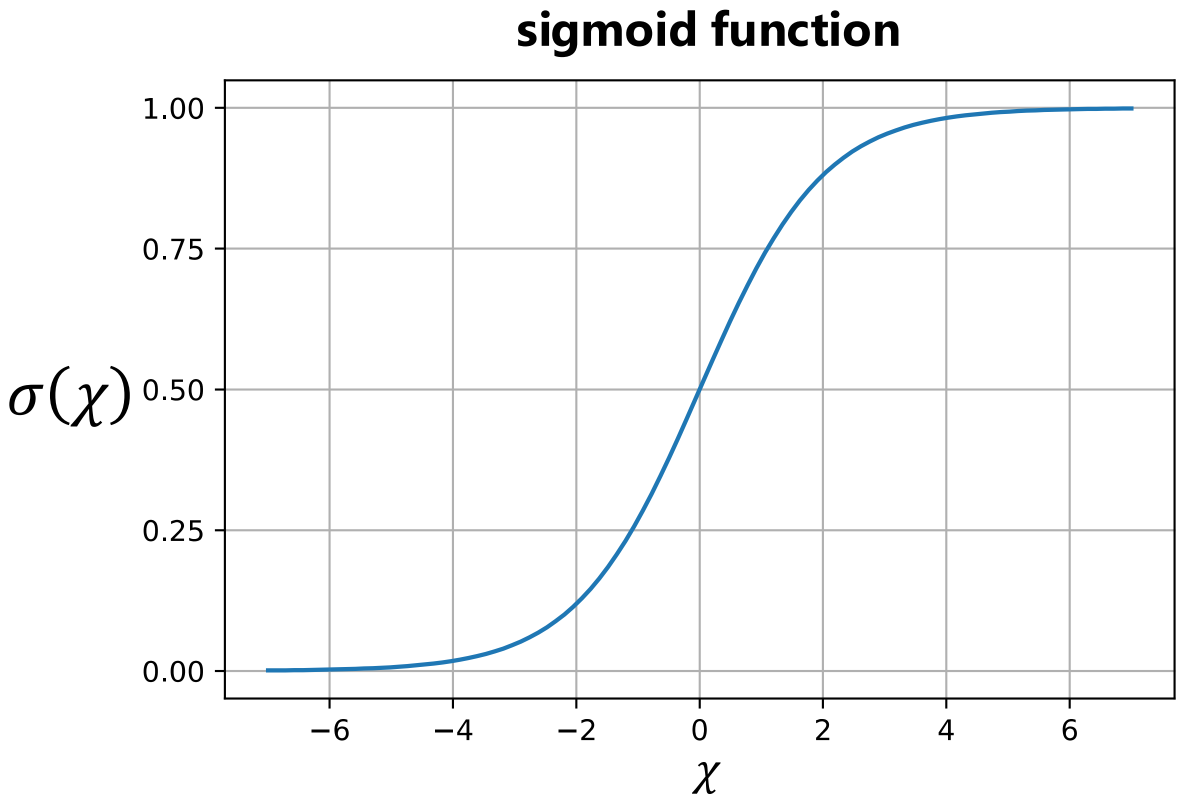
Linear probability model

- How to map the linear model output to a probability value in $[0, 1]$?

Linear probability model

- How to map the linear model output to a probability value in $[0, 1]$?
- Common choice – **sigmoid function**:

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

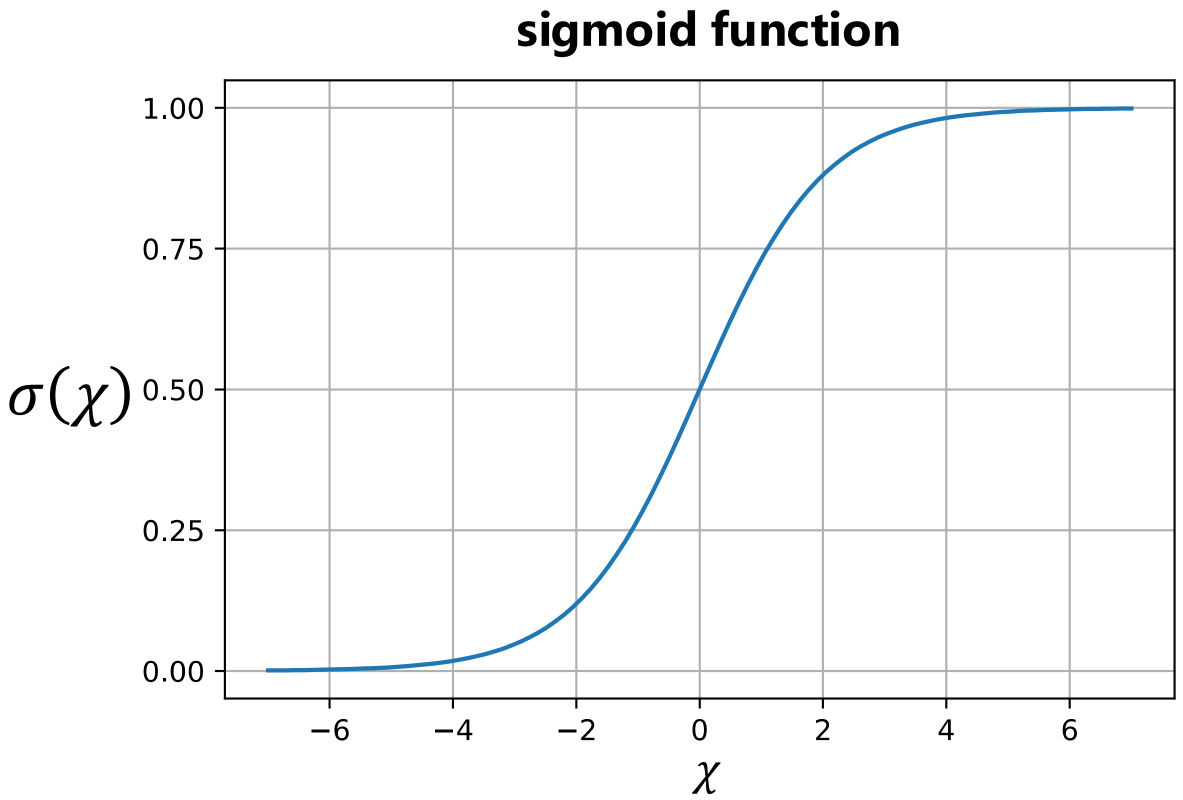


Linear probability model

- How to map the linear model output to a probability value in $[0, 1]$?
- Common choice – **sigmoid function**:

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

- I.e. $P(y = +1|x) = \sigma\left(\theta^{\mathrm{T}} x\right)$

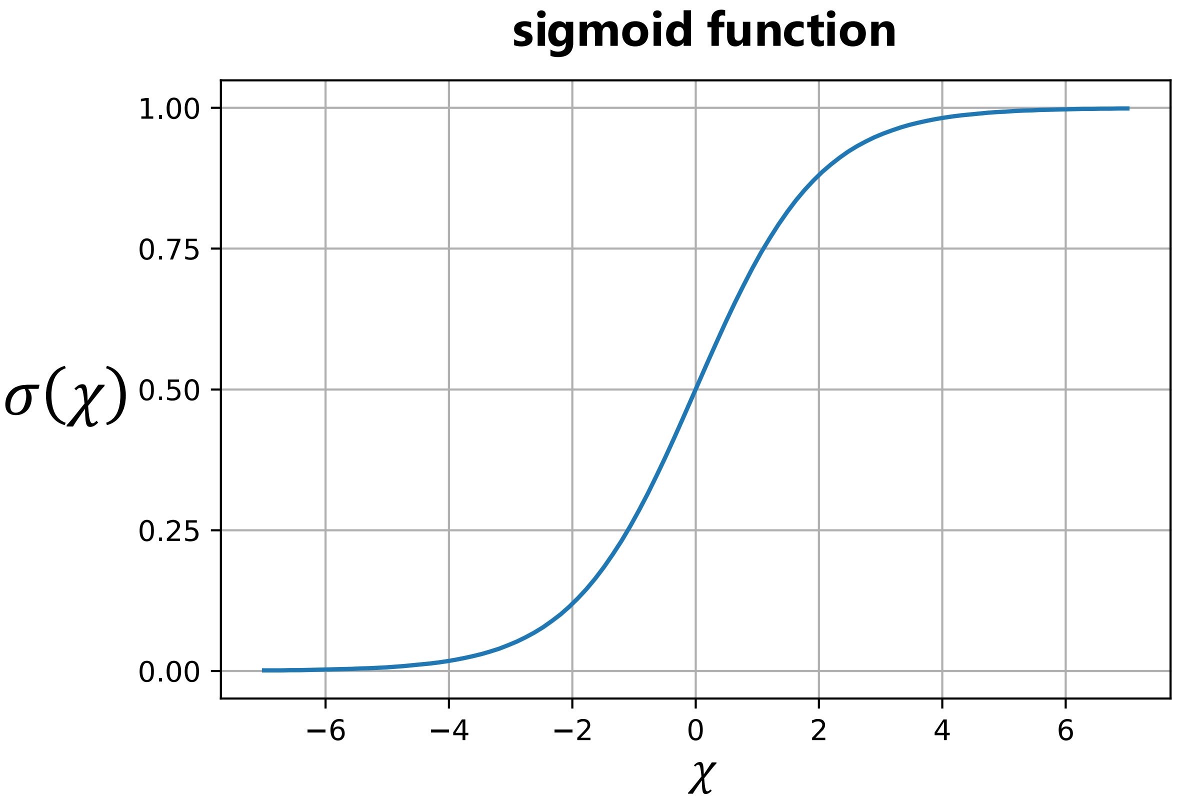


Linear probability model

- How to map the linear model output to a probability value in $[0, 1]$?
- Common choice – **sigmoid function**:

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

- I.e. $P(y = +1|x) = \sigma\left(\theta^{\mathrm{T}} x\right)$
- Then, $\theta^{\mathrm{T}} x$ has the meaning of **log odds ratio** between the two classes:



$$\log \frac{P(y = +1|x)}{P(y = -1|x)} = \log\left(\frac{1}{1 + e^{-\theta^{\mathrm{T}} x}} \cdot \frac{1 + e^{-\theta^{\mathrm{T}} x}}{e^{-\theta^{\mathrm{T}} x}}\right) = \theta^{\mathrm{T}} x$$

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathcal{L} = -\sum_{i=1\ldots N} \left[\mathbb{I}[y_i = +1] \cdot \log \widehat{f_\theta}(x_i) + \mathbb{I}[y_i = -1] \cdot \log\left(1 - \widehat{f_\theta}(x_i)\right) \right]$$

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathcal{L} = -\sum_{i=1\ldots N}\left[\mathbb{I}[y_i = +1] \cdot \log \widehat{f_\theta}(x_i) + \mathbb{I}[y_i = -1] \cdot \log\left(1 - \widehat{f_\theta}(x_i)\right)\right]$$

$$1 - \sigma(x) = \sigma(-x)$$

$$= -\sum_{i=1\ldots N}\left[\mathbb{I}[y_i = +1] \cdot \log \sigma\left(\theta^{\mathrm{T}} x_i\right) + \mathbb{I}[y_i = -1] \cdot \log \sigma\left(-\theta^{\mathrm{T}} x_i\right)\right]$$

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathcal{L} = -\sum_{i\,=\,1\ldots N}\left[\mathbb{I}[y_i = +1] \cdot \log \widehat{f_\theta}(x_i) + \mathbb{I}[y_i = -1] \cdot \log\left(1 - \widehat{f_\theta}(x_i)\right)\right]$$

$1 - \sigma(x) = \sigma(-x)$

$$= -\sum_{i\,=\,1\ldots N}\left[\mathbb{I}[y_i = +1] \cdot \log \sigma\left(\theta^{\mathrm{T}} x_i\right) + \mathbb{I}[y_i = -1] \cdot \log \sigma\left(-\theta^{\mathrm{T}} x_i\right)\right]$$

$$= -\sum_{i\,=\,1\ldots N} \log \sigma\left(\theta^{\mathrm{T}} x_i \cdot y_i\right)$$

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathcal{L} = -\sum_{i\,=\,1\ldots N}\left[\mathbb{I}[y_i = +1]\cdot\log\widehat{f_\theta}(x_i) + \mathbb{I}[y_i = -1]\cdot\log\left(1 - \widehat{f_\theta}(x_i)\right)\right]$$

$1 - \sigma(x) = \sigma(-x)$

$$= -\sum_{i\,=\,1\ldots N}\left[\mathbb{I}[y_i = +1]\cdot\log\sigma\left(\theta^{\mathrm{T}}x_i\right) + \mathbb{I}[y_i = -1]\cdot\log\sigma\left(-\theta^{\mathrm{T}}x_i\right)\right]$$

$$= -\sum_{i\,=\,1\ldots N}\log\sigma\left(\theta^{\mathrm{T}}x_i\cdot y_i\right) = \sum_{i\,=1\ldots N}\log\left(1 + e^{-\theta^{\mathrm{T}}x_i\cdot y_i}\right)$$

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathcal{L} = -\sum_{i=1\ldots N} \left[\mathbb{I}[y_i = +1] \cdot \log \widehat{f_\theta}(x_i) + \mathbb{I}[y_i = -1] \cdot \log\left(1 - \widehat{f_\theta}(x_i)\right) \right]$$

