# Decision-theoretic Machine Learning

Krzysztof Dembczyński and Wojciech Kotłowski

Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland


Poznań University of Technology, Summer 2019

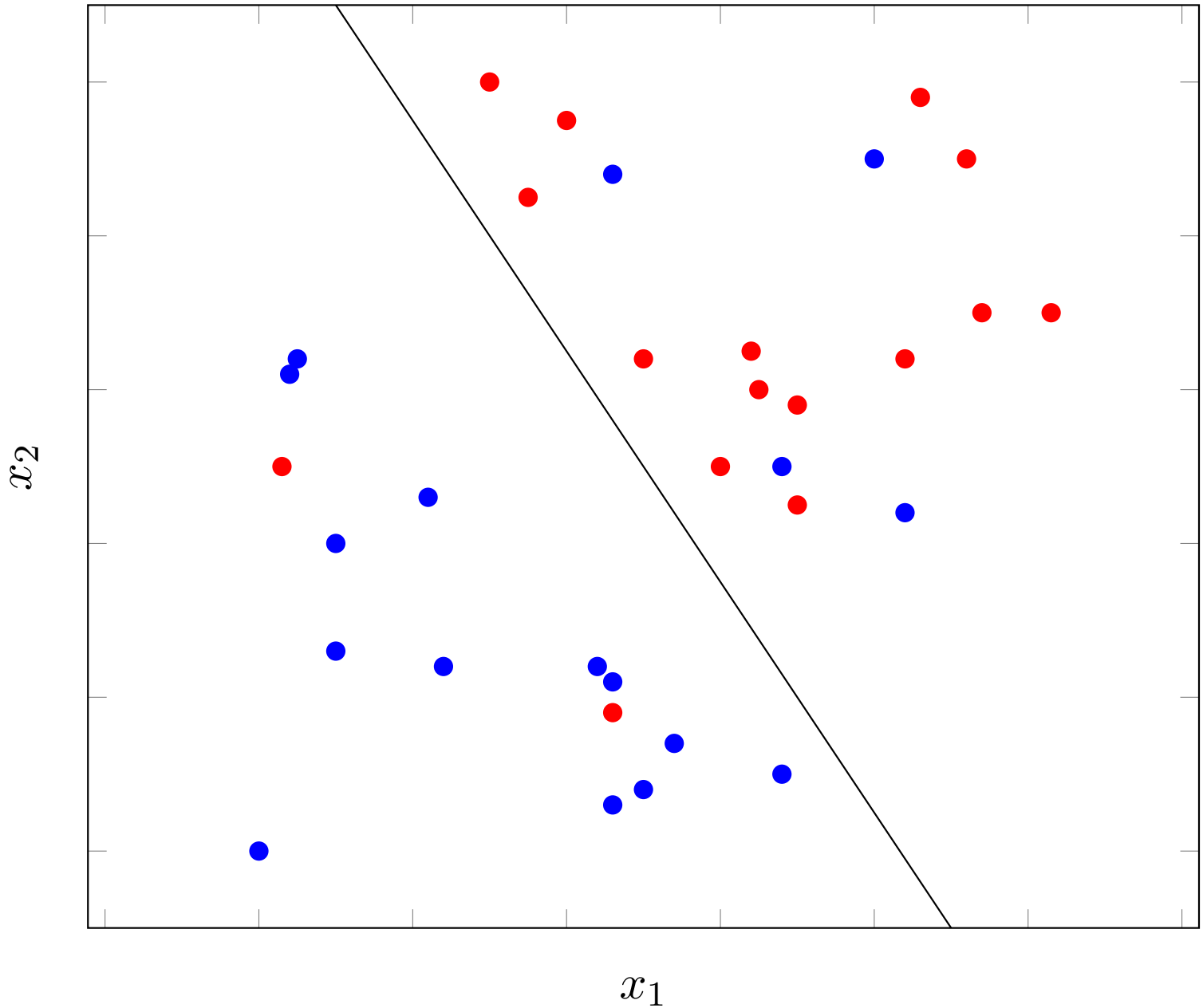
# Agenda

1. Introduction to Machine Learning
2. **Binary Classification**
3. Bipartite Ranking
4. Multi-Label Classification

# Outline

# Outline

# Linear models for classification



# Linear models for classification

- Let the output variable be $y \in \{-1, 1\}$ (or alternatively $y \in \{0, 1\}$).
- Prediction function $h(\boldsymbol{x}) \in \{-1, 1\}$ (or alternatively $h(\boldsymbol{x}) \in \{0, 1\}$).

# Linear models for classification

- Let the output variable be $y \in \{-1, 1\}$ (or alternatively $y \in \{0, 1\}$).
- Prediction function $h(\boldsymbol{x}) \in \{-1, 1\}$ (or alternatively $h(\boldsymbol{x}) \in \{0, 1\}$).
- Loss is measured usually in terms of $0/1$ loss which can be expressed by:

$$\ell_{0/1}(y, h(\boldsymbol{x})) = [\![ yh(\boldsymbol{x}) \leq 0 ]\!]$$

# Linear models for classification

- Let the output variable be $y \in \{-1, 1\}$ (or alternatively $y \in \{0, 1\}$).
- Prediction function $h(\boldsymbol{x}) \in \{-1, 1\}$ (or alternatively $h(\boldsymbol{x}) \in \{0, 1\}$).
- Loss is measured usually in terms of 0/1 loss which can be expressed by:

$$\ell_{0/1}(y, h(\boldsymbol{x})) = [\![ yh(\boldsymbol{x}) \leq 0 ]\!]$$

- Solve:

$$\widehat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}(y_i, h(\boldsymbol{x}_i)) = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} [\![ y_i h(\boldsymbol{x}_i) \leq 0 ]\!]$$

# Linear models for classification

- Let the output variable be $y \in \{-1, 1\}$ (or alternatively $y \in \{0, 1\}$).
- Prediction function $h(\boldsymbol{x}) \in \{-1, 1\}$ (or alternatively $h(\boldsymbol{x}) \in \{0, 1\}$).
- Loss is measured usually in terms of $0/1$ loss which can be expressed by:

$$\ell_{0/1}(y, h(\boldsymbol{x})) = [\![ yh(\boldsymbol{x}) \leq 0 ]\!]$$

- Solve:

$$\widehat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}(y_i, h(\boldsymbol{x}_i)) = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} [\![ y_i h(\boldsymbol{x}_i) \leq 0 ]\!]$$

- **Hard** to optimize $h$ directly.

# Linear models for classification

- Usually done in two phases:

# Linear models for classification

- Usually done in two phases:
  - Learn a **continuous function** $f \in \mathcal{F}$:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [\![ y_i f(\boldsymbol{x}_i) \leq 0 ]\!]$$

# Linear models for classification

- Usually done in two phases:
  - Learn a **continuous function** $f \in \mathcal{F}$:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [\![ y_i f(\boldsymbol{x}_i) \leq 0 ]\!]$$

  - **Threshold** $f$ at $0$:

$$\widehat{h} = \operatorname{sgn}\left(\widehat{f}\right), \qquad \mathcal{H} = \{h \colon h = \operatorname{sgn}(f), f \in \mathcal{F}\},$$

    so that $[\![ y_i h(\boldsymbol{x}_i) \leq 0 ]\!] = [\![ y_i f(\boldsymbol{x}_i) \leq 0 ]\!]$.

# Linear models for classification

- Solve:

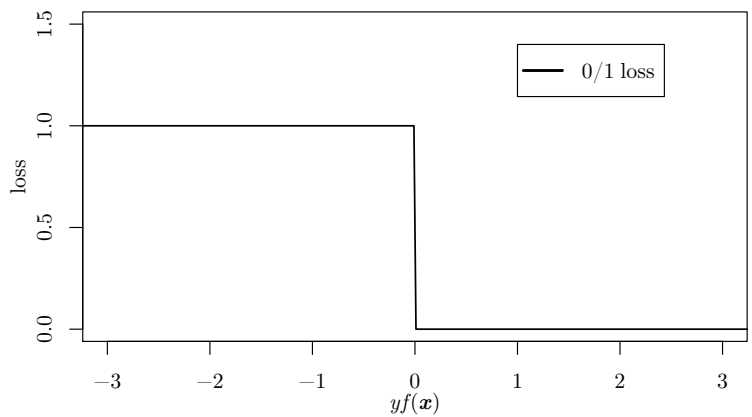
$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [\![ y_i f(\boldsymbol{x}_i) \leq 0 ]\!]$$

# Linear models for classification

- Solve:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [\![ y_i f(\boldsymbol{x}_i) \leq 0 ]\!]$$

- Still **hard** to optimize: $0/1$ loss is **discontinuous** and **non-convex**.
  - e.g., when $\mathcal{H}$ is a class of linear function, the problem known to be **NP-hard**.

