Classification with Linear Models

Losses for linear classification, logistic regression, multiclass classification

Machine Learning and Data Mining, 2023

Presented by: Abdalaziz Al-Maeeni

Prepared by: Artem Maevskiy

National Research University Higher School of Economics

LAMBDA • HSE

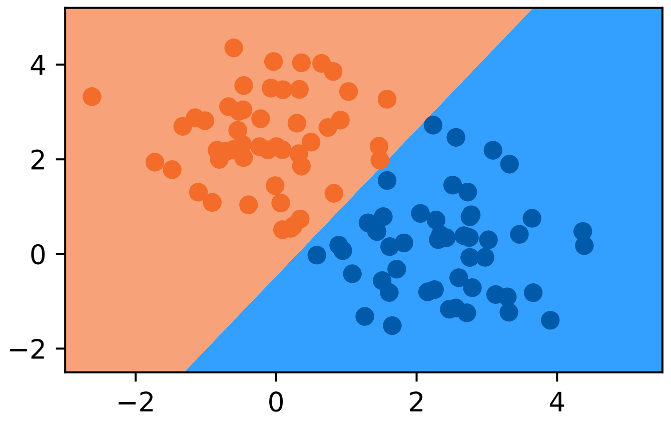
September 30, 2023

Can’t we just use linear regression for classification?

Classification with linear regression

Classification:

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

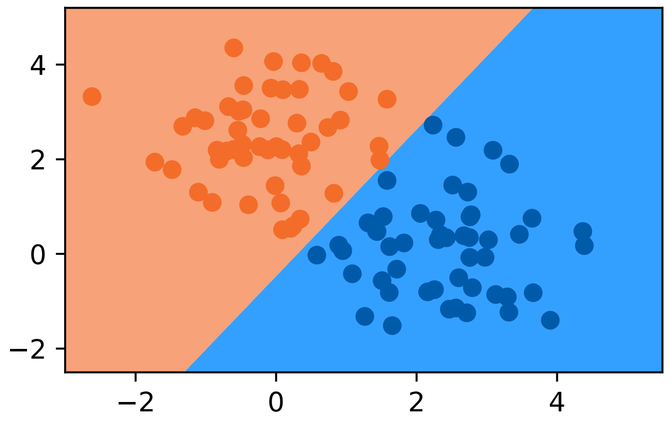


- For binary classification task, assign:
  - $y = +1$ for positive class
  - $y = -1$ for negative class

Classification with linear regression

Classification:

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

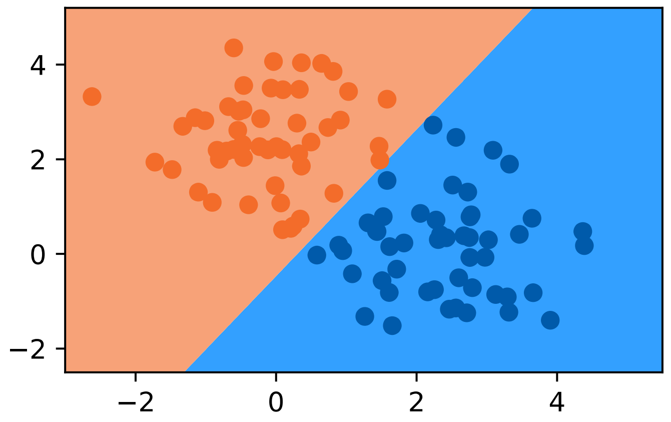


- For binary classification task, assign:
  - $y = +1$ for positive class
  - $y = -1$ for negative class
- Solve linear regression for $\hat{y} = \theta^T x$ with MSE loss

Classification with linear regression

Classification:

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

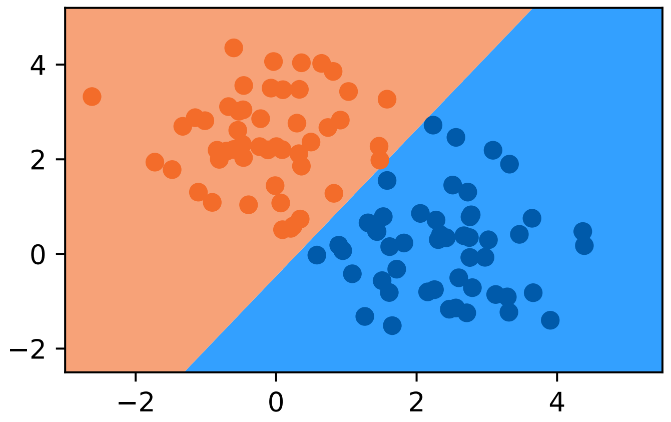


- For binary classification task, assign:
  - $y = +1$ for positive class
  - $y = -1$ for negative class
- Solve linear regression for $\hat{y} = \theta^T x$ with MSE loss
- Classify with $\operatorname{sign}\left[\hat{y}\right]$

Classification with linear regression

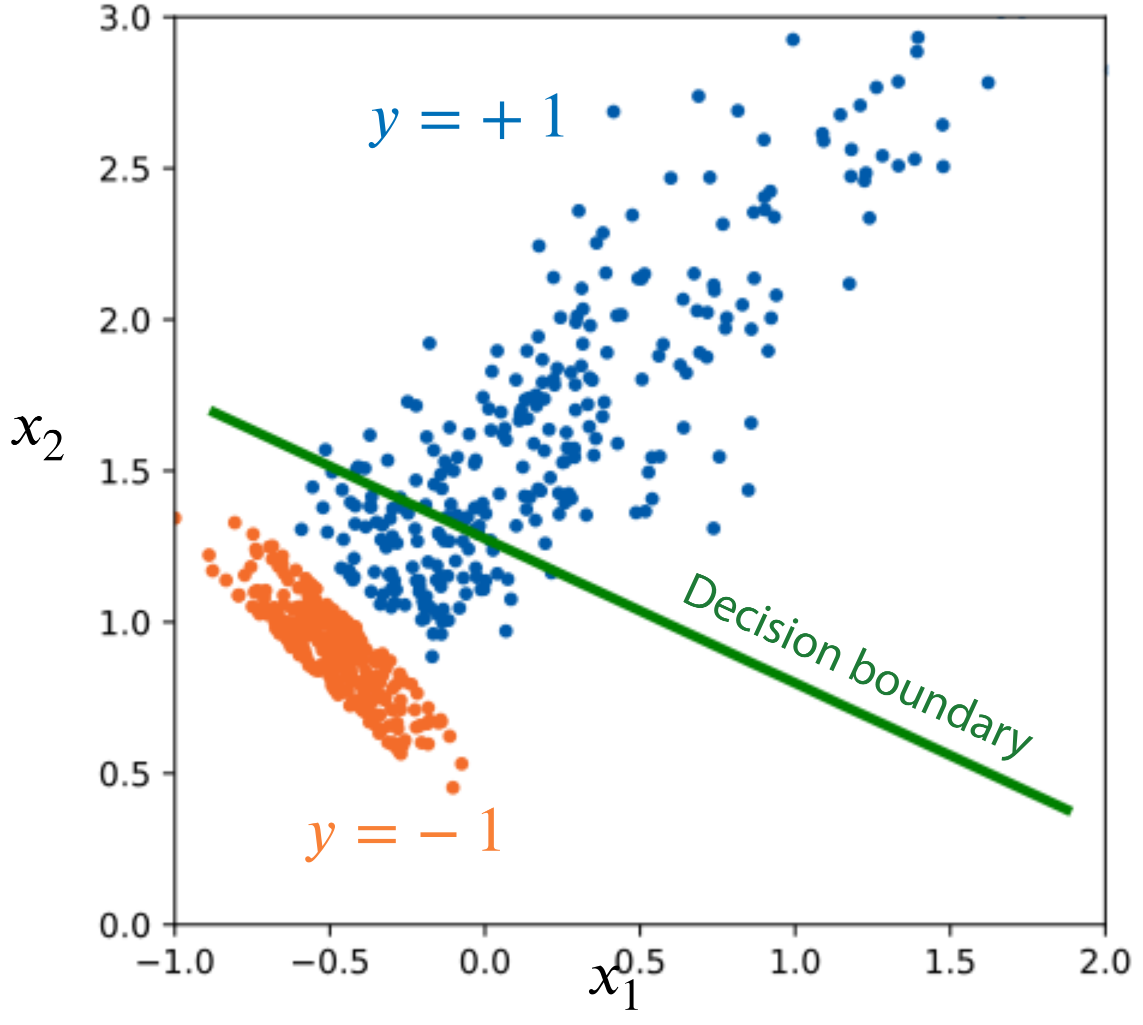
Classification:

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



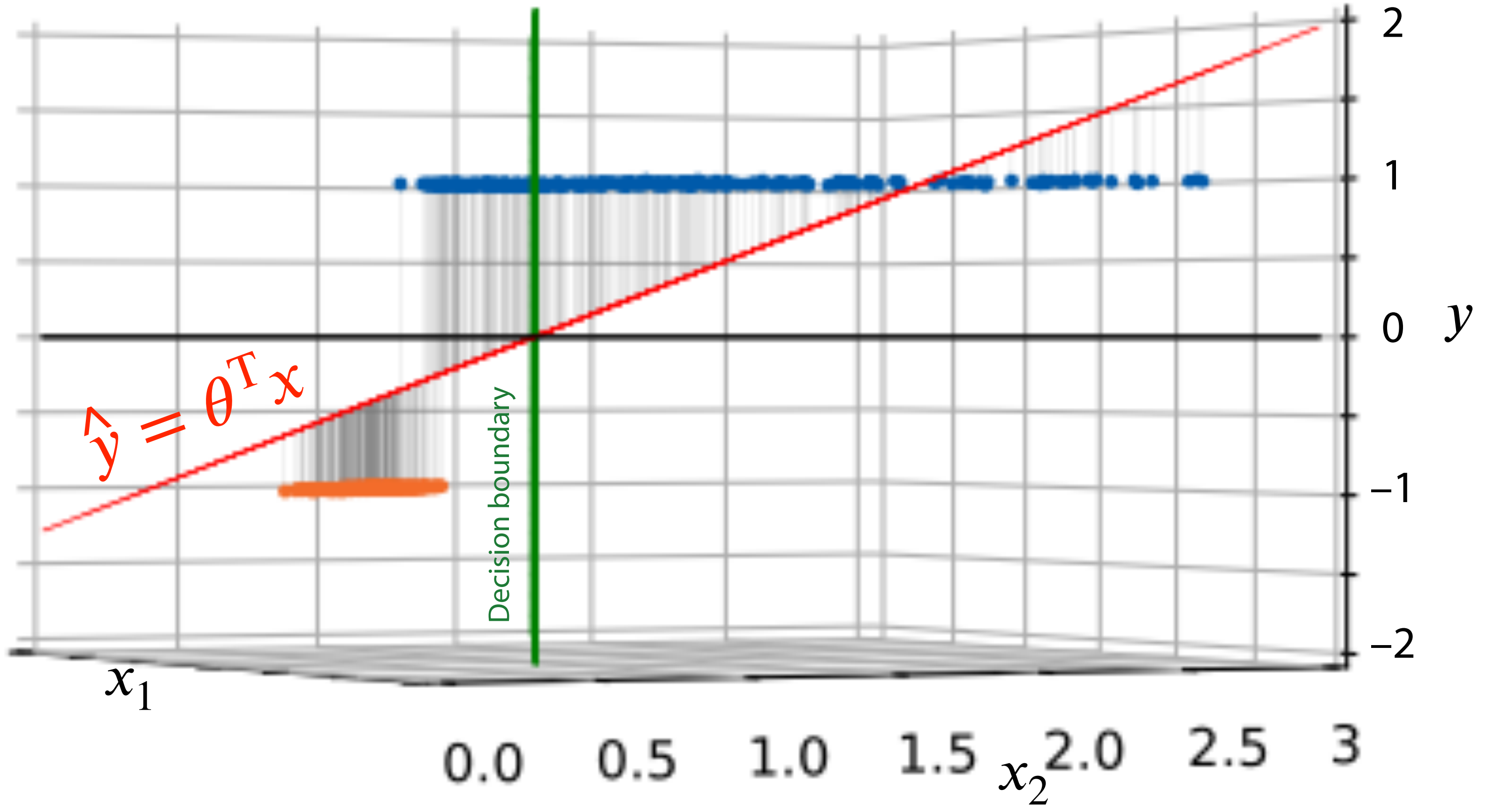
- For binary classification task, assign:
  - $y = +1$ for positive class
  - $y = -1$ for negative class
- Solve linear regression for $\hat{y} = \theta^T x$ with MSE loss
- Classify with $\text{sign}\left[\hat{y}\right]$
- Any problems with this approach?