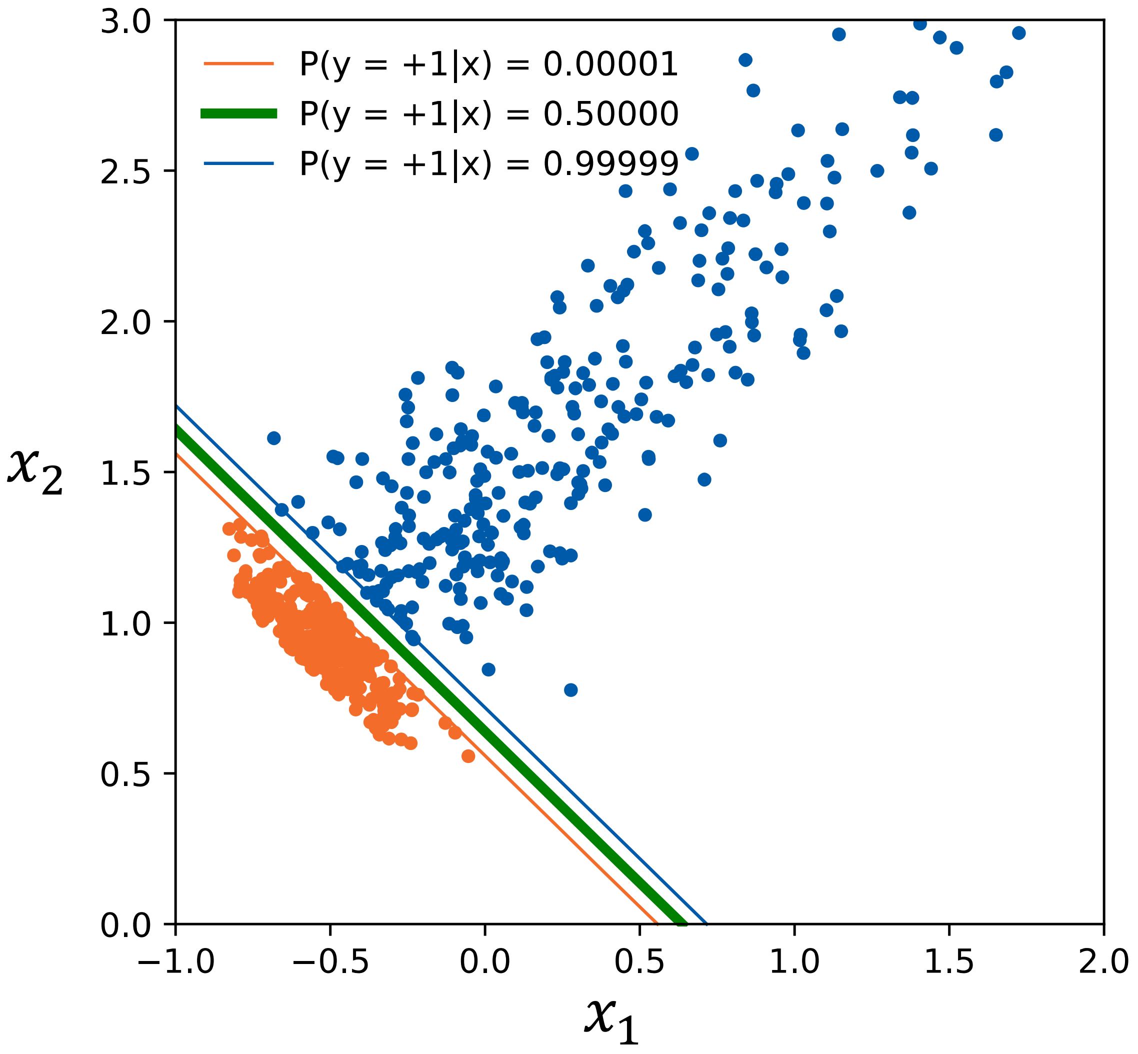
$1 - \sigma(x) = \sigma(-x)$

$$= -\sum_{i=1\ldots N} \left[\mathbb{I}[y_i = +1] \cdot \log \sigma\left(\theta^{\mathrm{T}} x_i\right) + \mathbb{I}[y_i = -1] \cdot \log \sigma\left(-\theta^{\mathrm{T}} x_i\right) \right]$$

$$= -\sum_{i=1\ldots N} \log \sigma\left(\theta^{\mathrm{T}} x_i \cdot y_i\right) = \sum_{i=1\ldots N} \log\left(1 + e^{-\theta^{\mathrm{T}} x_i \cdot y_i}\right)$$