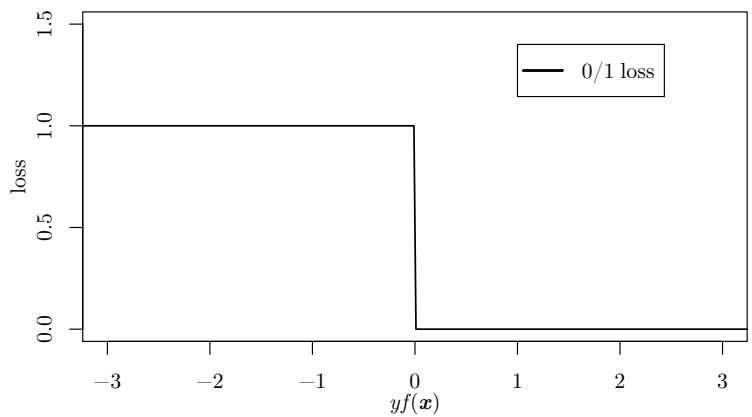


# Linear models for classification

- Solve:

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [\![ y_i f(\boldsymbol{x}_i) \leq 0 ]\!]$$

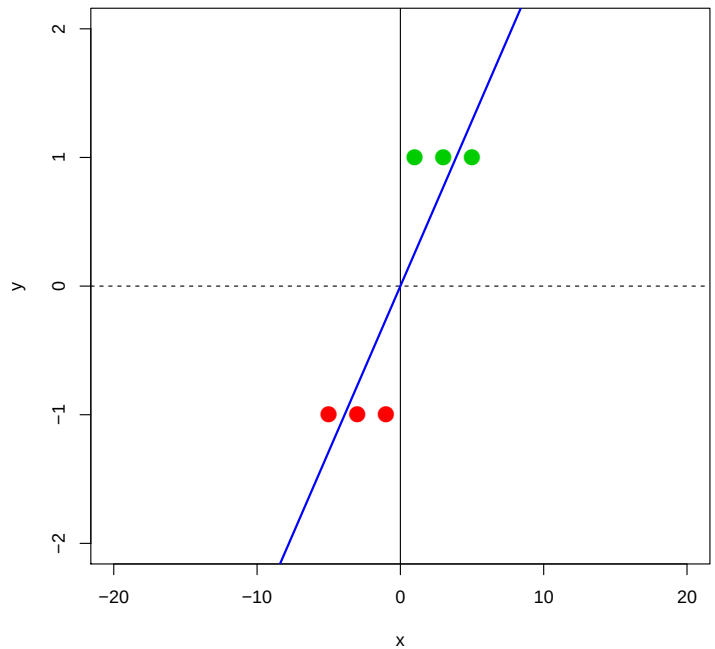
- Still **hard** to optimize: $0/1$ loss is **discontinuous** and **non-convex**.
  - e.g., when $\mathcal{H}$ is a class of linear function, the problem known to be **NP-hard**.




- **Solution**: use some **convex relaxation** of $0/1$ loss.

# Linear regression for classification

- Let us assume that $y \in \{-1, 1\}$.
- We can try to use linear regression to solve binary classification problem:

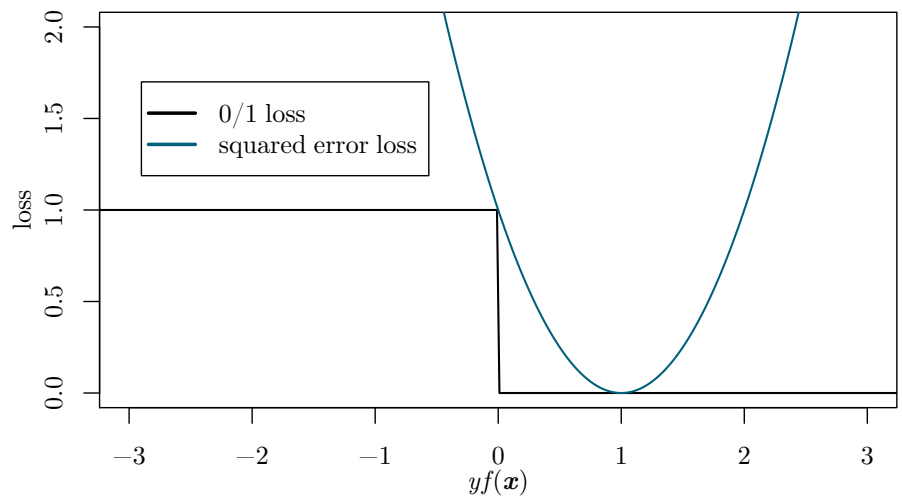



- Use a threshold to decide which class is predicted:
  Positive class: $f(\boldsymbol{x}) > 0$ Negative class: $f(\boldsymbol{x}) < 0$
- Minimization of squared loss is not bad, since it leads to estimation of the conditional probability, i.e., $P(y = 1|\boldsymbol{x}) = \frac{1}{2}\mathbb{E}(y|\boldsymbol{x}) + \frac{1}{2}$.

# Linear regression for classification

- Effectively, we replaced 0/1 loss by **squared error loss**:

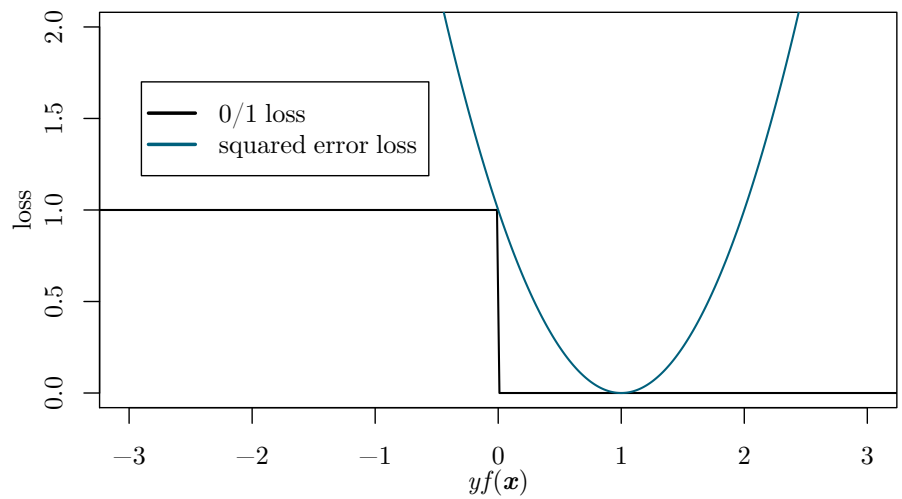
$$\ell_{\text{sq}}(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2 = (1 - yf(\boldsymbol{x}))^2.$$



# Linear regression for classification

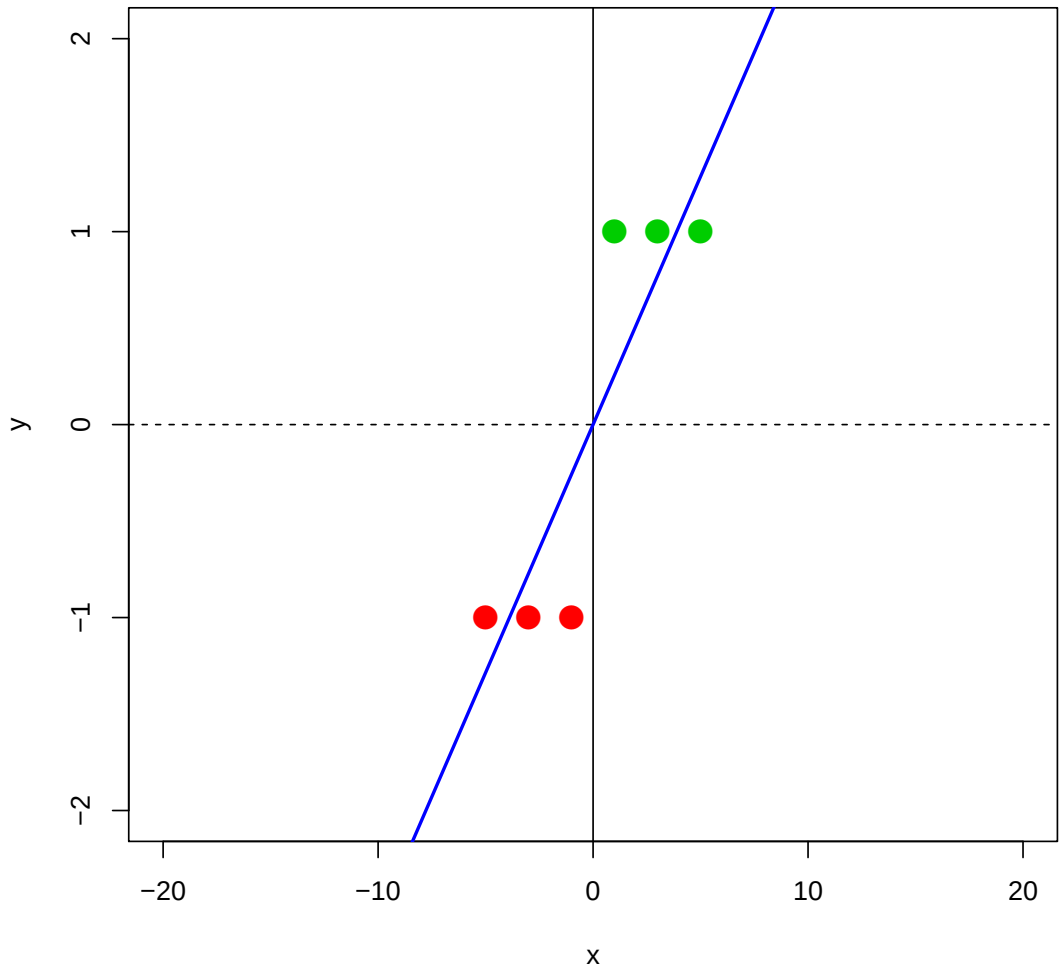
- Effectively, we replaced 0/1 loss by **squared error loss**:

$$\ell_{\text{sq}}(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2 = (1 - yf(\boldsymbol{x}))^2.$$