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?

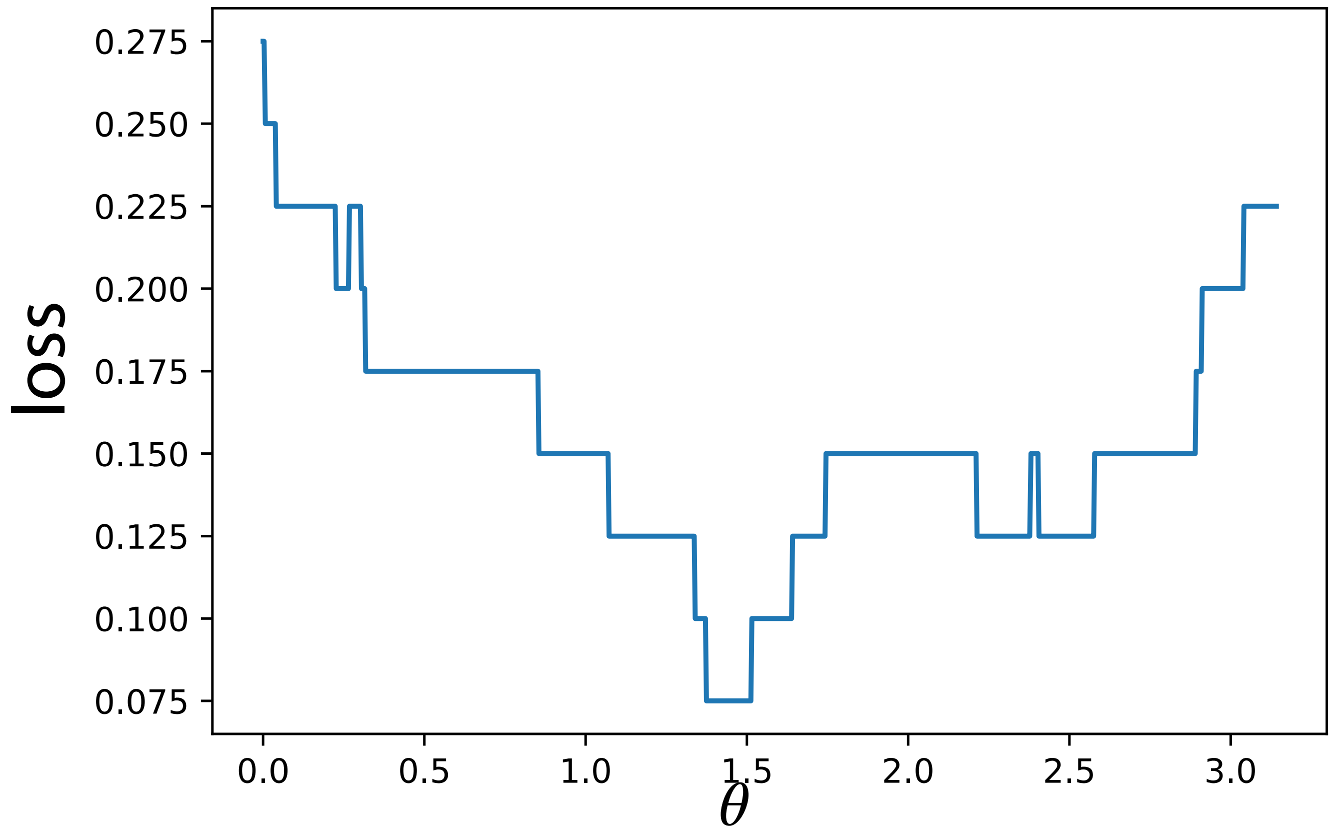
Classification loss functions

0-1 Loss

- Probability of an error

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

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

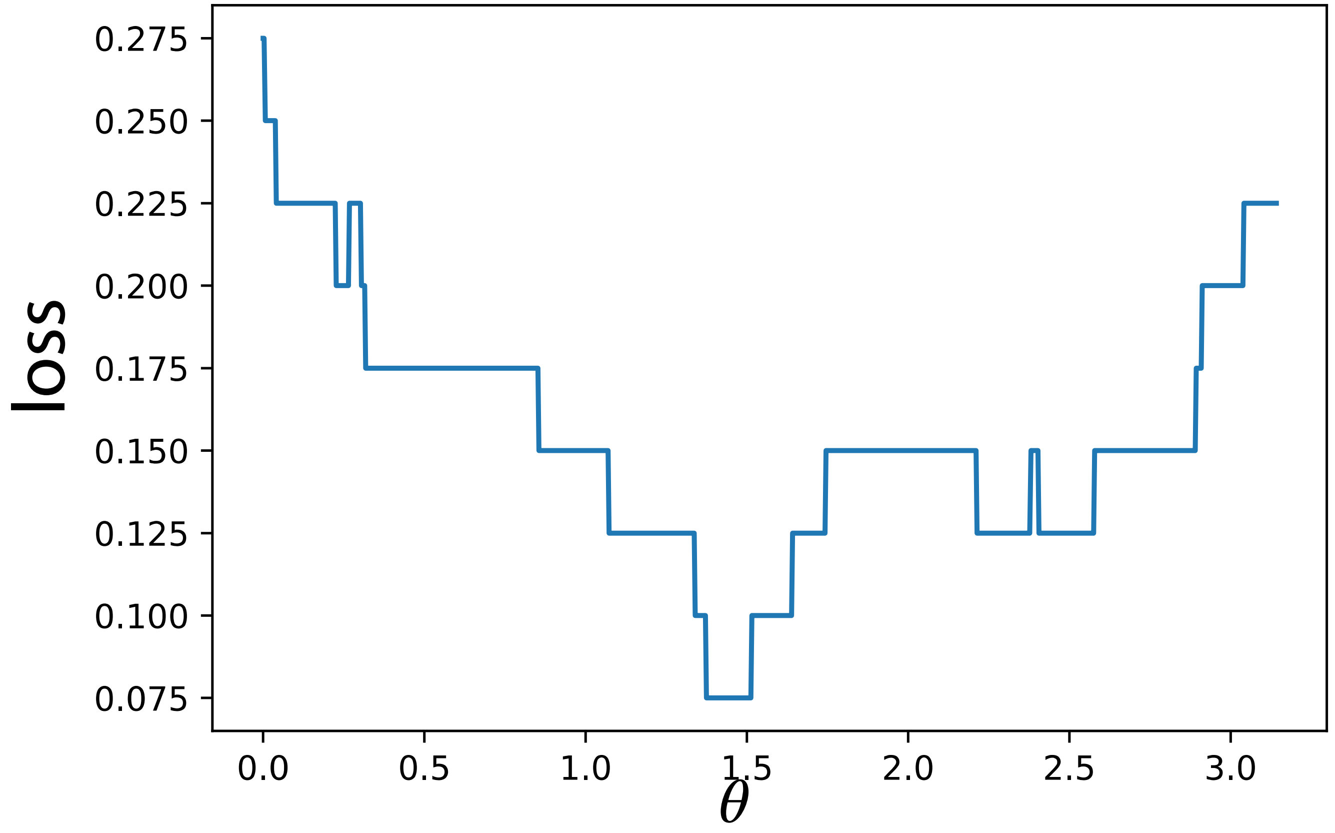


0-1 Loss

- Probability of an error

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

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



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

*other techniques exist (still quite limited)