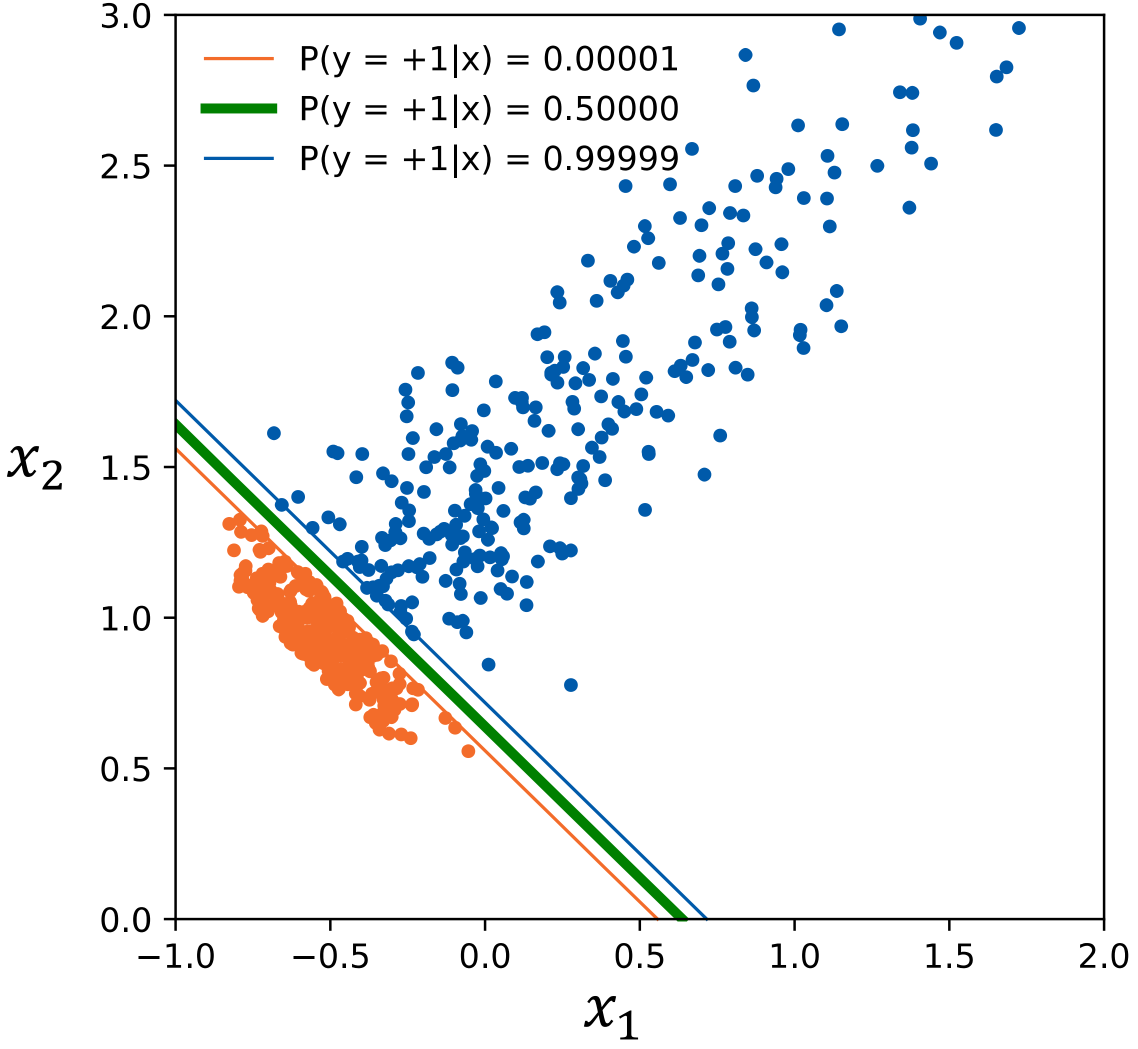
- This can be optimized **numerically**

Example



- Now the boundary is at the right place

Example

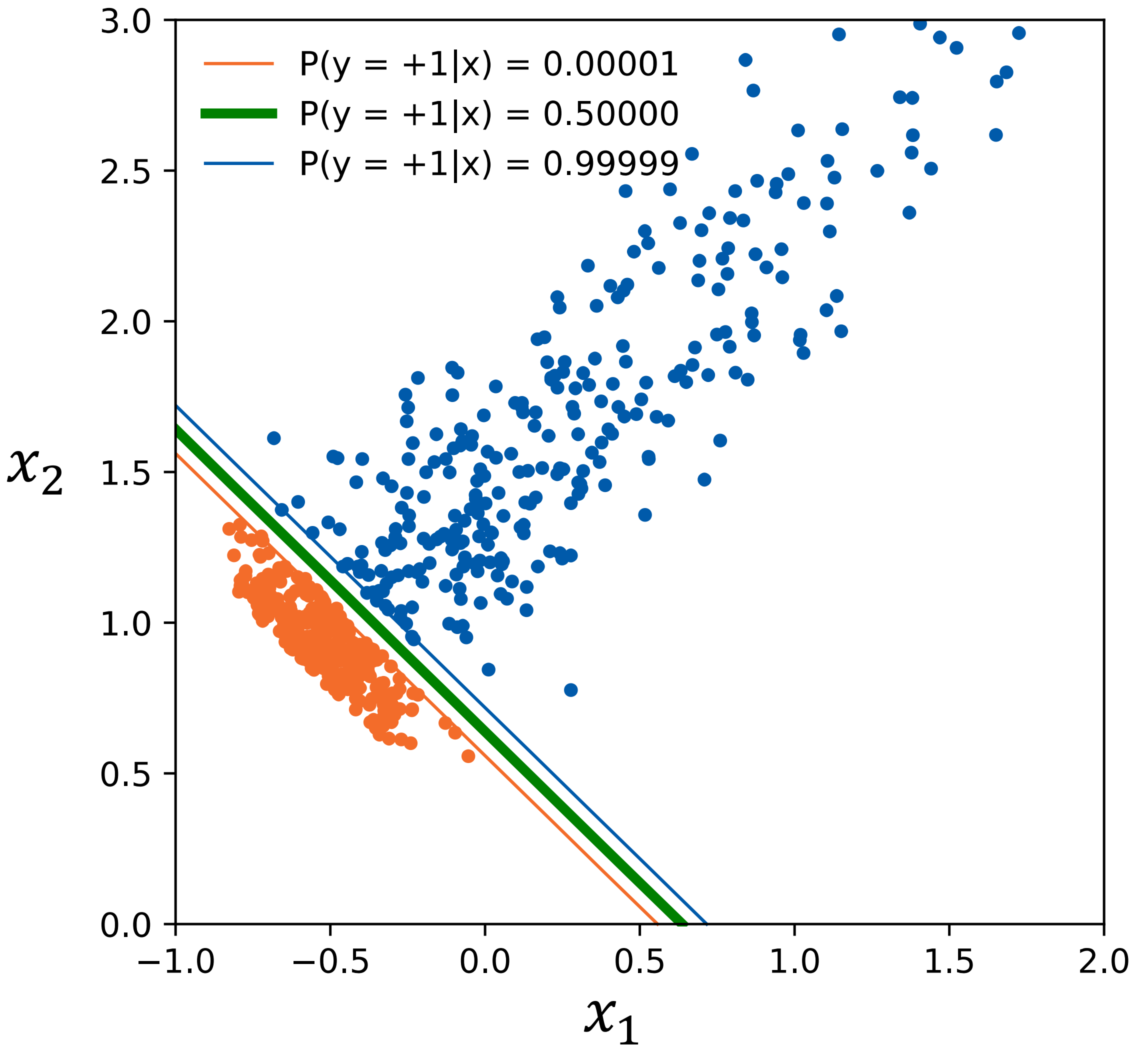


- Now the boundary is at the right place
- Note: when classes are linearly separable for any correct decision boundary

$$\theta \longrightarrow C \cdot \theta, \text{ for some } C > 1 \in \mathbb{R}$$

keeps the boundary at the same place, yet improves the loss

Example



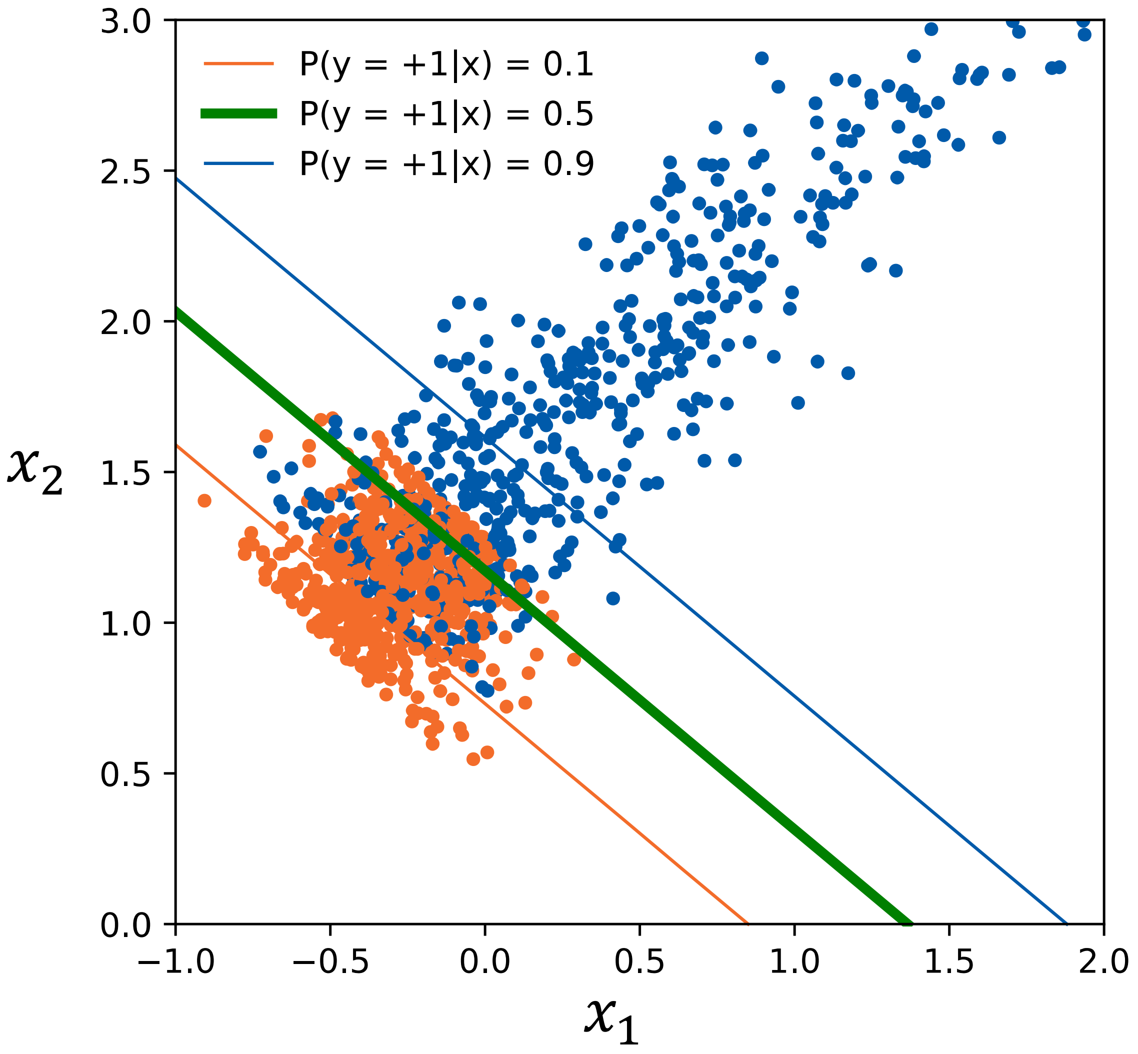
- Now the boundary is at the right place
- Note: when classes are linearly separable for any correct decision boundary