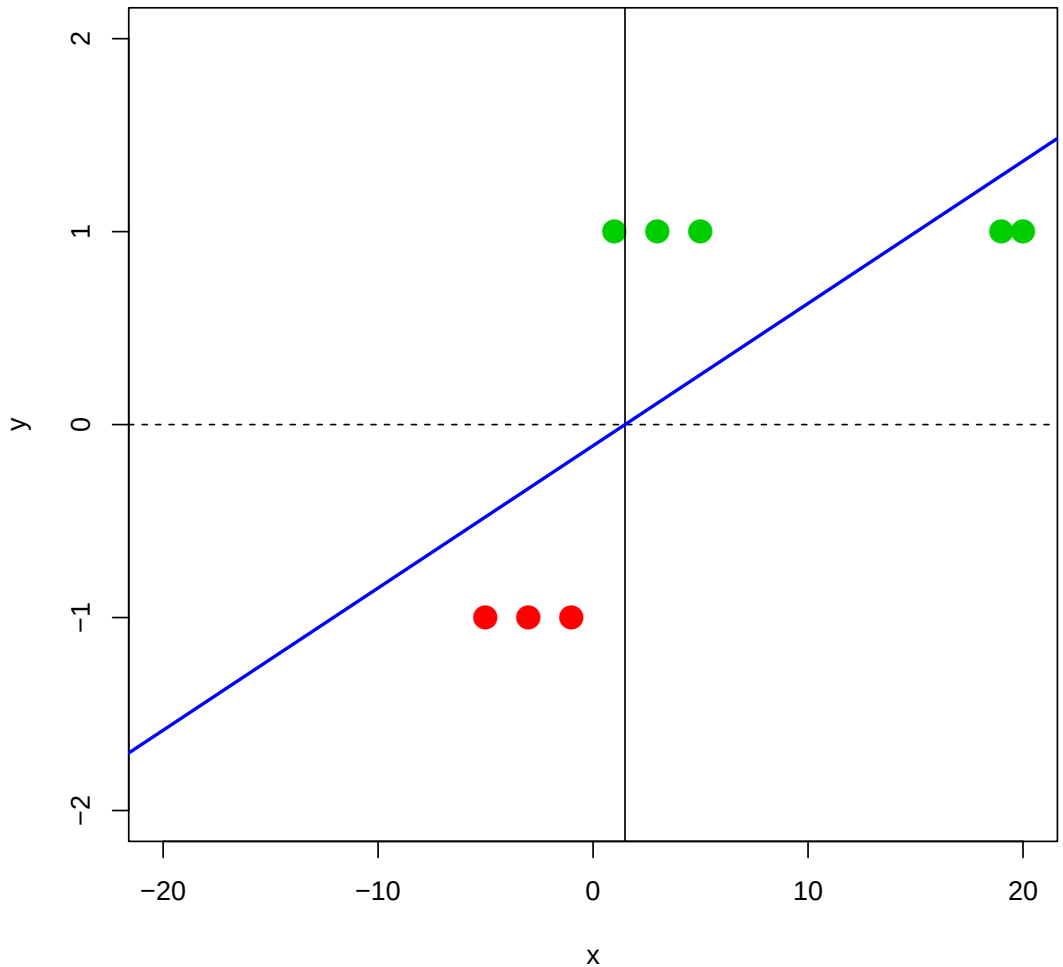



- Works nicely in practice, but has several drawbacks . . .

# Linear regression for classification



# Linear regression for classification




- It tries to minimize the squared error of all examples, even those that are correctly classified.

# Linear models for classification

- One possibility is to consider only those examples that are incorrectly classified.
- The quantity $yf(\boldsymbol{x})$ is usually referred to as **margin**.
- If $y$ and $f(\boldsymbol{x})$ have the same sign (i.e., the prediction is correct), then

$$yf(\boldsymbol{x}) > 0\,,$$

otherwise

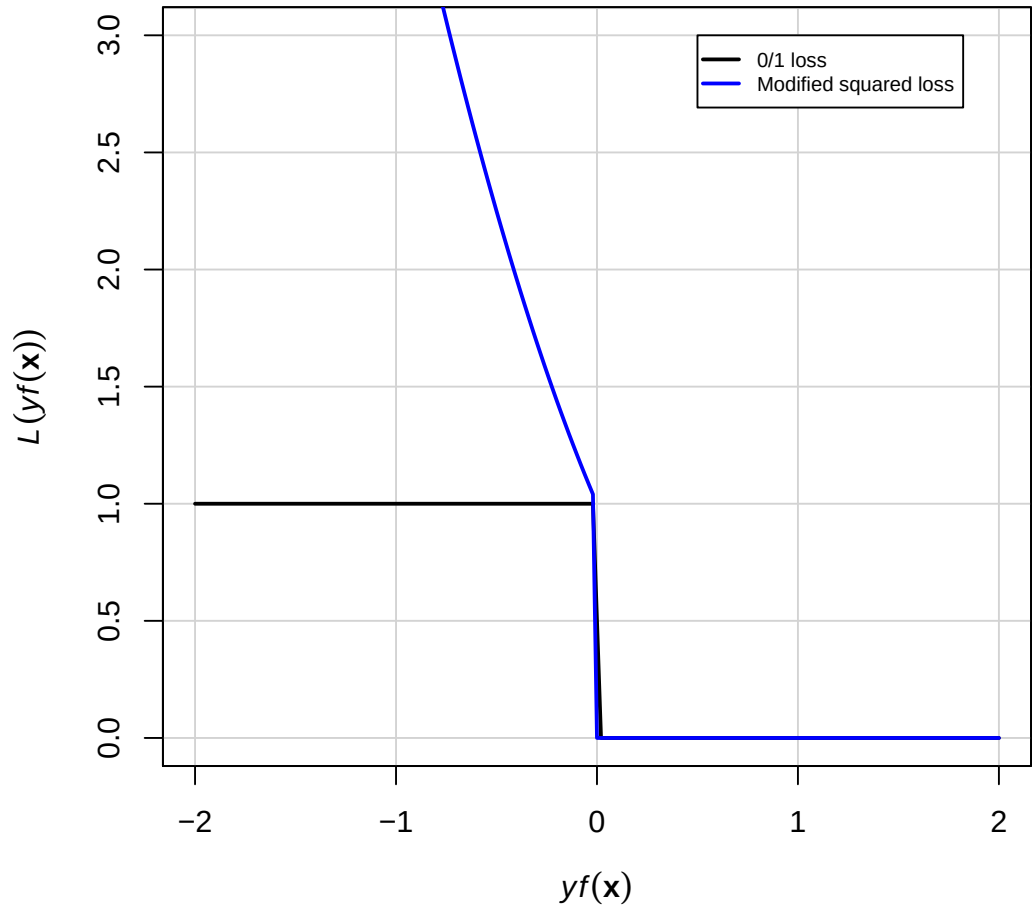
$$yf(\boldsymbol{x}) \leq 0\,.$$

# Linear models for classification

- We could consider the following loss function:

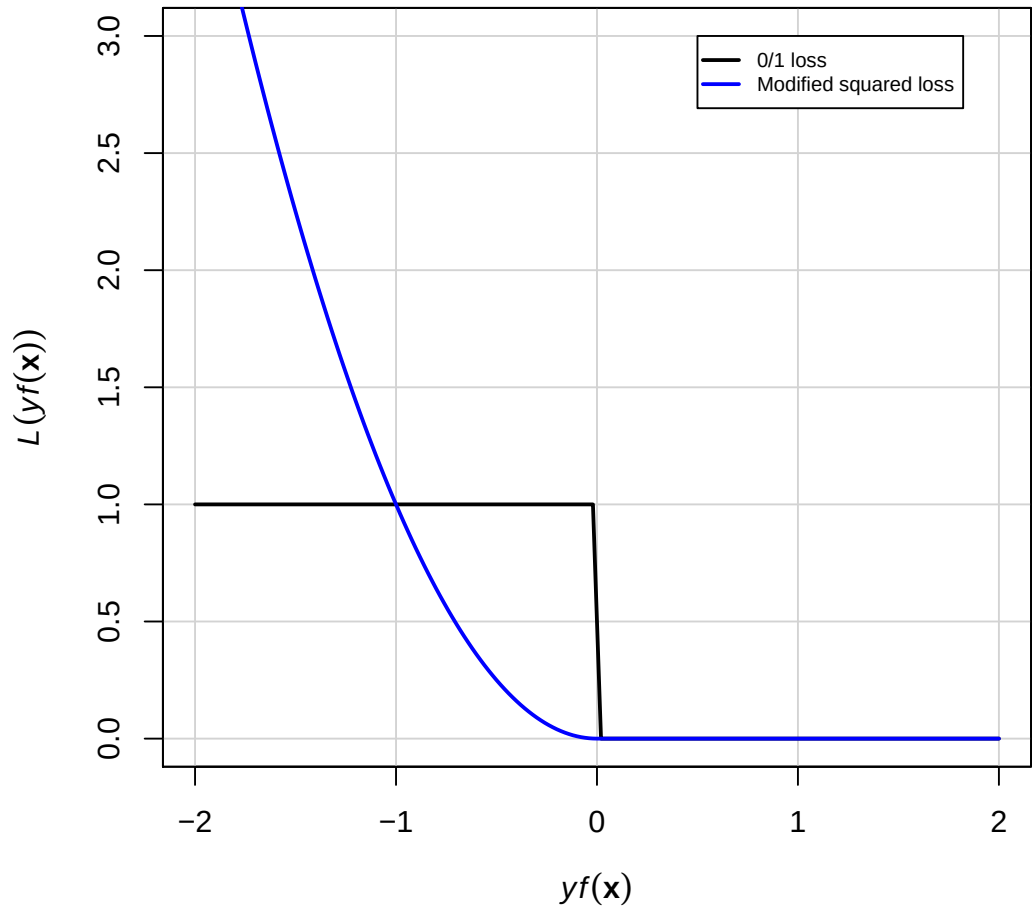
$$\ell(y, f(\boldsymbol{x})) = \begin{cases} 0, & \text{if } yf(\boldsymbol{x}) > 0 \\ (1 - yf(\boldsymbol{x}))^2, & \text{otherwise}\,. \end{cases}$$

# Linear models for classification



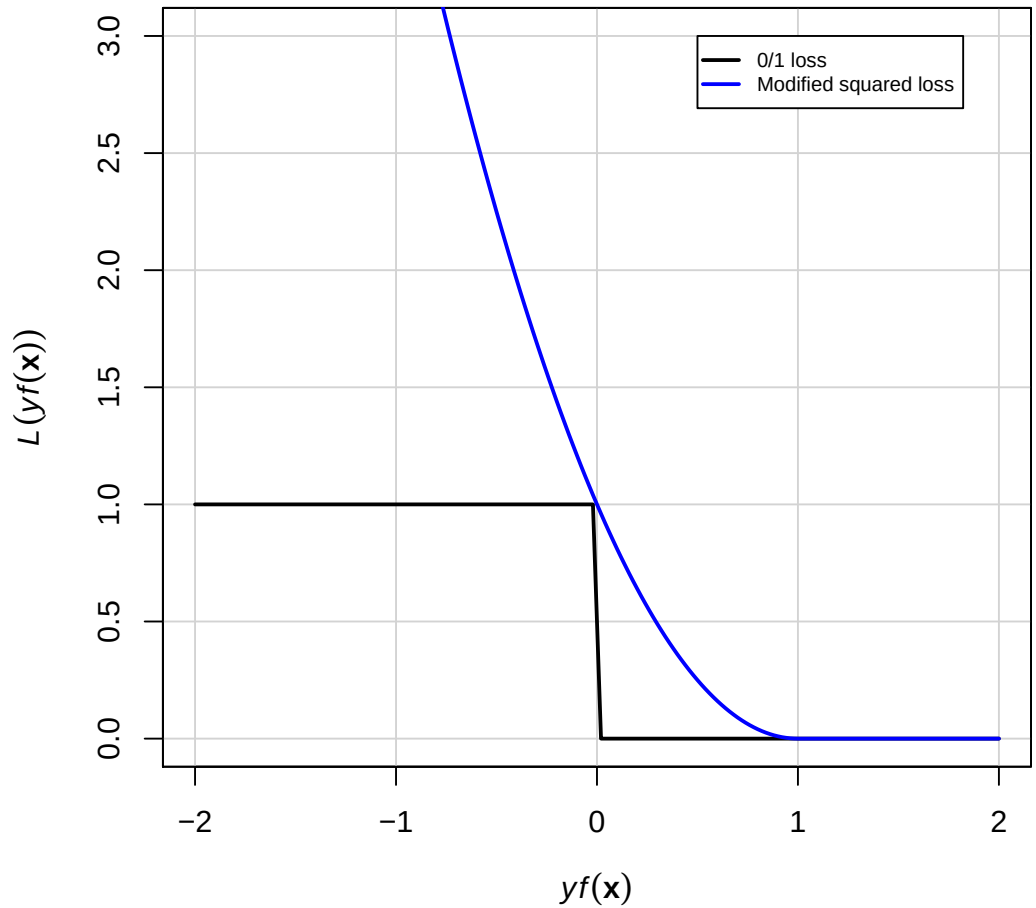

- This definition does not lead to a "nice" shape of the loss.

# Linear models for classification




- A better solution: $\ell(y, f(\boldsymbol{x})) = (\max\{0, \epsilon - yf(\boldsymbol{x})\})^2$.

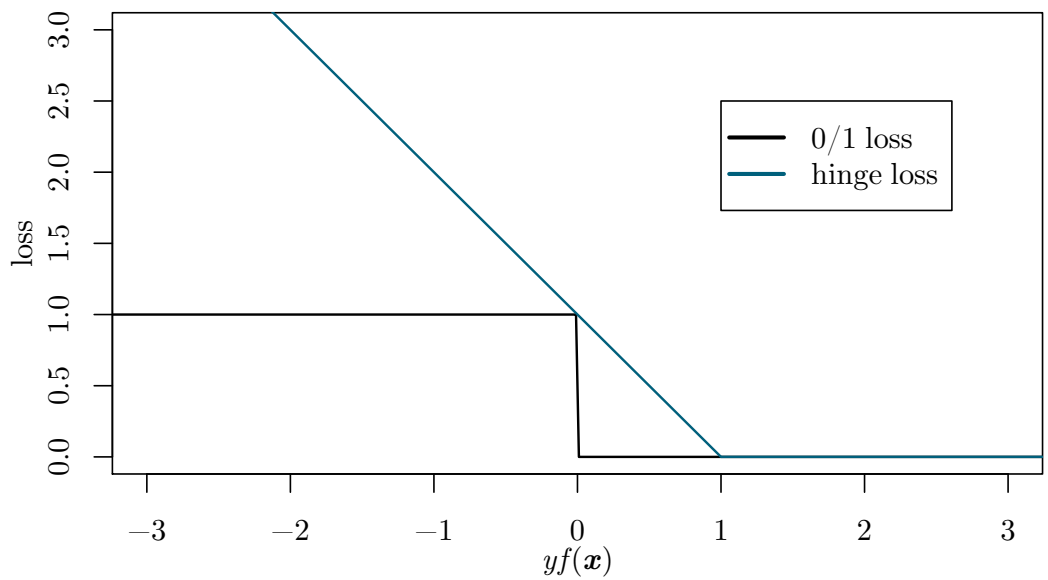
# Linear models for classification




- A better solution: $\ell(y, f(\boldsymbol{x})) = (\max\{0, 1 - yf(\boldsymbol{x})\})^2$.

# Linear models for classification

- Similarly, if we use absolute error instead of squared error, we get the following loss functions:
  - perceptron-like loss function: $\ell(y, f(\boldsymbol{x})) = \max\{0, \epsilon - yf(\boldsymbol{x})\}$,
  - hinge loss: $\ell(y, f(\boldsymbol{x})) = \max\{0, 1 - yf(\boldsymbol{x})\}$, used in support vector machines.