Margin

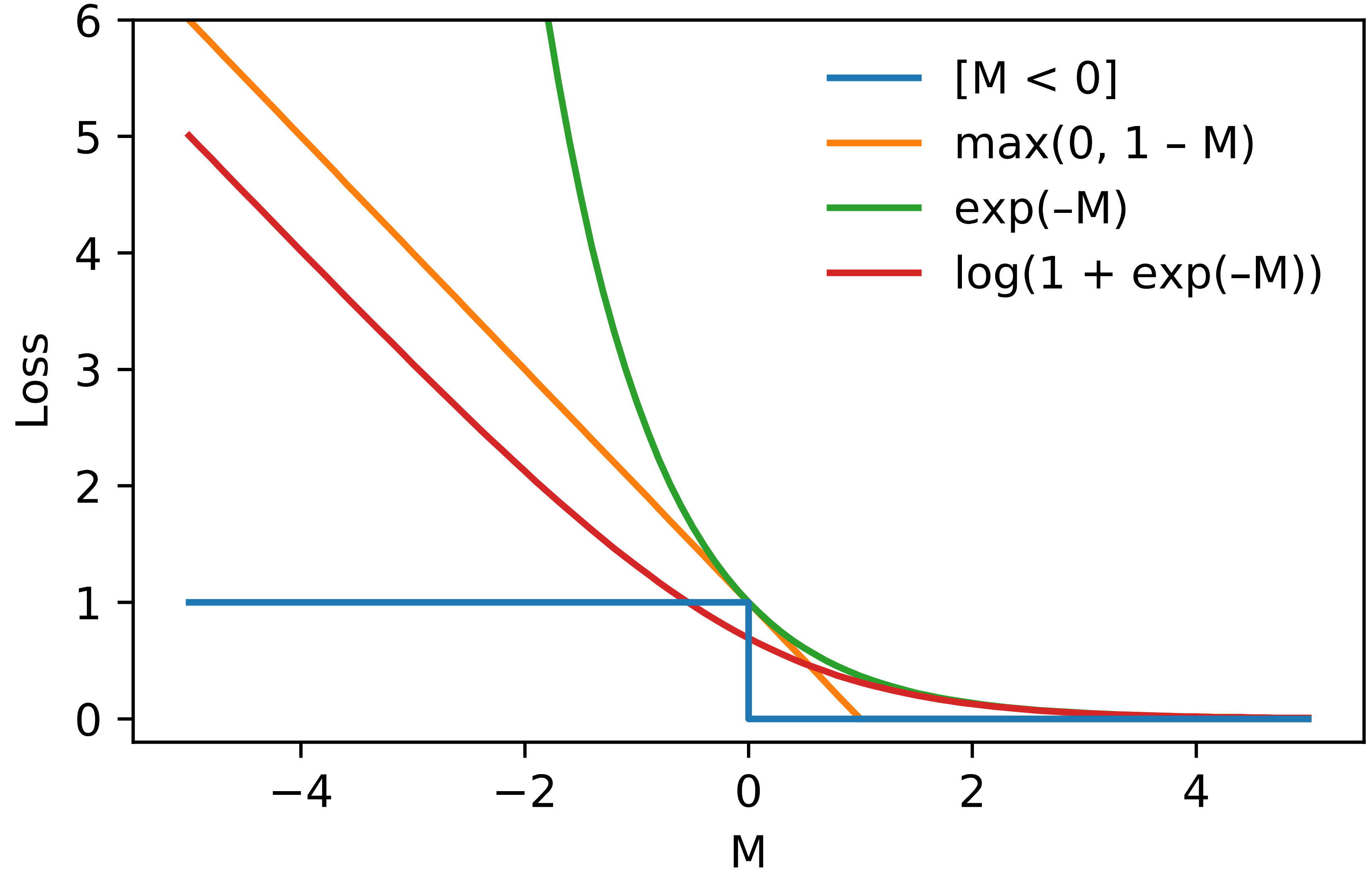
$$M = \theta^{\mathrm{T}} x \cdot y$$

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

$M > 0$ – 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

$$P\left(y=+1 \,\middle|\, x\right)=\hat{f}_\theta(x)$$
$$P\left(y=-1 \,\middle|\, x\right)=1-\hat{f}_\theta(x)$$

Idea

- Let’s model the class probabilities

$$P\left(y = +1 \middle| x\right) = \hat{f}_\theta(x)$$
$$P\left(y = -1 \middle| x\right) = 1 - \hat{f}_\theta(x)$$

- Fit with maximum (log) likelihood

Idea

$$\text{Likelihood} = \prod_{i=1\ldots N} P\left(y_i \; x_i\right)$$

- Let's model the class probabilities

$$P\left(y = +1 \middle| x\right) = \hat{f}_\theta(x)$$
$$P\left(y = -1 \middle| x\right) = 1 - \hat{f}_\theta(x)$$

- Fit with maximum (log) likelihood

Idea

- Let’s model the class probabilities

$$P\left(y = +1 \,\middle|\, x\right) = \hat{f}_\theta(x)$$
$$P\left(y = -1 \,\middle|\, x\right) = 1 - \hat{f}_\theta(x)$$

$$\text{Likelihood} = \prod_{i=1\ldots N} P\left(y_i \;\; x_i\right)$$
$$= \prod_{i=1\ldots N} \left[\mathbb{1}\left[y_i = +1\right] \cdot \hat{f}_\theta\left(x_i\right)\right.$$
$$\left. + \mathbb{1}\left[y_i = -1\right] \cdot \left(1 - \hat{f}_\theta\left(x_i\right)\right)\right]$$

- Fit with maximum (log) likelihood

Idea

- Let's model the class probabilities

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

$$= \prod_{i=1\ldots N} \Big[\mathbb{1}[y_i = +1] \cdot \hat{f}_\theta(x_i)$$

$$+ \mathbb{1}[y_i = -1] \cdot \big(1 - \hat{f}_\theta(x_i)\big) \Big]$$

$$P\big(y = +1 \,\big|\, x\big) = \hat{f}_\theta(x)$$
$$P\big(y = -1 \,\big|\, x\big) = 1 - \hat{f}_\theta(x)$$

- Fit with maximum (log) likelihood

$$\theta = \underset{\theta}{\operatorname{argmax}} \sum_{i=1\ldots N} \Big[\mathbb{1}[y_i = +1] \cdot \log \hat{f}_\theta(x_i) + \mathbb{1}[y_i = -1] \cdot \log\big(1 - \hat{f}_\theta(x_i)\big) \Big]$$

Idea

- Let's model the class probabilities

$$\text{Likelihood} = \prod_{i=1\ldots N} P\left(y_i \;\; x_i\right)$$

$$= \prod_{i=1\ldots N} \Big[\mathbb{1}\left[y_i = +1\right] \cdot \hat{f}_\theta\left(x_i\right)$$

$$+\mathbb{1}\left[y_i = -1\right] \cdot \left(1 - \hat{f}_\theta\left(x_i\right)\right)\Big]$$

$$P\left(y = +1 \,\middle|\, x\right) = \hat{f}_\theta(x)$$
$$P\left(y = -1 \,\middle|\, x\right) = 1 - \hat{f}_\theta(x)$$

- Fit with maximum (log) likelihood

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

- Predict the class with highest probability*

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

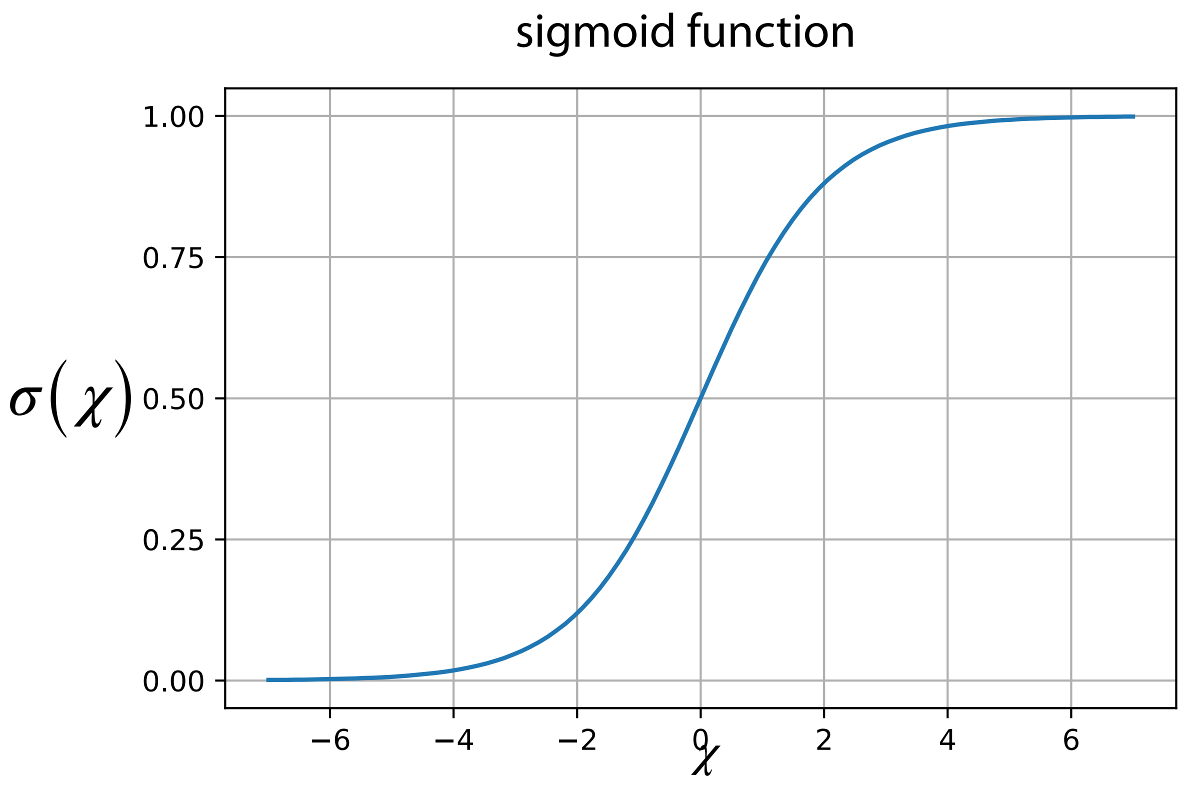
Linear probability model

- 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:

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



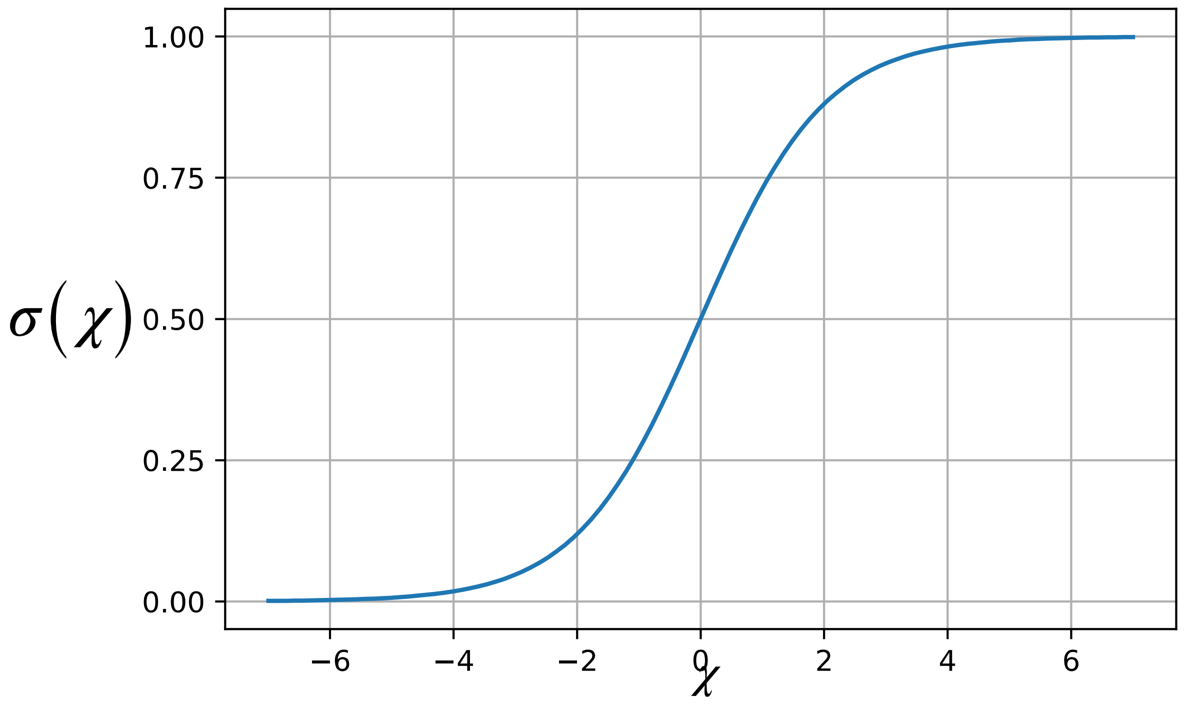
Linear probability model

- 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. $P(y = +1 \mid x) = \sigma(\theta^{\mathrm{T}} x)$