$$\theta \longrightarrow C \cdot \theta, \text{ for some } C > 1 \in \mathbb{R}$$

  keeps the boundary at the same place, yet improves the loss

- ideal fit when sigmoid turns into a step function (at infinitely large θ)

Example



- When classes overlap the loss has a finite minimum
- Predicted class probability changes smoothly

Multiclass Logistic Regression

Multinomial Logistic Regression

- Similarly to the binary case, we'll model the class probabilities
- Let's model **unnormalized** class probabilities like this:

$$\tilde{P}(y = k|x) = \exp \theta_k^{\mathrm{T}} x$$

Note: now we have K parameter vectors

Multinomial Logistic Regression

- Similarly to the binary case, we'll model the class probabilities
- Let's model **unnormalized** class probabilities like this:

Note: now we have K parameter vectors

$$\tilde{P}(y = k|x) = \exp \theta_k^{\mathrm{T}} x$$

- Then, the **normalized** probabilities are:

$$P(y = k|x) = \frac{\tilde{P}(y = k|x)}{\sum_{k'=1\dots K} \tilde{P}(y = k'|x)} = \frac{\exp \theta_k^{\mathrm{T}} x}{\sum_{k'=1\dots K} \exp \theta_{k'}^{\mathrm{T}} x}$$

  – This function is called **softmax** and is commonly used in neural networks

Multinomial Logistic Regression

- Plugging everything into the negative log likelihood we get our loss function:

$$\mathcal{L} = -\sum_{i=1\dots N} \log \frac{\exp \theta_{y_i}^{\mathrm{T}} x_i}{1 + \sum_{k'=1\dots K-1} \exp \theta_{k'}^{\mathrm{T}} x_i}$$

$$(\theta_K = 0)$$

- Again, this can be optimized **numerically**

Multiclass classification: general approach

General idea

For a problem with K classes introduce K predictors:

$$\hat{f}_k(x)\colon \mathcal{X} \to \mathbb{R}, \text{ for } \mathrm{k} = 1, \dots, K$$

each of which outputs a corresponding **class score**.

Predict the class with the **highest score**:

$$\hat{y}_i = \underset{k}{\operatorname{argmax}}\, \hat{f}_k(x_i)$$

Example: binary → multiclass

- Any binary linear classification model can be converted to multiclass with **one-vs-rest** strategy

Example: binary → multiclass

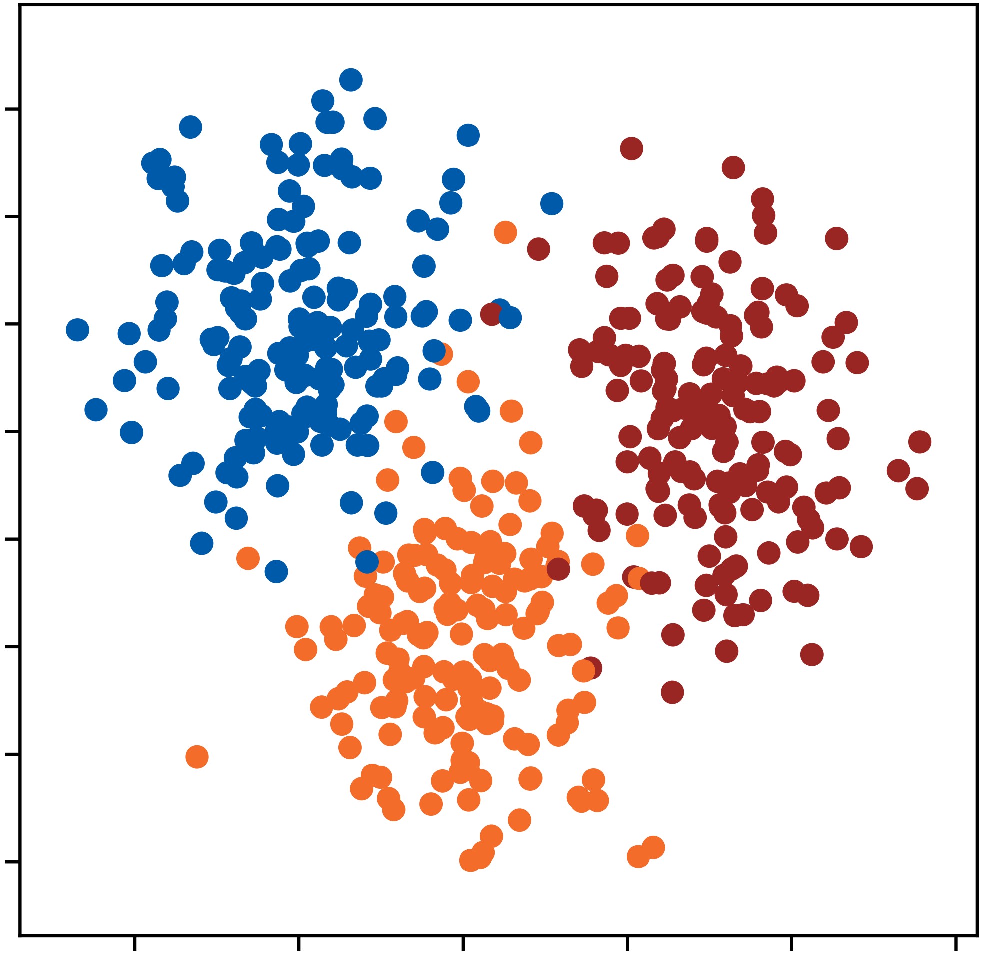
- Any binary linear classification model can be converted to multiclass with **one-vs-rest** strategy
- For each class k train a binary model $\widehat{f_k}(x) = \theta_{(k)}^{\mathrm{T}} x$ separating the given class from all others, $\hat{y}_{(k)}^{1-\mathrm{vs}-\mathrm{rest}} = \mathrm{sign}\left[\widehat{f_k}(x)\right]$