# Perceptron

- Perceptron uses a linear model:

$$f(\boldsymbol{x}) = w_0 + \sum_{j=0}^{n} w_j x_j = \boldsymbol{w} \cdot \boldsymbol{x}$$

# Perceptron

- Perceptron uses a linear model:

$$f(\boldsymbol{x}) = w_0 + \sum_{j=0}^{n} w_j x_j = \boldsymbol{w} \cdot \boldsymbol{x}$$

- We replace 0/1 loss by:

$$\begin{aligned} \ell_{\text{perc}}(y, f(\boldsymbol{x})) &= \max\{0, \epsilon - y f(\boldsymbol{x})\} \\ &= \max\{0, \epsilon - y \boldsymbol{w} \cdot \boldsymbol{x}\} \end{aligned}$$

# Perceptron

- Perceptron uses a linear model:

$$f(\boldsymbol{x}) = w_0 + \sum_{j=0}^{n} w_j x_j = \boldsymbol{w} \cdot \boldsymbol{x}$$

- We replace 0/1 loss by:

$$\begin{aligned} \ell_{\text{perc}}(y, f(\boldsymbol{x})) &= \max\{0, \epsilon - y f(\boldsymbol{x})\} \\ &= \max\{0, \epsilon - y \boldsymbol{w} \cdot \boldsymbol{x}\} \end{aligned}$$

- We solve

$$\begin{aligned} \widehat{f} &= \arg\min_{f} \frac{1}{n} \sum_{i=1}^{n} \max\{0, \epsilon - y_i f(\boldsymbol{x}_i)\} \\ &= \arg\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \max\{0, \epsilon - y_i \boldsymbol{w} \cdot \boldsymbol{x}_i\} \end{aligned}$$

  using the stochastic gradient descent algorithm.

# Stochastic gradient descent

Input: learning rate $\alpha$
$\boldsymbol{w} = \mathbf{0}$; *//(or use random values)*
**while** (approximate minimum is obtained) {
    Randomly shuffle examples in the training set
    **for** i=1 to n {
        $\boldsymbol{w} := \boldsymbol{w} - \alpha \frac{\partial \ell(y_i, f(\boldsymbol{x}_i))}{\partial \boldsymbol{w}}$
    }
}

# Perceptron

- Learning algorithm for perceptron:
  - For a misclassified example ($y\boldsymbol{w} \cdot \boldsymbol{x} \leq 0$), the gradient of $\ell_{\text{perc}}(y, f(\boldsymbol{x}))$ with respect to $\boldsymbol{w}$ is given by:

$$\frac{\partial \ell_{\text{perc}}(y, f(\boldsymbol{x}))}{\partial \boldsymbol{w}} = -y\boldsymbol{x}\,.$$

  - For a correctly classified example ($y\boldsymbol{w} \cdot \boldsymbol{x} > 0$), the gradient is $0$.

# Perceptron

- Learning algorithm for perceptron:
  - For a misclassified example ($y\boldsymbol{w} \cdot \boldsymbol{x} \leq 0$), the gradient of $\ell_{\text{perc}}(y, f(\boldsymbol{x}))$ with respect to $\boldsymbol{w}$ is given by:

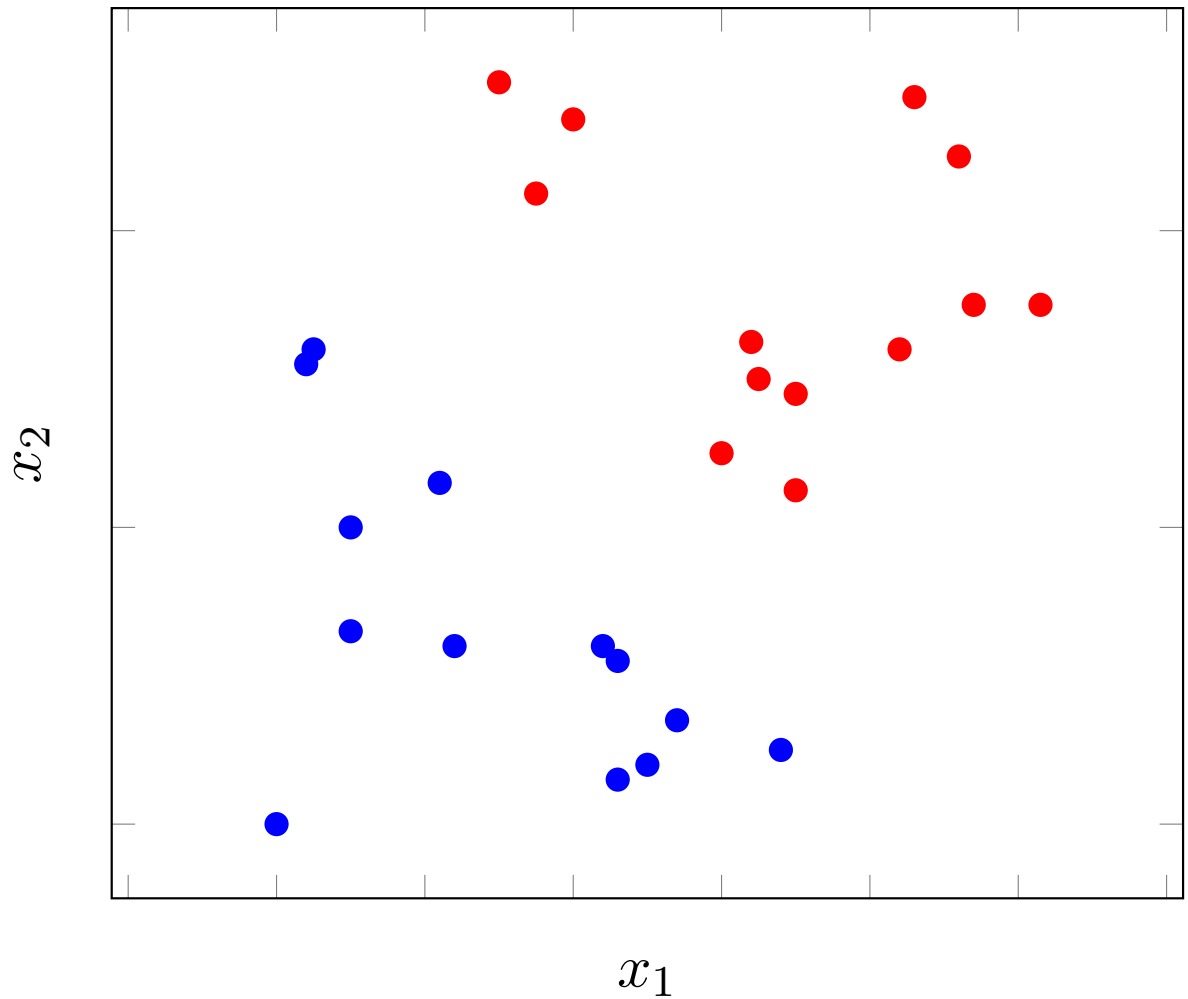
$$\frac{\partial \ell_{\text{perc}}(y, f(\boldsymbol{x}))}{\partial \boldsymbol{w}} = -y\boldsymbol{x}\,.$$

  - For a correctly classified example ($y\boldsymbol{w} \cdot \boldsymbol{x} > 0$), the gradient is $0$.
  - Therefore, the update has the form:

$$\begin{aligned} \boldsymbol{w}^t &= \boldsymbol{w}^{t-1} + \alpha y\boldsymbol{x} && \text{if} \quad y\boldsymbol{w} \cdot \boldsymbol{x} > 0 \\ \boldsymbol{w}^t &= \boldsymbol{w}^{t-1} && \text{if} \quad y\boldsymbol{w} \cdot \boldsymbol{x} \leq 0\,. \end{aligned}$$