sigmoid function

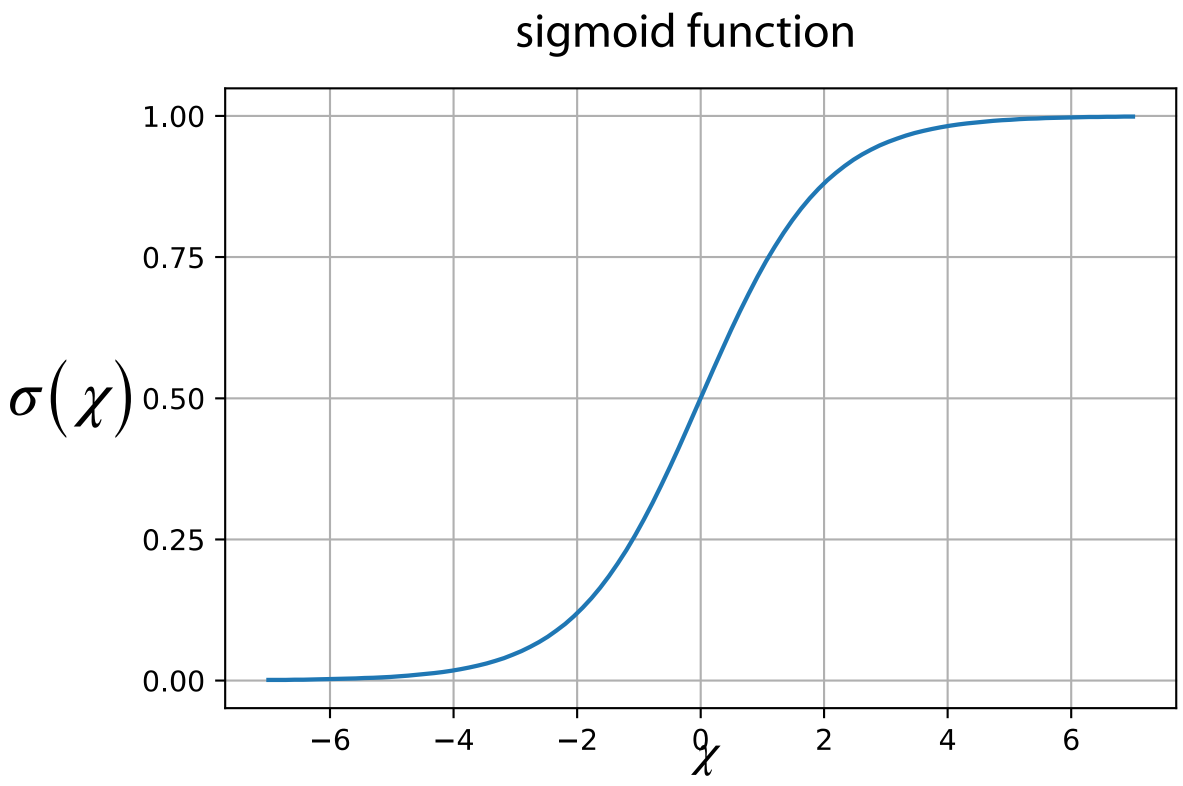


Linear probability model

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

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

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



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

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathscr{L} = -\sum_{i=1\ldots N}\left[\mathbb{1}\left[y_i = +1\right]\cdot\log\hat{f}_\theta\left(x_i\right) + \mathbb{1}\left[y_i = -1\right]\cdot\log\left(1 - \hat{f}_\theta\left(x_i\right)\right)\right]$$

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathscr{L} = -\sum_{i=1\ldots N}\left[\mathbb{I}\left[y_i = +1\right]\cdot\log\hat{f}_\theta\left(x_i\right) + \mathbb{I}\left[y_i = -1\right]\cdot\log\left(1 - \hat{f}_\theta\left(x_i\right)\right)\right]$$

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

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

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathscr{L} = -\sum_{i=1\ldots N} \left[\mathbb{I}\left[y_i = +1\right] \cdot \log \hat{f}_\theta(x_i) + \mathbb{I}\left[y_i = -1\right] \cdot \log\left(1 - \hat{f}_\theta(x_i)\right)\right]$$

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

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

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

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathscr{L} = -\sum_{i=1\ldots N}\left[\mathbb{I}\left[y_i = +1\right]\cdot\log\hat{f}_\theta\left(x_i\right) + \mathbb{I}\left[y_i = -1\right]\cdot\log\left(1-\hat{f}_\theta\left(x_i\right)\right)\right]$$

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

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

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

Bringing it all together

- Use negative log likelihood as our loss function:

$$\mathscr{L} = -\sum_{i=1\ldots N}\left[\mathbb{I}\left[y_i = +1\right]\cdot\log\hat{f}_\theta\left(x_i\right) + \mathbb{I}\left[y_i = -1\right]\cdot\log\left(1 - \hat{f}_\theta\left(x_i\right)\right)\right]$$