Example: binary → multiclass

- Any binary linear classification model can be converted to multiclass with **one-vs-rest** strategy
- For each class k train a binary model $\widehat{f}_k(x) = \theta_{(k)}^{\mathrm{T}} x$ separating the given class from all others, $\hat{y}_{(k)}^{1-\mathrm{vs}-\mathrm{rest}} = \mathrm{sign}\big[\widehat{f}_k(x)\big]$
- Use the outputs of $\widehat{f}_k$ as class scores for multiclass classification:

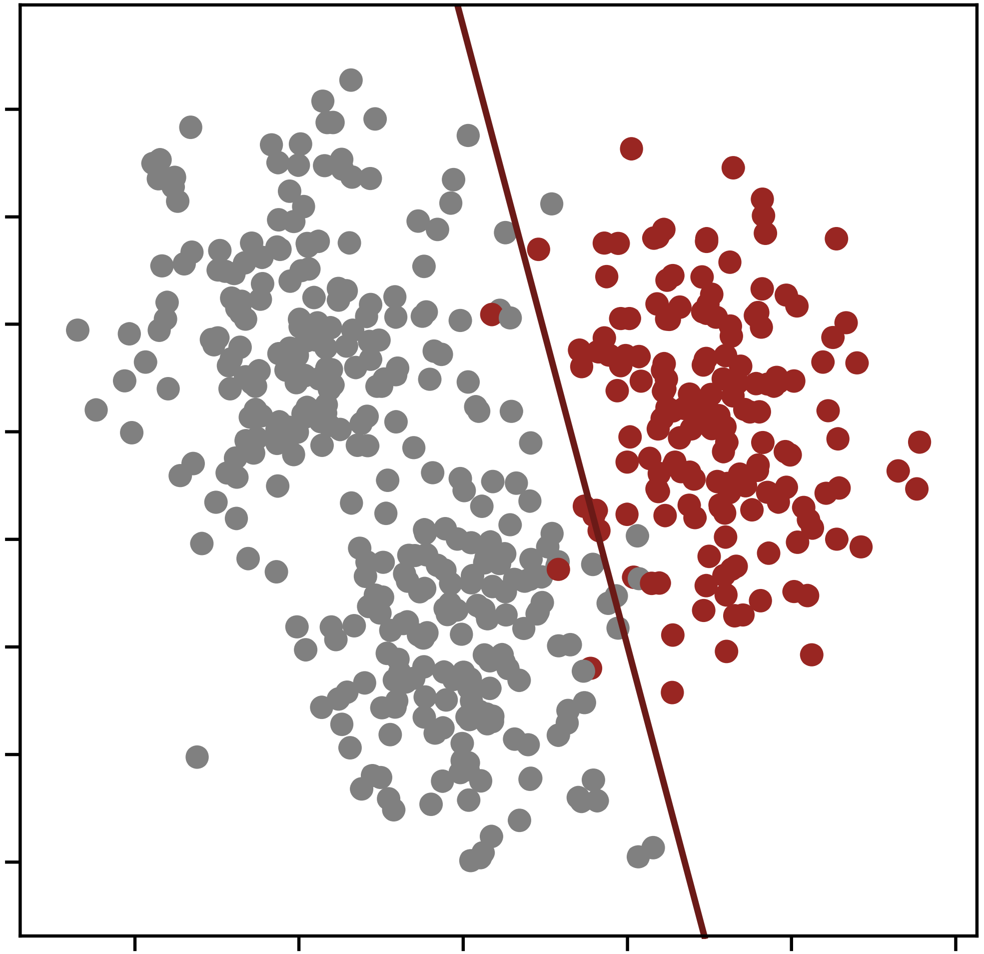
$$\hat{y}_i = \underset{k}{\mathrm{argmax}}\, \widehat{f}_k(x_i)$$

Example



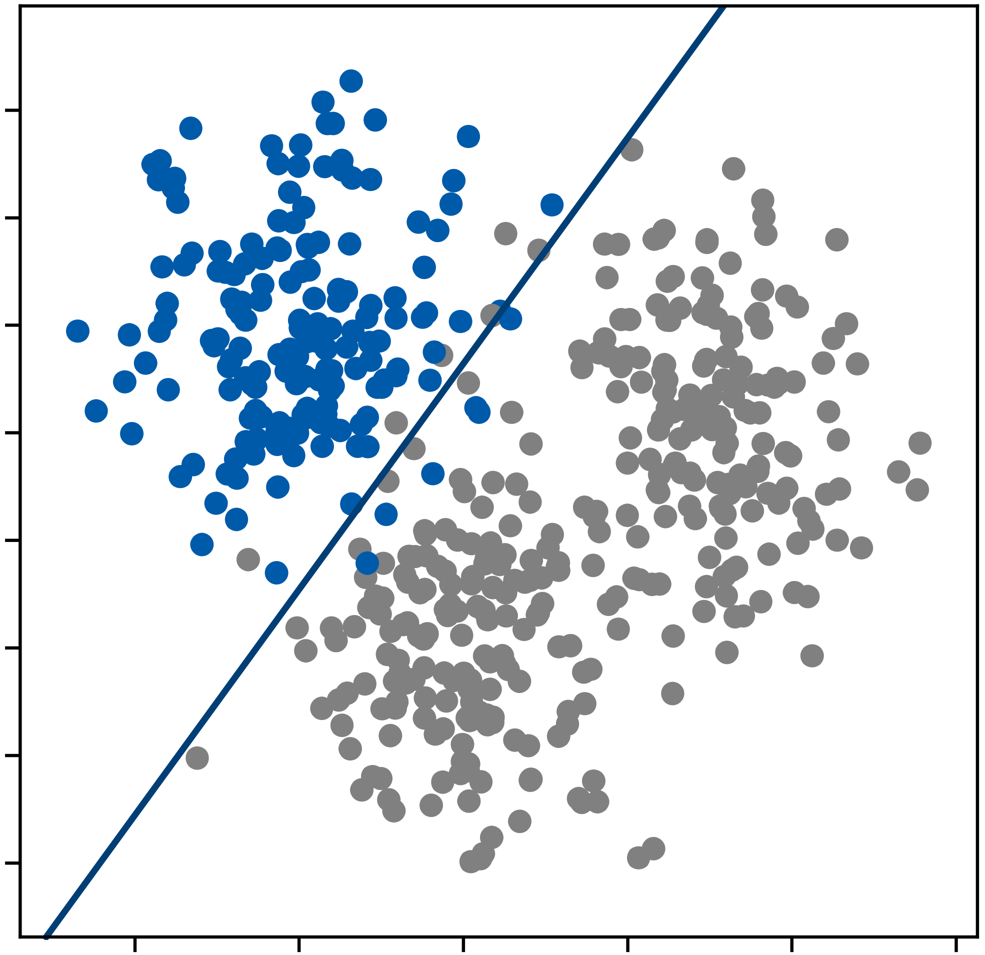
- Consider the following 3 class problem