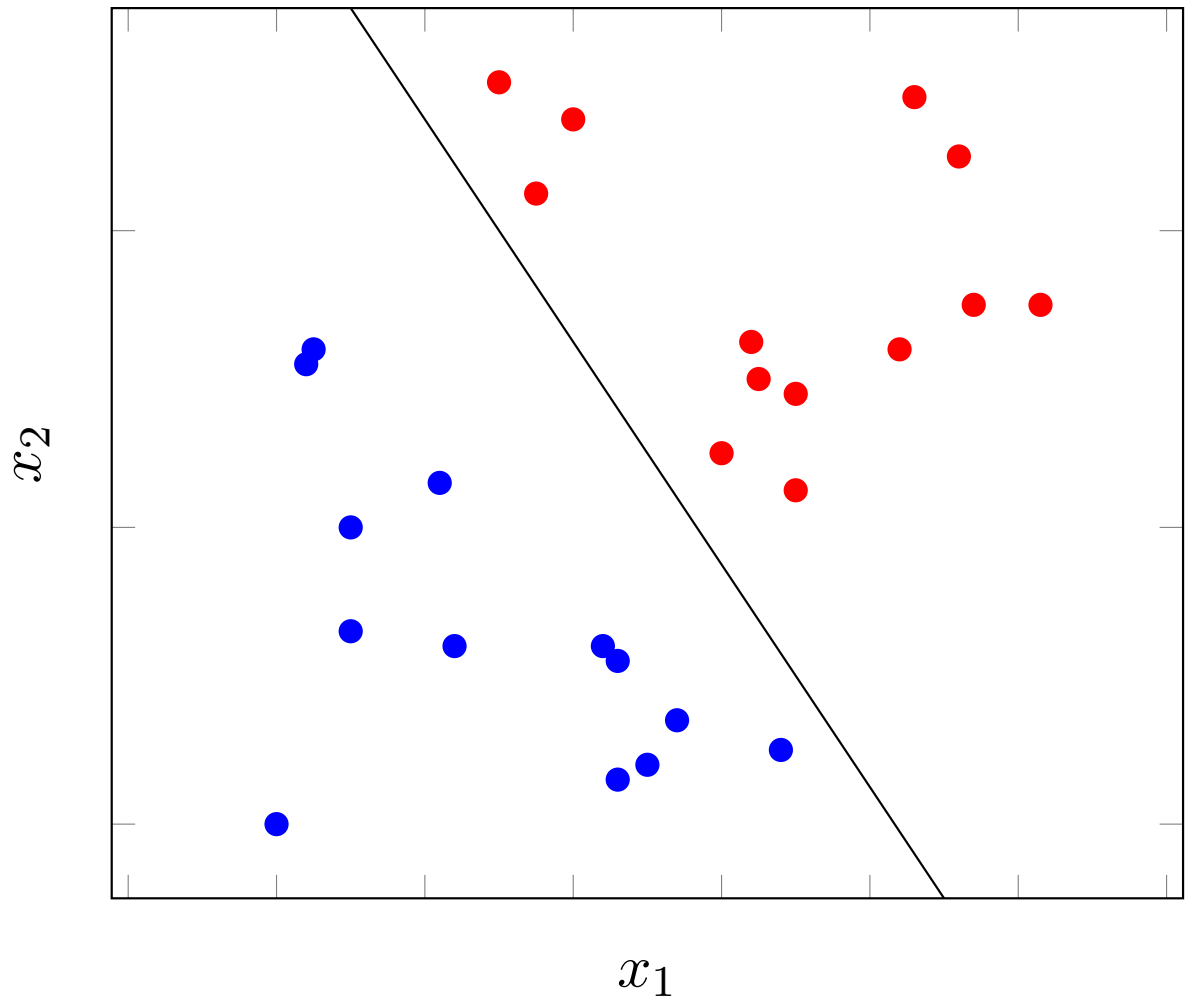
# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



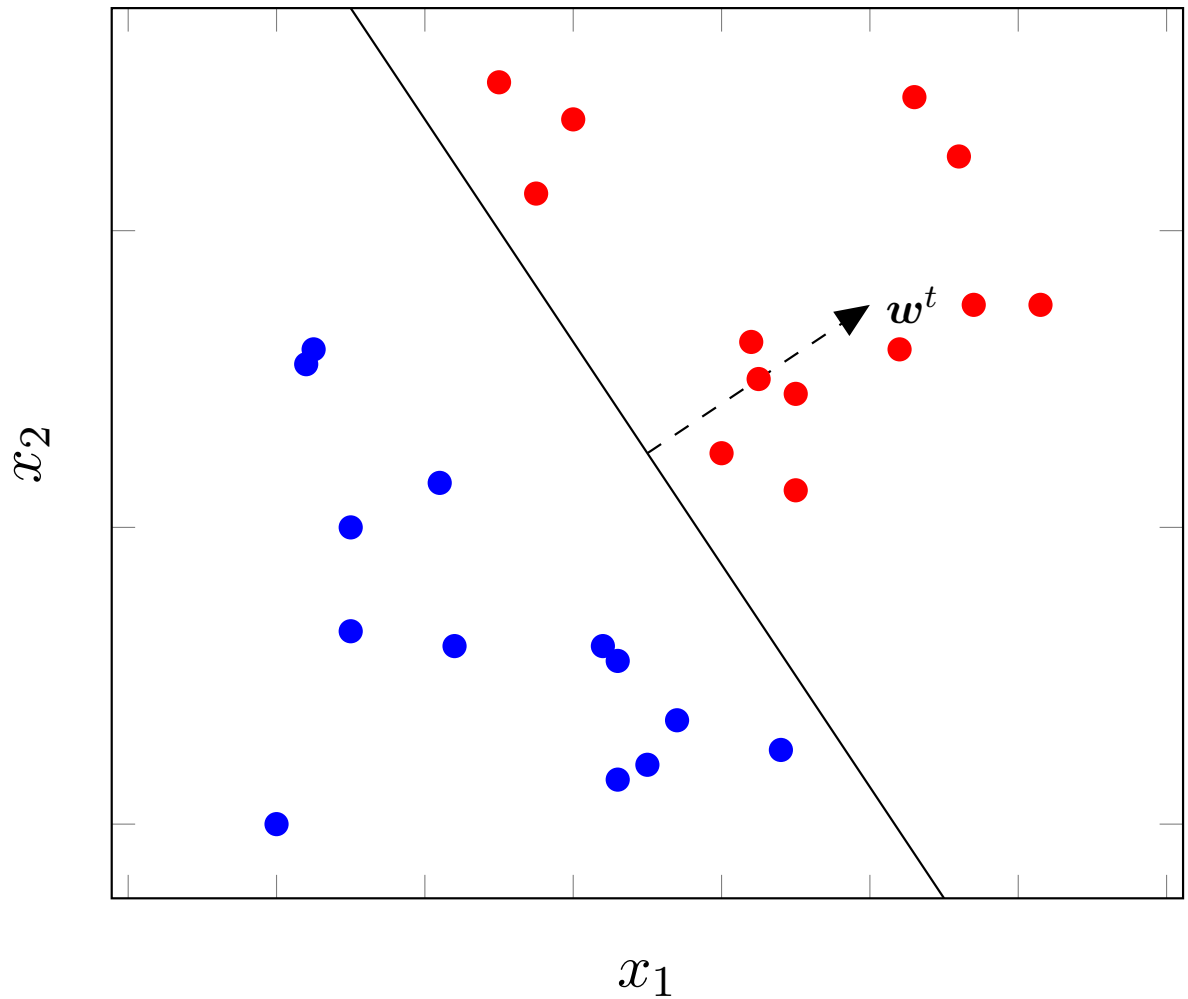
# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



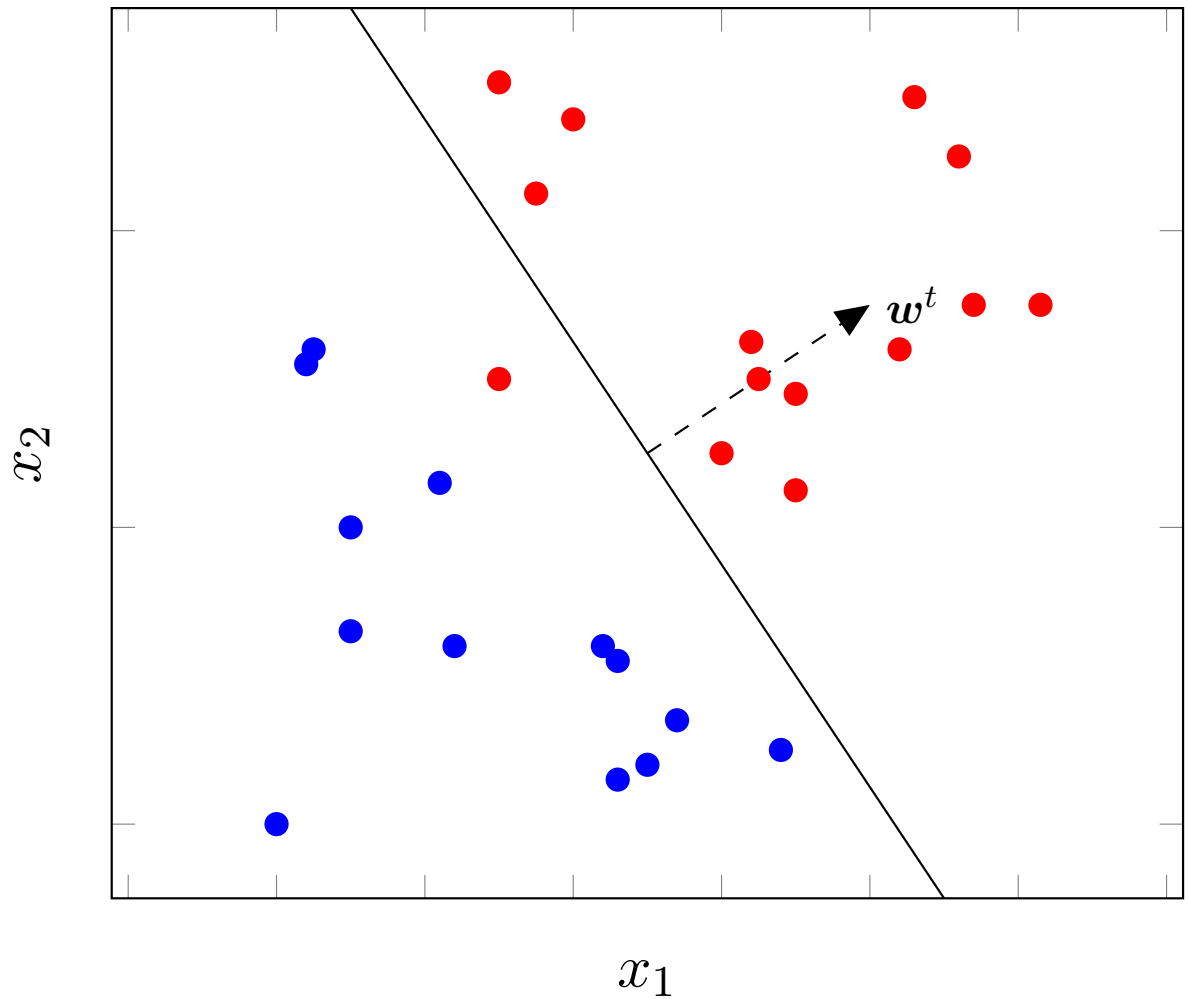
# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