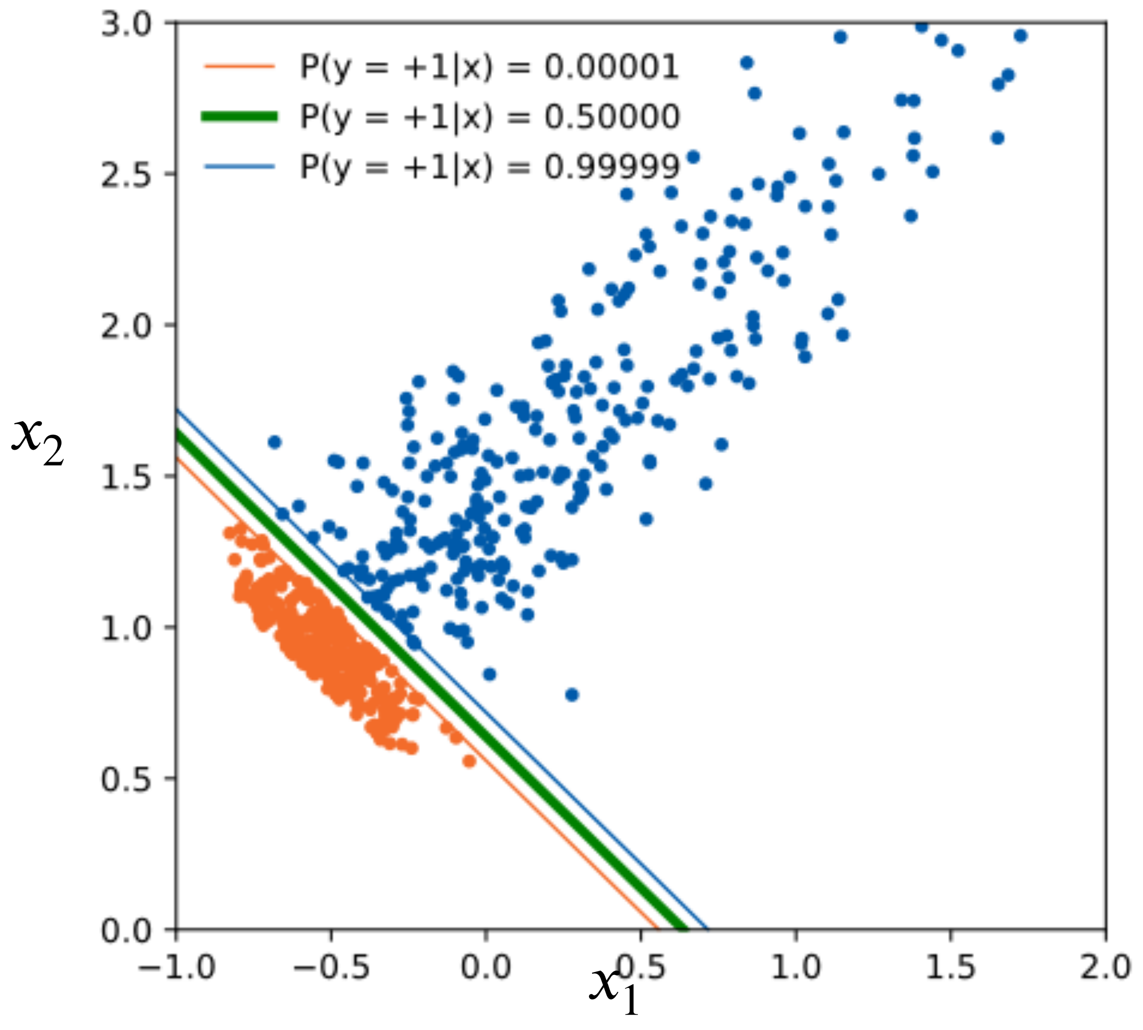
$1 - \sigma(x) = \sigma(-x)$

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

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

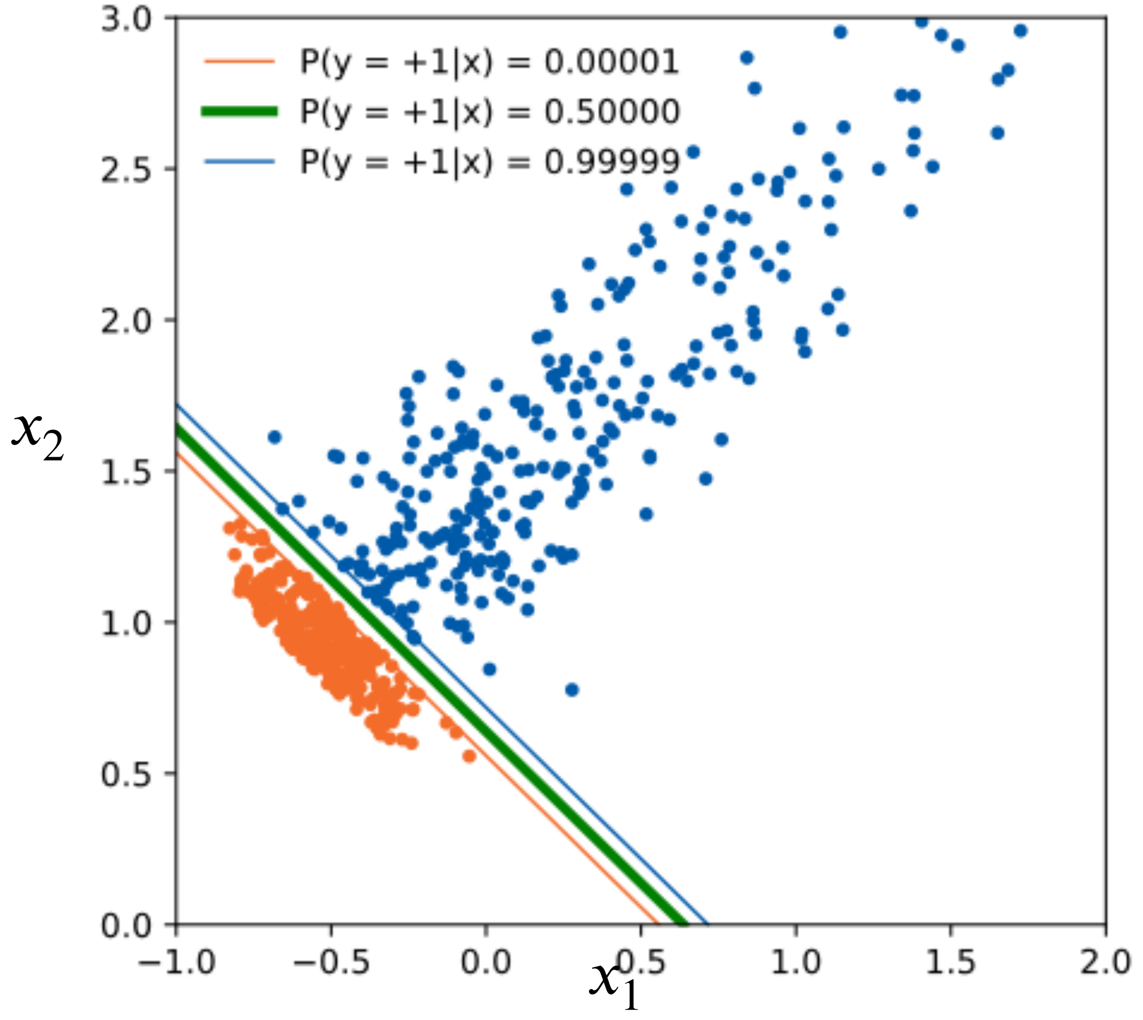
- This can be optimized numerically

Example



- Now the boundary is at the right place

Example

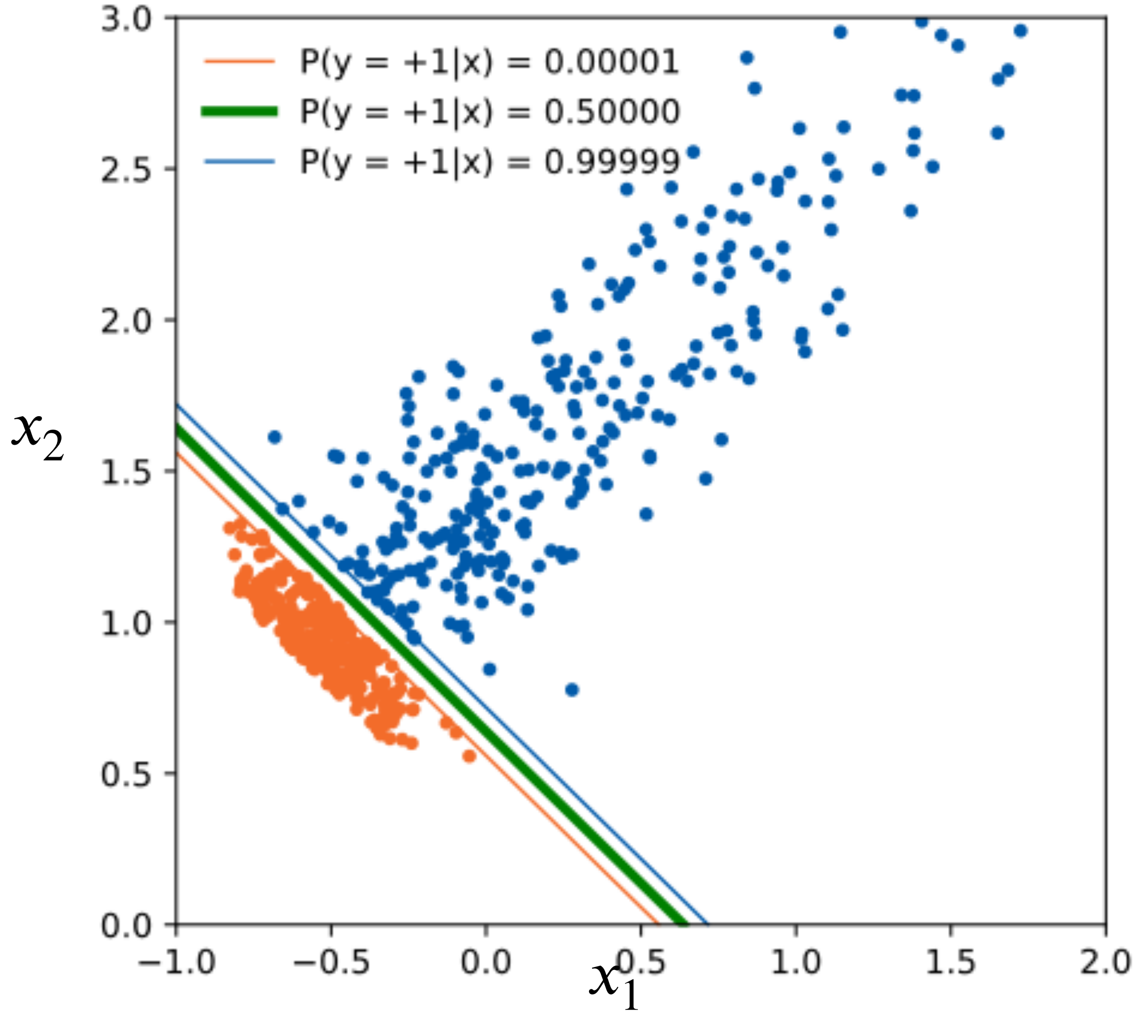


- Now the boundary is at the right place
- Note: when classes are linearly separable for any correct decision boundary

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

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

Example



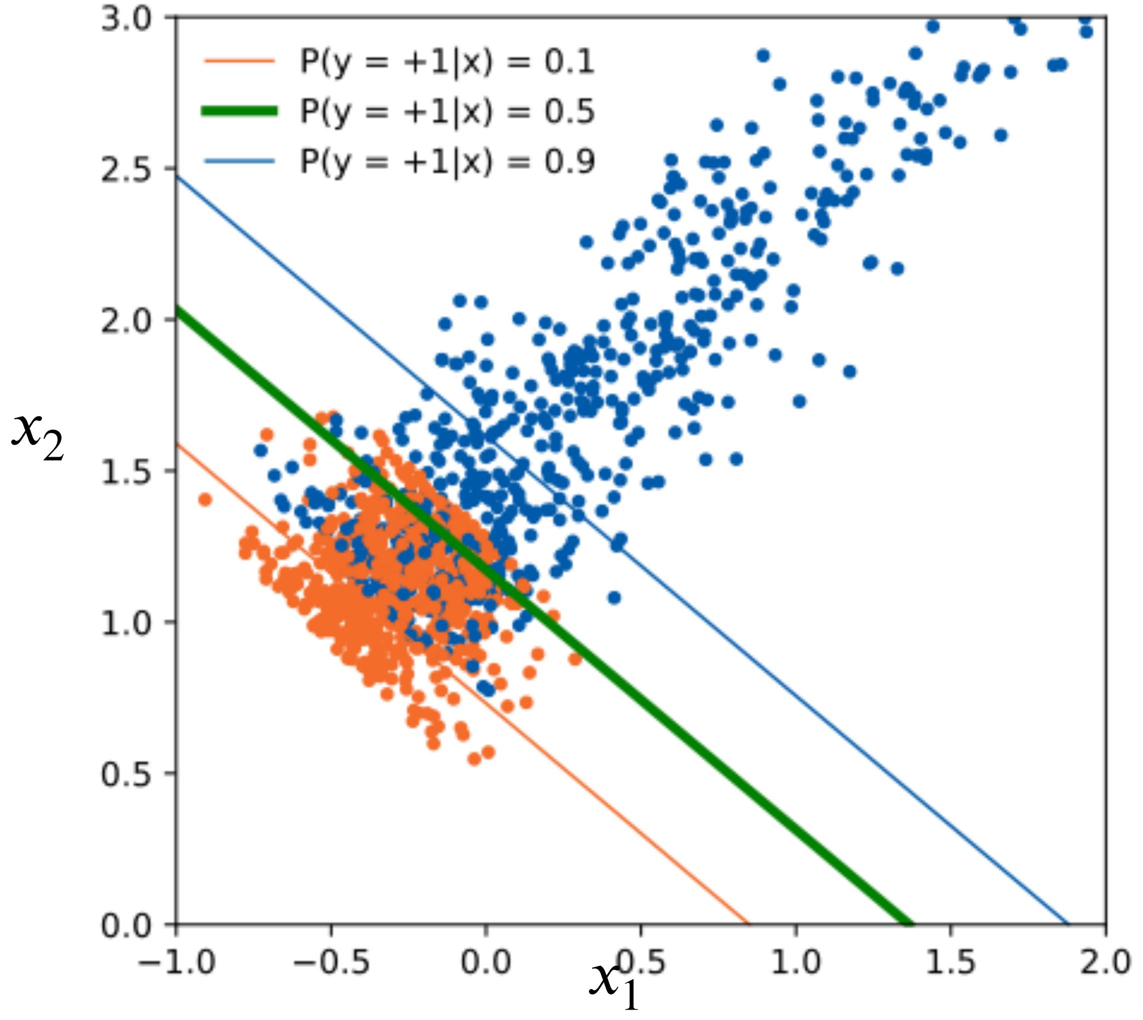
- Now the boundary is at the right place
- Note: when classes are linearly separable for any correct decision boundary

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

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:

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

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:

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

Note: now we have K parameter vectors

- Then, the normalized probabilities are:

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

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

Multinomial Logistic Regression

- Note that transforming all $\theta_k \longrightarrow \theta_k + v$ by some constant vector v does not affect the normalized probability

$$\tilde{P}\left(y = k \,\middle|\, x\right) = e^{\theta_k^{\mathrm{T}} x} \longrightarrow e^{v^{\mathrm{T}} x} \cdot e^{\theta_k^{\mathrm{T}} x} = e^{v^{\mathrm{T}} x} \cdot \tilde{P}\left(y = k \,\middle|\, x\right)$$

Multinomial Logistic Regression

- Note that transforming all $\theta_k \longrightarrow \theta_k + v$ by some constant vector v does not affect the normalized probability

$$\tilde{P}\left(y = k \middle| x\right) = e^{\theta_k^{\mathrm{T}} x} \longrightarrow e^{v^{\mathrm{T}} x} \cdot e^{\theta_k^{\mathrm{T}} x} = e^{v^{\mathrm{T}} x} \cdot \tilde{P}\left(y = k \middle| x\right)$$

$$P\left(y = k \middle| x\right) = \frac{\tilde{P}\left(y = k \middle| x\right)}{\sum_{k'=1\ldots K} \tilde{P}\left(y = k' \;\; x\right)} \longrightarrow \frac{e^{v^{\mathrm{T}} x} \cdot \tilde{P}\left(y = k \middle| x\right)}{\sum_{k'=1\ldots K} e^{v^{\mathrm{T}} x} \cdot \tilde{P}\left(y = k' \;\; x\right)} = P\left(y = k \middle| x\right)$$

Multinomial Logistic Regression

- Note that transforming all $\theta_k \longrightarrow \theta_k + \upsilon$ by some constant vector υ does not affect the normalized probability

$$\tilde{P}(y = k \mid x) = e^{\theta_k^{\mathrm{T}} x} \longrightarrow e^{\upsilon^{\mathrm{T}} x} \cdot e^{\theta_k^{\mathrm{T}} x} = e^{\upsilon^{\mathrm{T}} x} \cdot \tilde{P}(y = k \mid x)$$

$$P(y = k \mid x) = \frac{\tilde{P}(y = k \mid x)}{\sum_{k'=1\ldots K} \tilde{P}(y = k' \; x)} \longrightarrow \frac{e^{\upsilon^{\mathrm{T}} x} \cdot \tilde{P}(y = k \mid x)}{\sum_{k'=1\ldots K} e^{\upsilon^{\mathrm{T}} x} \cdot \tilde{P}(y = k' \; x)} = P(y = k \mid x)$$

- This means we can set one of the vectors θ_k to 0, e.g. the last one:

$$\theta_K = 0$$

Multinomial Logistic Regression

- We now have $K - 1$ parameter vectors
- Individual linear outputs $\theta_k^{\mathrm{T}} x$ now have the meaning of log odds ratio between the classes k and K:

$$\log \frac{P(y = k \mid x)}{P(y = K \mid x)} = \log \frac{\widetilde{P}(y = k \mid x)}{\widetilde{P}(y = K \mid x)} = \log \frac{e^{\theta_k^{\mathrm{T}} x}}{e^0} = \theta_k^{\mathrm{T}} x$$

Multinomial Logistic Regression

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

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

$$\left(\theta_K = 0\right)$$

- Again, this can be optimized numerically

Multiclass classification: general approach

General idea

For a problem with K classes introduce K predictors:

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

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

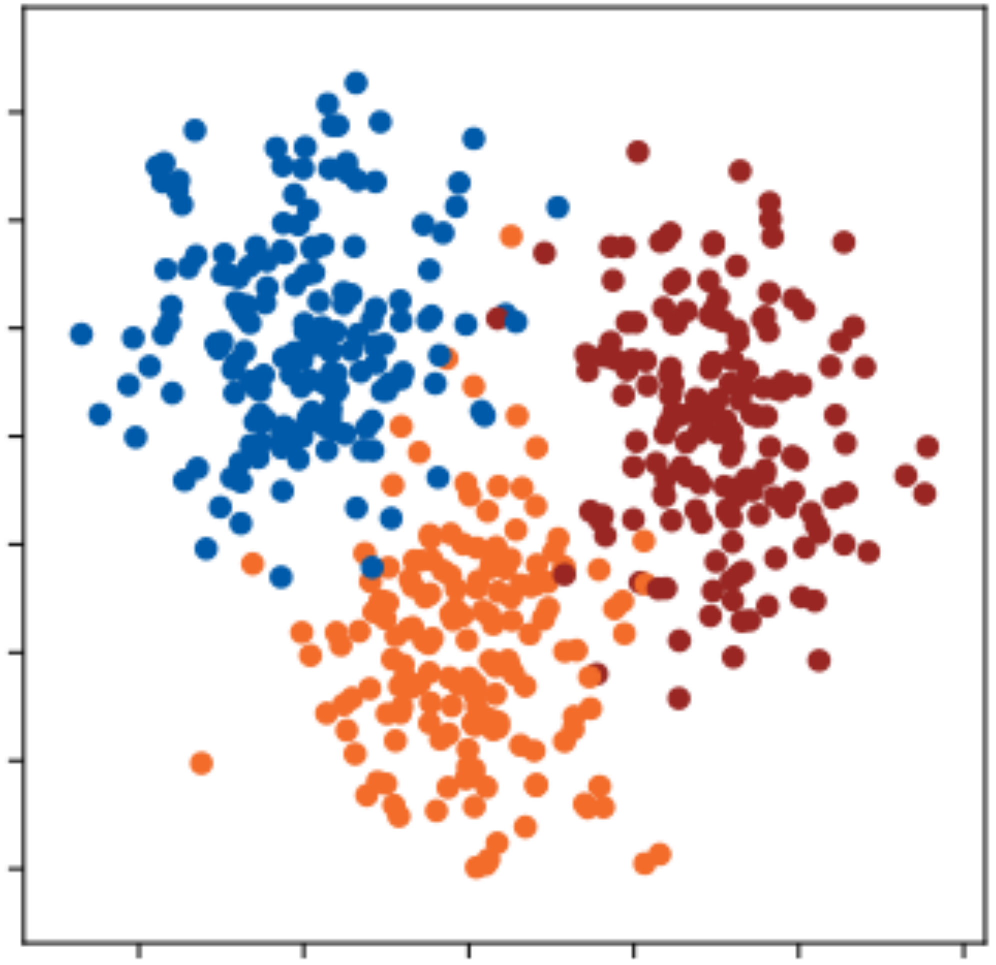
- Any binary linear classification model can be converted to multiclass with one-vs-rest strategy
- For each class k train a binary model $\hat{f}_k(x) = \theta_{(k)}^{\mathrm{T}} x$ separating the given class from all others, $\hat{y}_{(k)}^{1-\mathrm{vs}-\mathrm{rest}} = \operatorname{sign}\left[\hat{f}_k(x)\right]$

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 $\hat{f}_k(x) = \theta_{(k)}^{\mathrm{T}} x$ separating the given class from all others, $\hat{y}_{(k)}^{1-\mathrm{vs}-\mathrm{rest}} = \mathrm{sign}\left[\hat{f}_k(x)\right]$
- Use the outputs of $\hat{f}_k$ as class scores for multiclass classification:

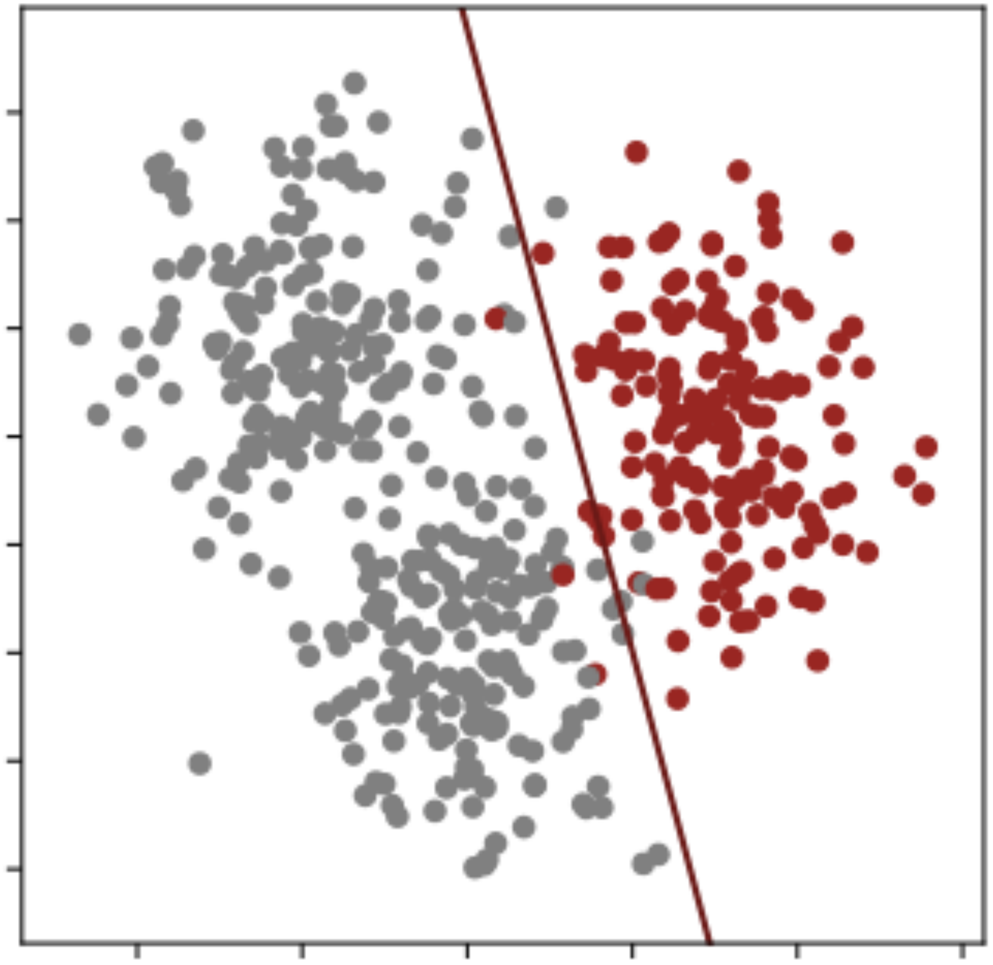
$$\hat{y}_i = \underset{k}{\mathrm{argmax}} \hat{f}_k(x_i)$$