Example



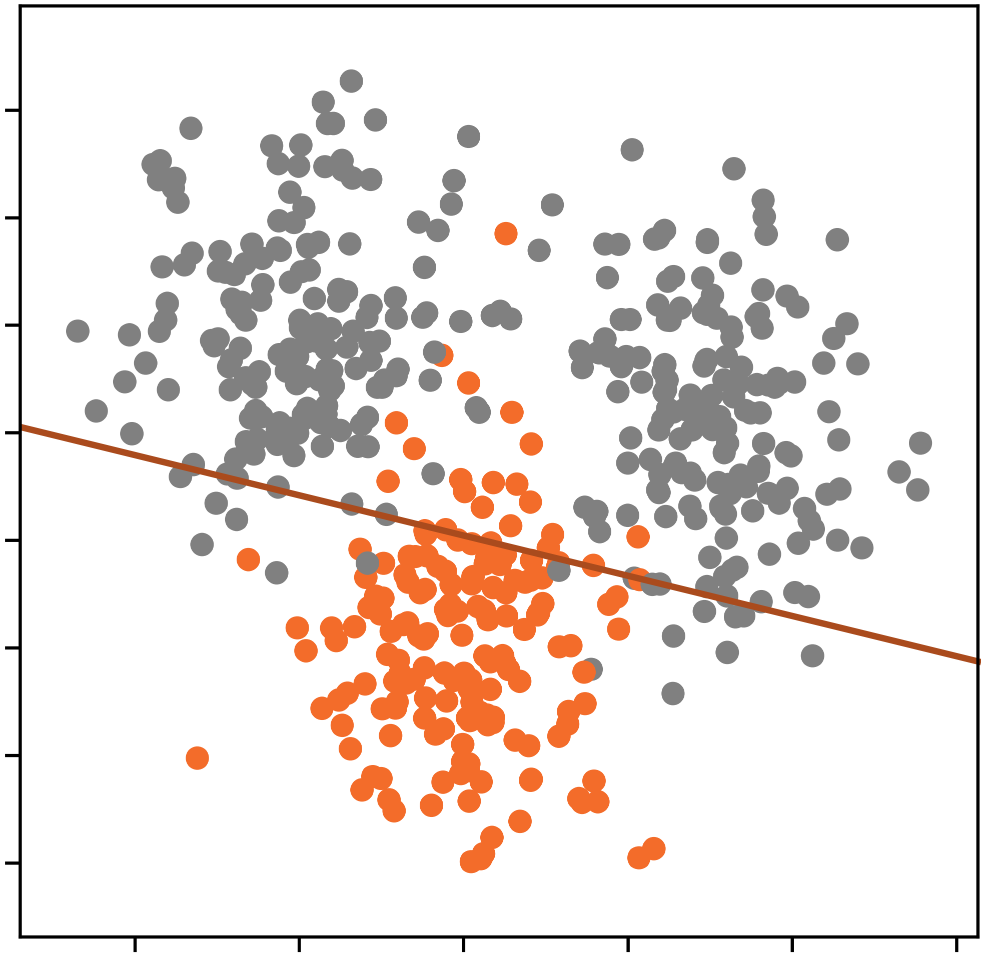
- “Class-1 VS rest” binary classifier

Example



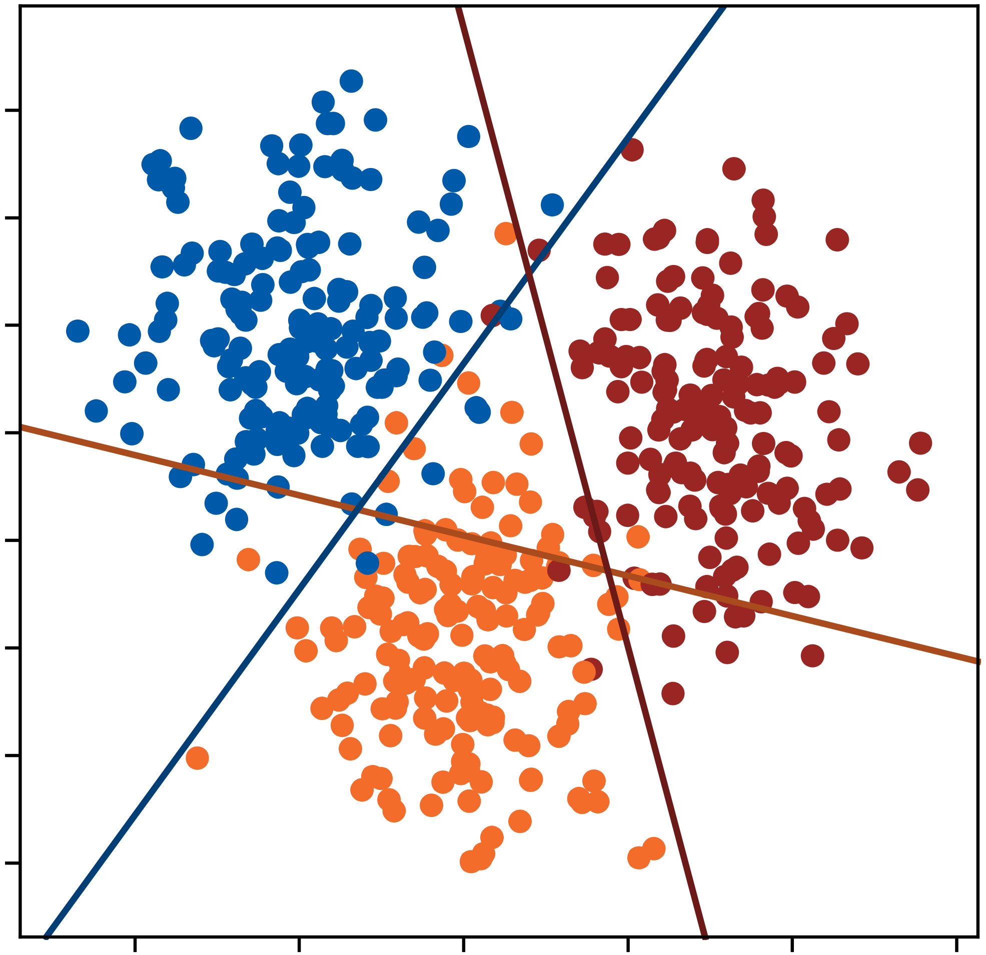
- "Class-2 VS rest" binary classifier

Example



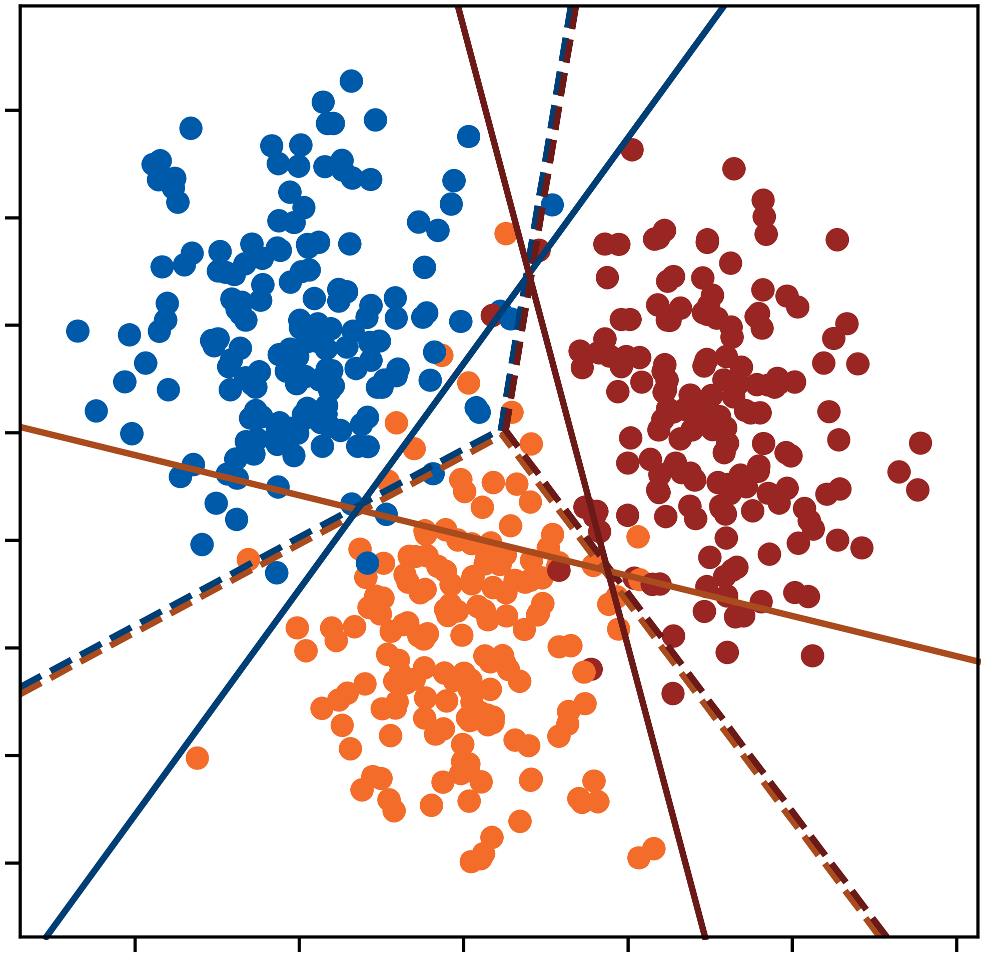
- "Class-3 VS rest" binary classifier

Example



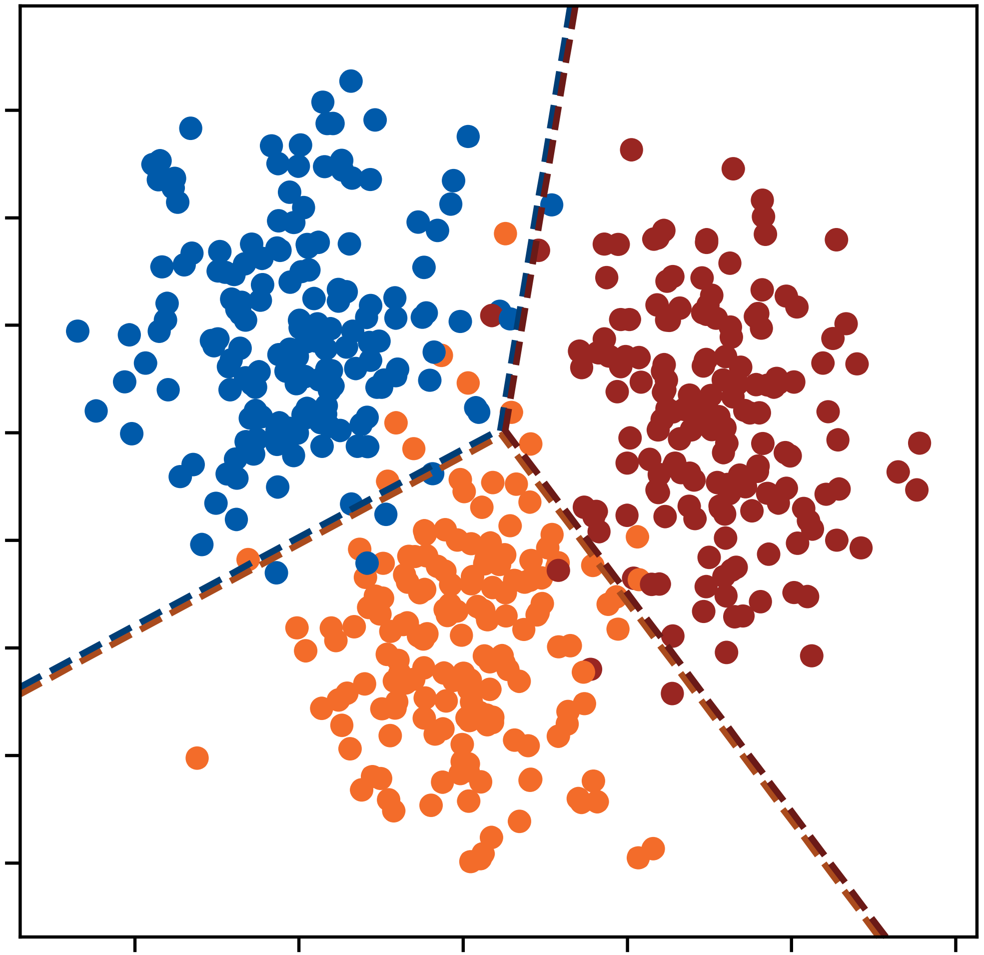
- $\widehat{f_k}(x) = 0$ lines (binary decision boundaries)

Example



- $\widehat{f_k}(x) = 0$ lines (binary decision boundaries)
- Adding decision boundaries for $\hat{y} = \underset{k}{\operatorname{argmax}} \widehat{f_k}(x)$

Example



- Adding decision boundaries for $\hat{y} = \underset{k}{\operatorname{argmax}} \widehat{f_k}(x)$

Summary

- Classification with linear regression and MSE loss may provide **biased results**
- 0-1 loss function is better, but is **hard to optimize** directly
- Various **differentiable upper bounds** on 0-1 loss may be used instead
- Logistic Regression combines such an upper bound with a **probabilistic model** using the **sigmoid function**
- Generalizing sigmoid function to a multiclass case yields **softmax function**
- Any binary linear classifier can be adapted to multiclass with the **one-vs-rest strategy**
- Food for thought: how can you mitigate the biased probability problems when using one-vs-rest strategy (as discussed on the previous slide)?

Thank you!

Majid Sohrabi

msohrabi@hse.ru