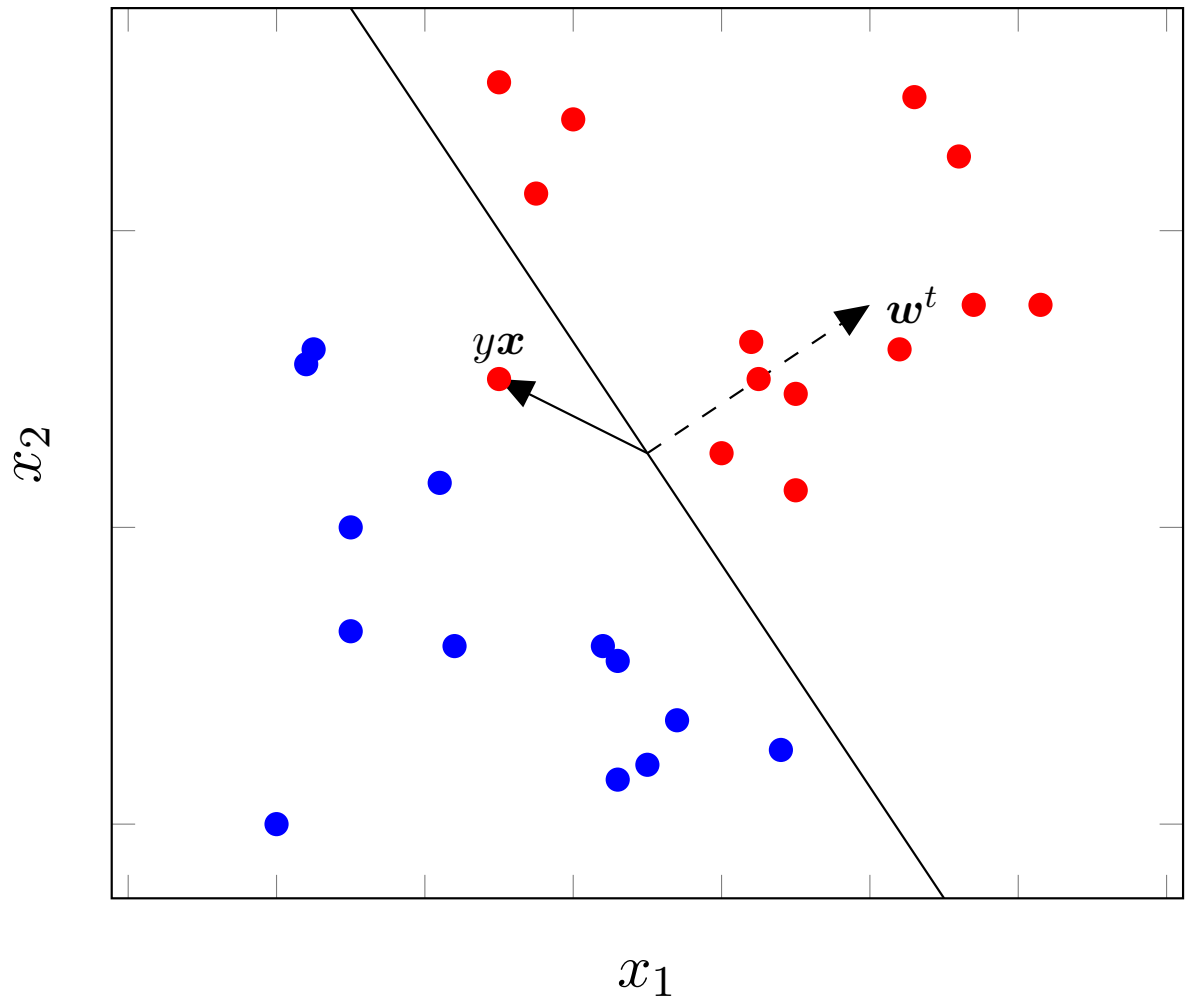
# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



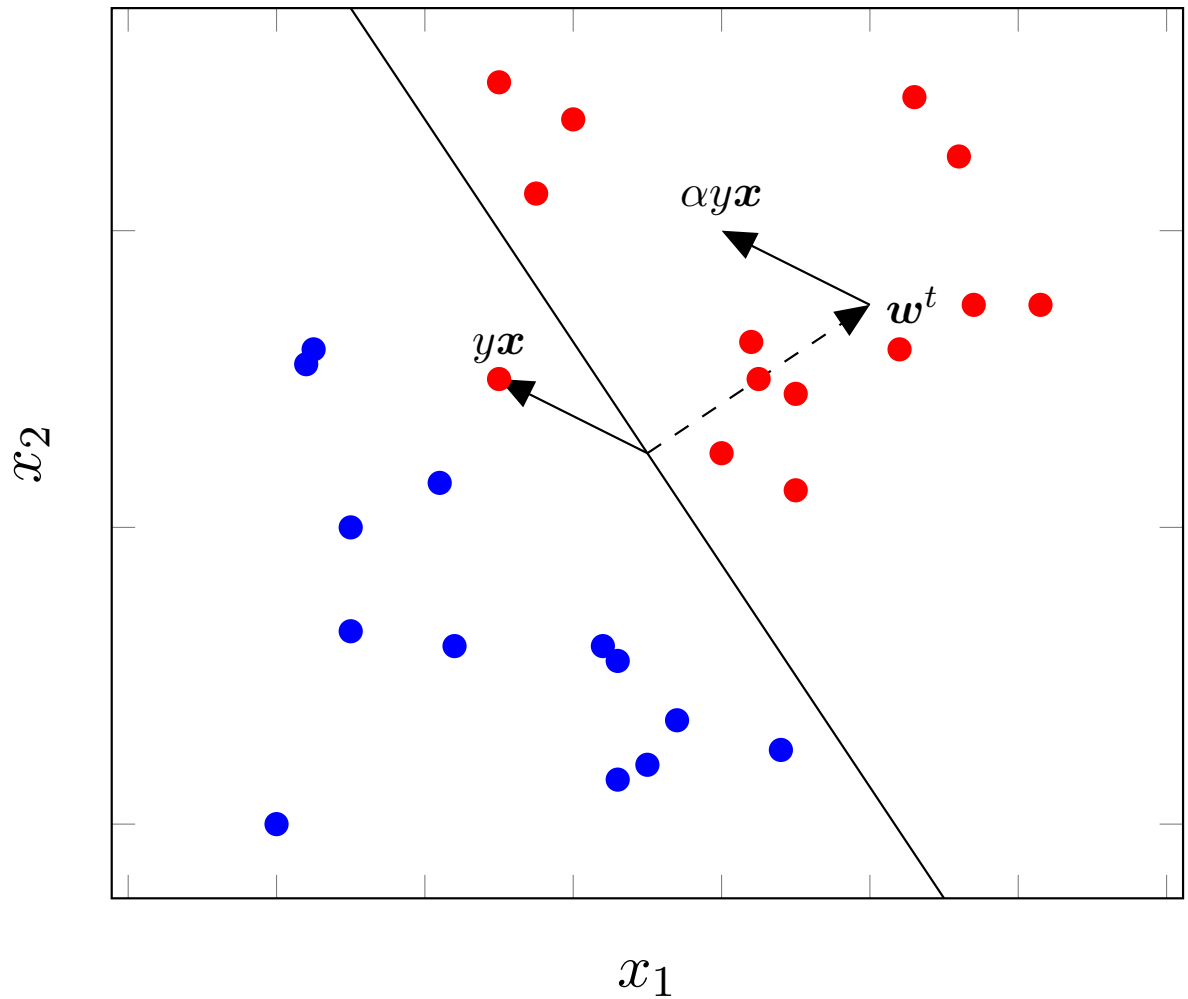
# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