Example



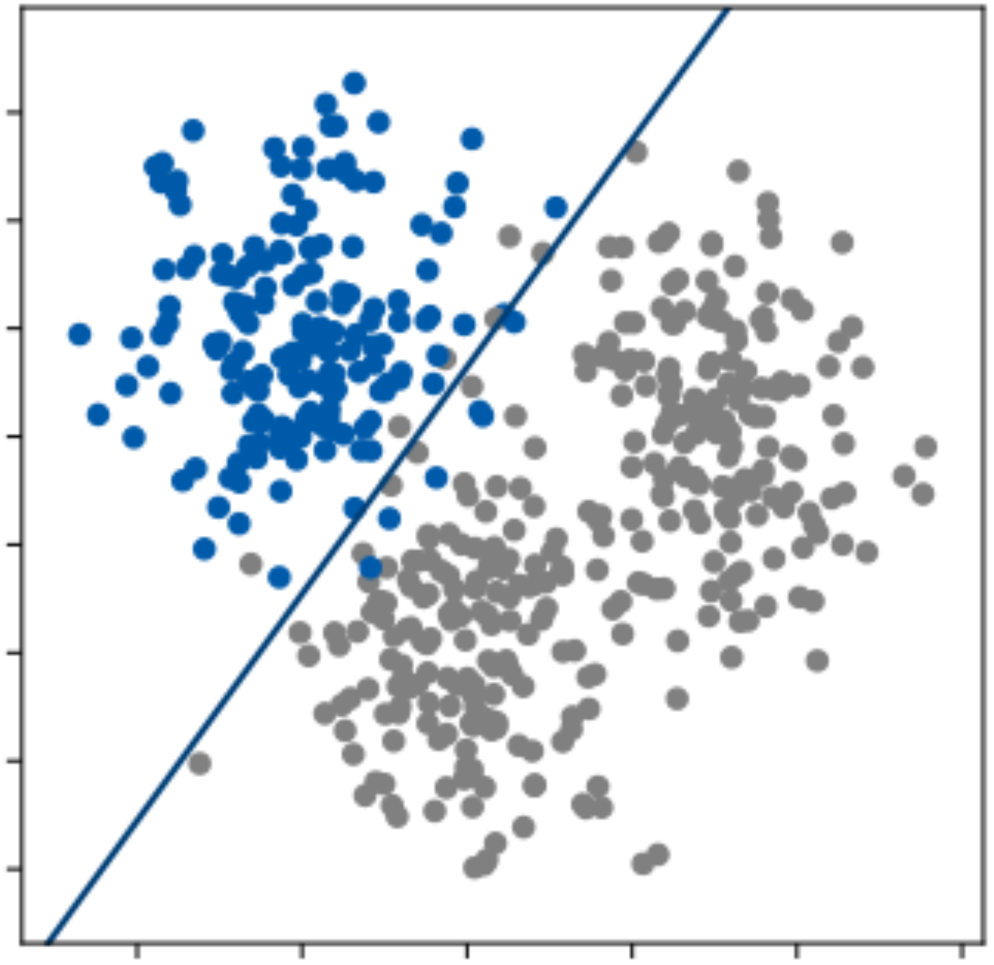
- Consider the following 3 class problem

Example



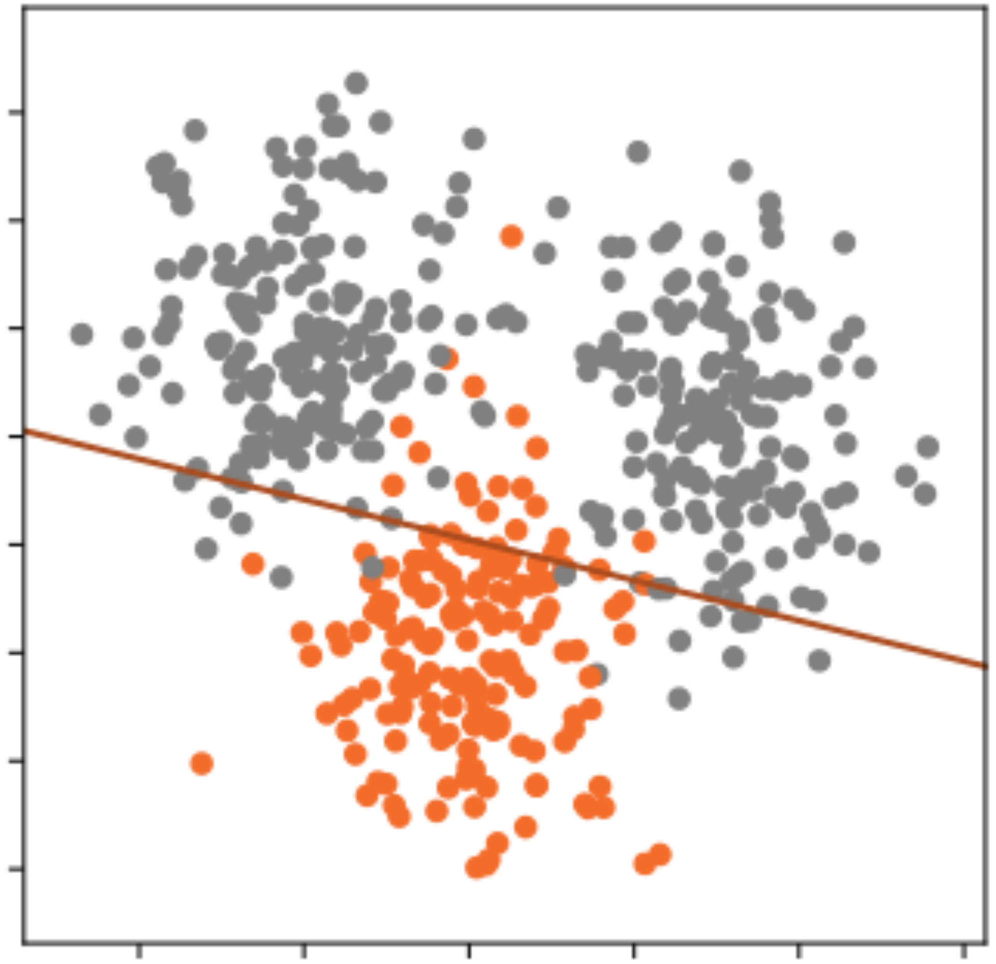
- “Class-1 VS rest” binary classifier

Example



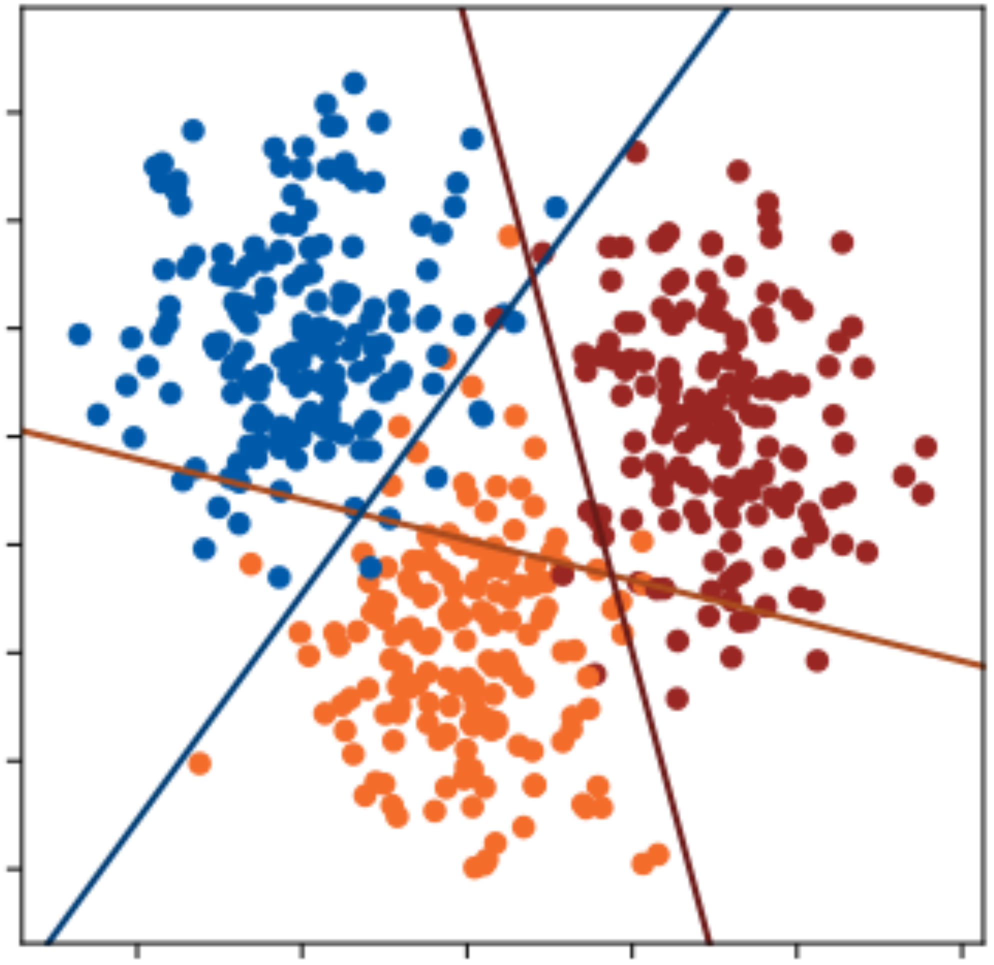
- “Class-2 VS rest” binary classifier

Example



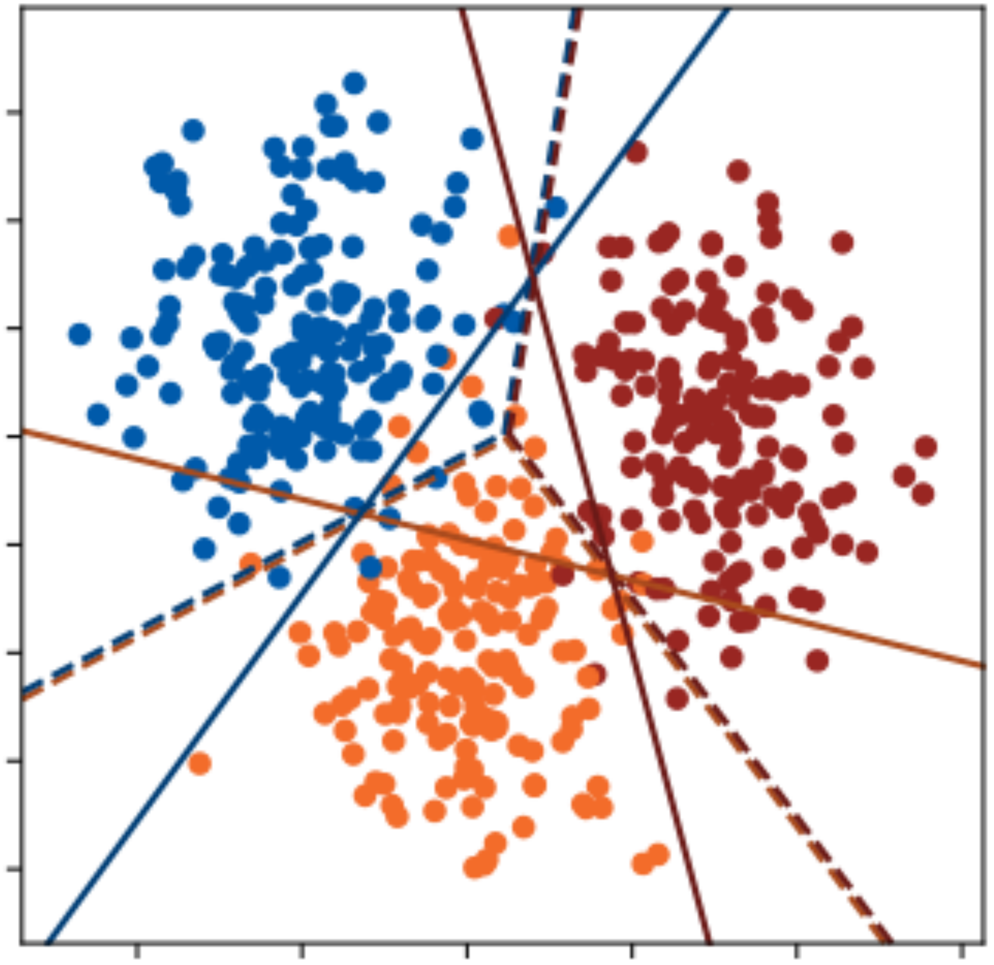
- “Class-3 VS rest” binary classifier

Example



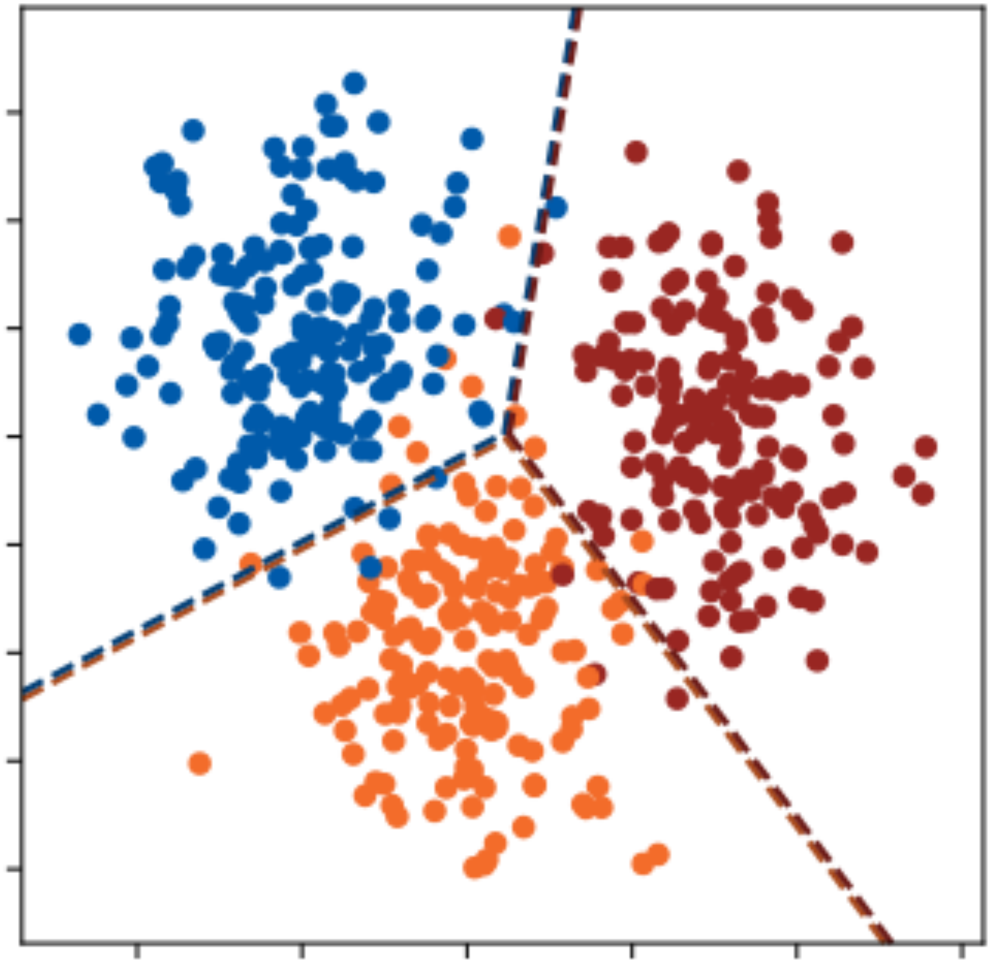
- $\hat{f}_k(x) = 0$ lines (binary decision boundaries)

Example



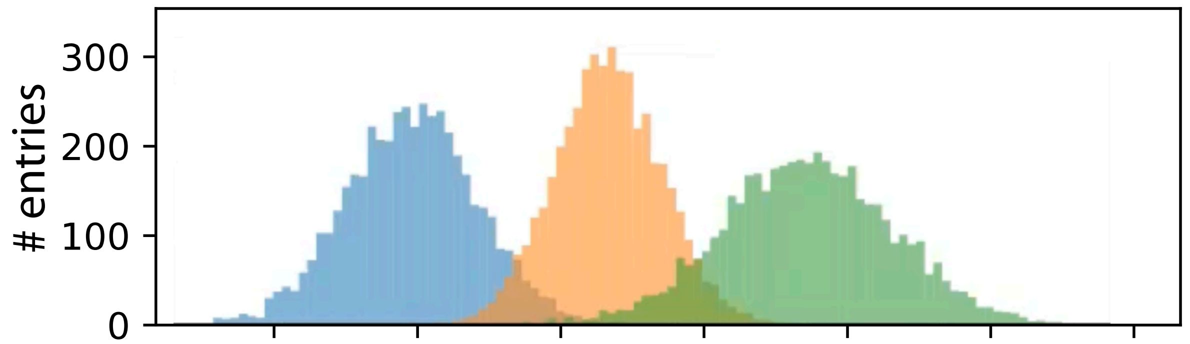
- $\hat{f}_k(x) = 0$ lines (binary decision boundaries)
- Adding decision boundaries for $\hat{y} = \underset{k}{\operatorname{argmax}} \hat{f}_k(x)$

Example



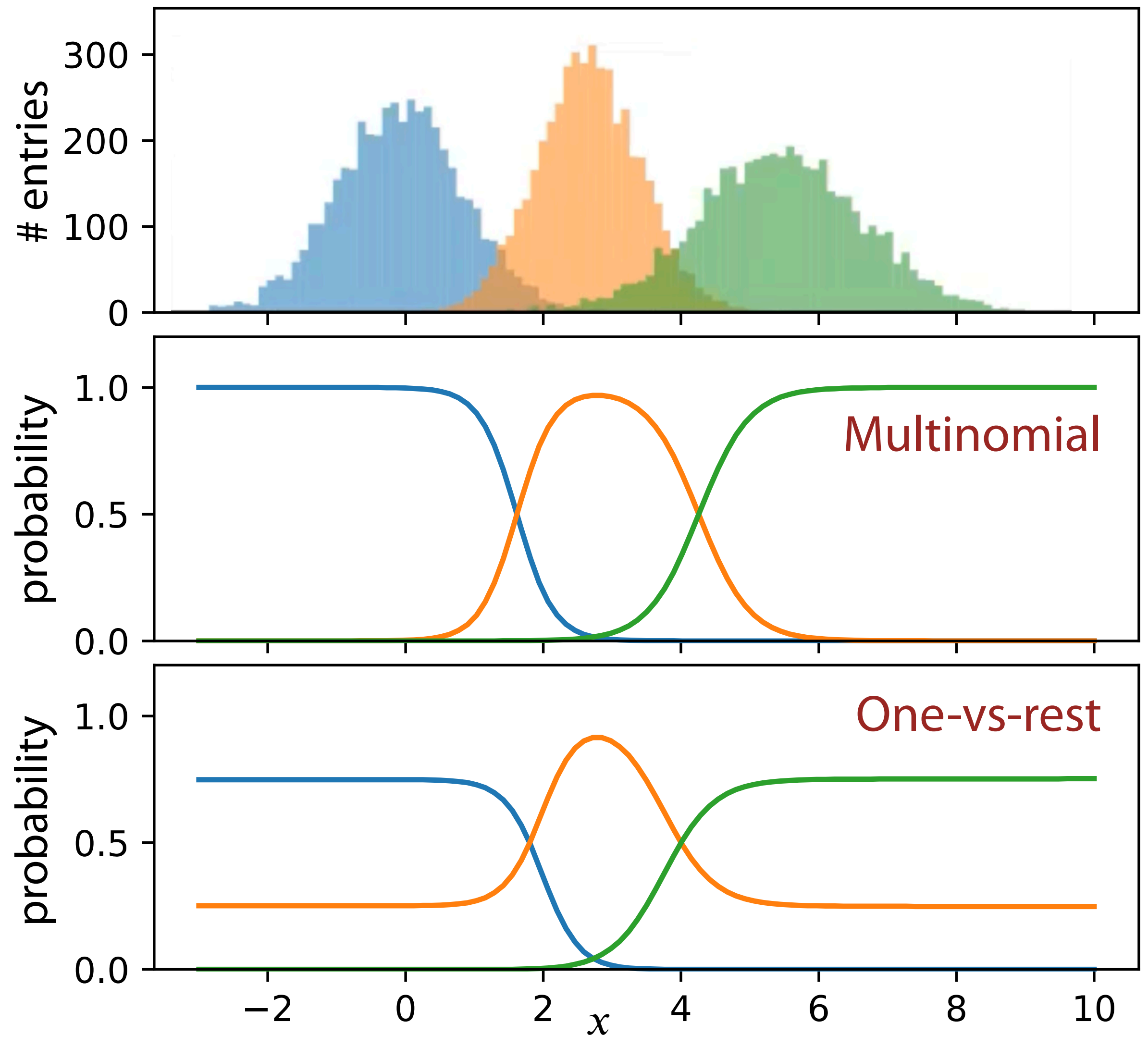
- Adding decision boundaries for $\hat{y} = \underset{k}{\operatorname{argmax}} \hat{f}_k(x)$

Logistic regression: multinomial or one-vs-rest?



Some of the binary classification tasks not linearly solvable

Logistic regression: multinomial or one-vs-rest?



Some of the binary classification tasks not linearly solvable

⇒ one-vs-rest results in biased class probabilities

Summary

- Classification with linear regression and MSE loss may provide biased results

Summary

- Classification with linear regression and MSE loss may provide biased results
- 0-1 loss function is better, but is hard to optimize directly

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

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

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

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

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!

al-maeeni@hse.ru

@afdee1c

@AFDEE1C