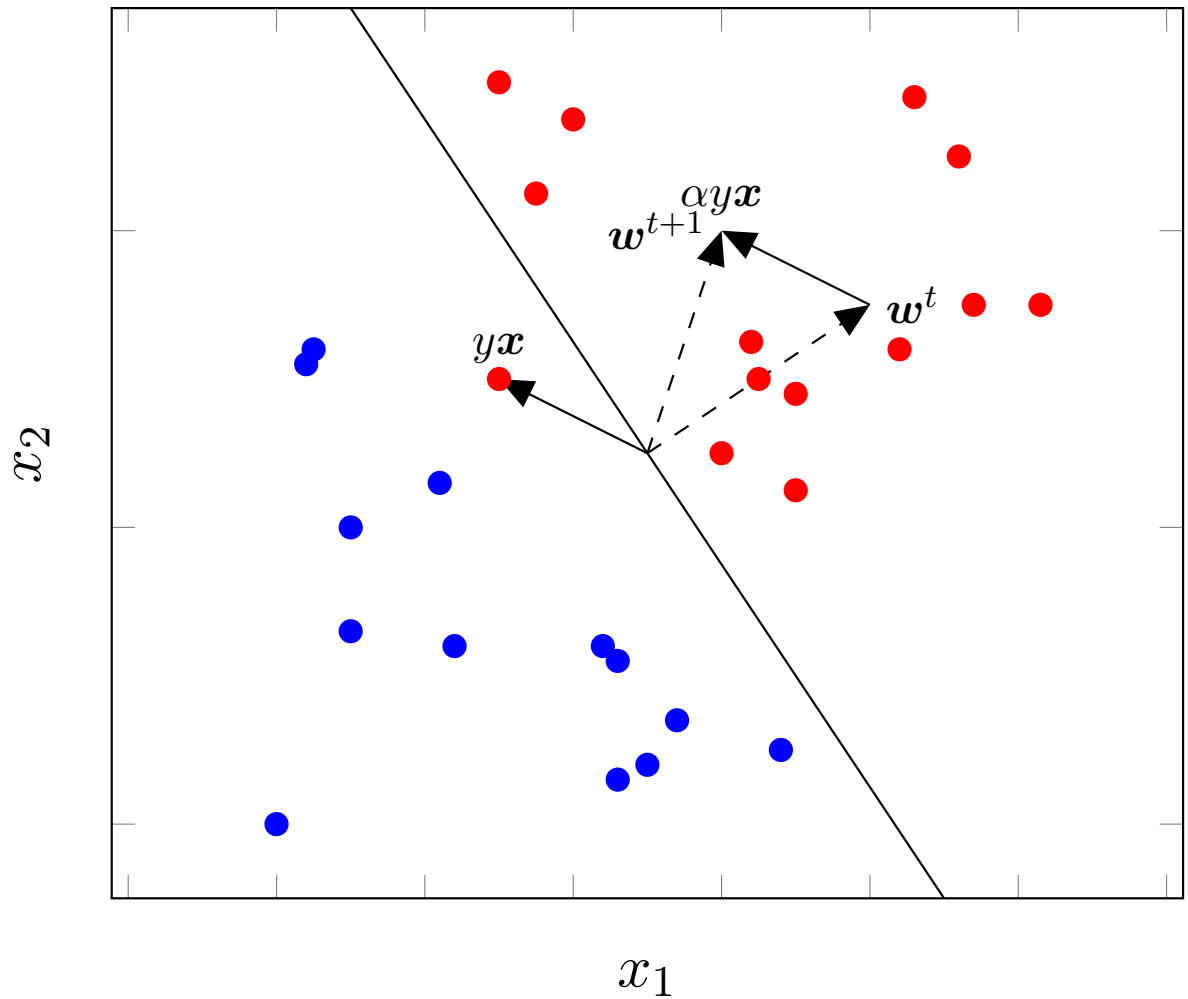
# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



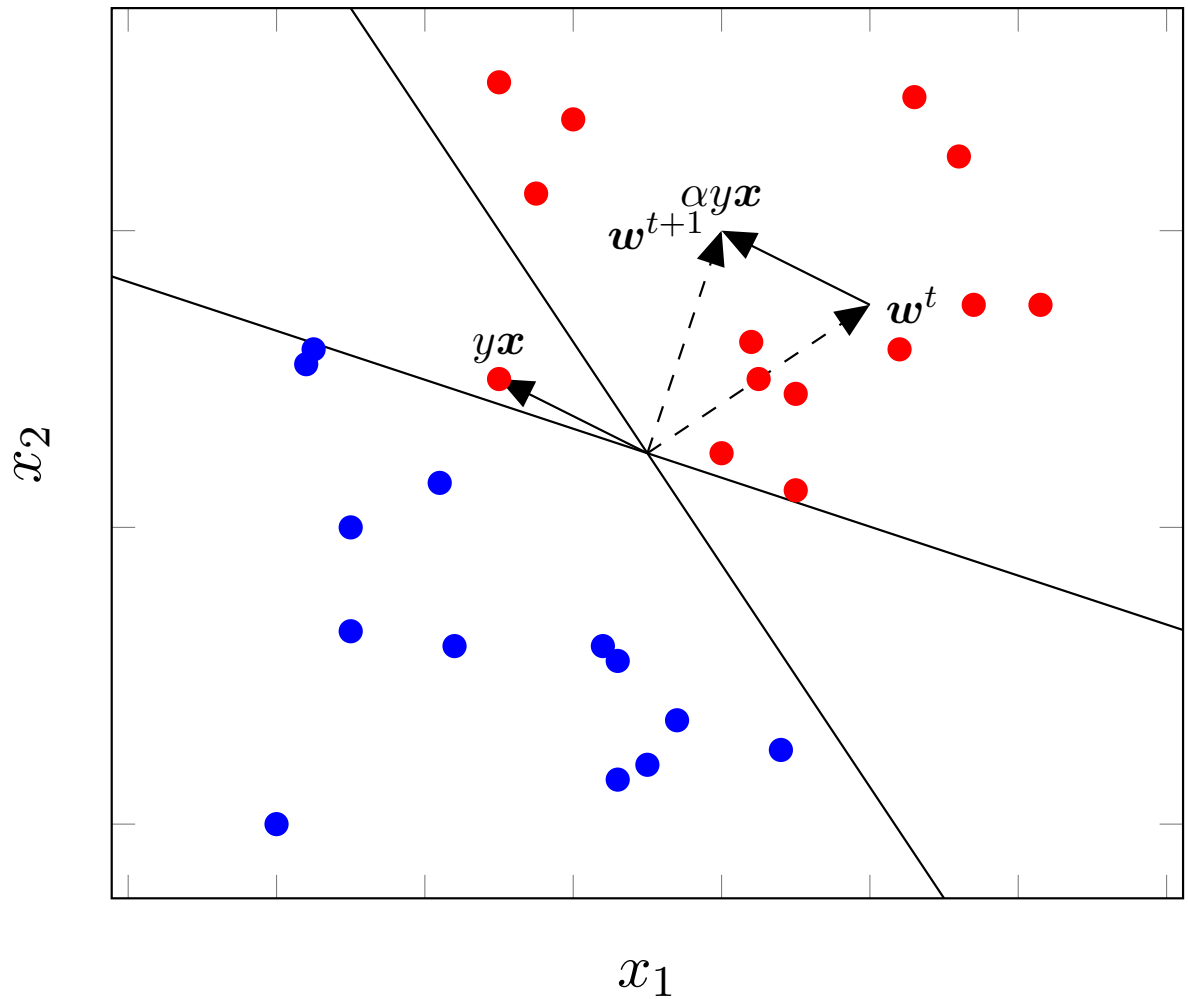
# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



# Perceptron – Graphical interpretation

- The update is simply a summation or substraction of two vectors:



# Linear models with hinge loss

- Loss function:

$$\ell_{\text{hinge}}(y, f(\boldsymbol{x})) = \max\{0, 1 - yf(\boldsymbol{x})\}$$

# Linear models with hinge loss

- Loss function:

$$\ell_{\text{hinge}}(y, f(\boldsymbol{x})) = \max\{0, 1 - yf(\boldsymbol{x})\}$$

- Linear model:

$$f(\boldsymbol{x}) = w_0 + \sum_{j=0}^{n} w_j x_j = \boldsymbol{w} \cdot \boldsymbol{x}$$

# Linear models with hinge loss

- Loss function:

$$\ell_{\text{hinge}}(y, f(\boldsymbol{x})) = \max\{0, 1 - yf(\boldsymbol{x})\}$$

- Linear model:

$$f(\boldsymbol{x}) = w_0 + \sum_{j=0}^{n} w_j x_j = \boldsymbol{w} \cdot \boldsymbol{x}$$

- Learning = fitting the model to the data by minimizing:

$$\begin{aligned}
\widehat{\boldsymbol{w}} &= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell_{\text{hinge}}(y_i, f(\boldsymbol{x}_i)) \\
&= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \max\{0, 1 - y_i f(\boldsymbol{x}_i)\} \\
&= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \max\{0, 1 - y_i \boldsymbol{w} \cdot \boldsymbol{x}_i\}
\end{aligned}$$

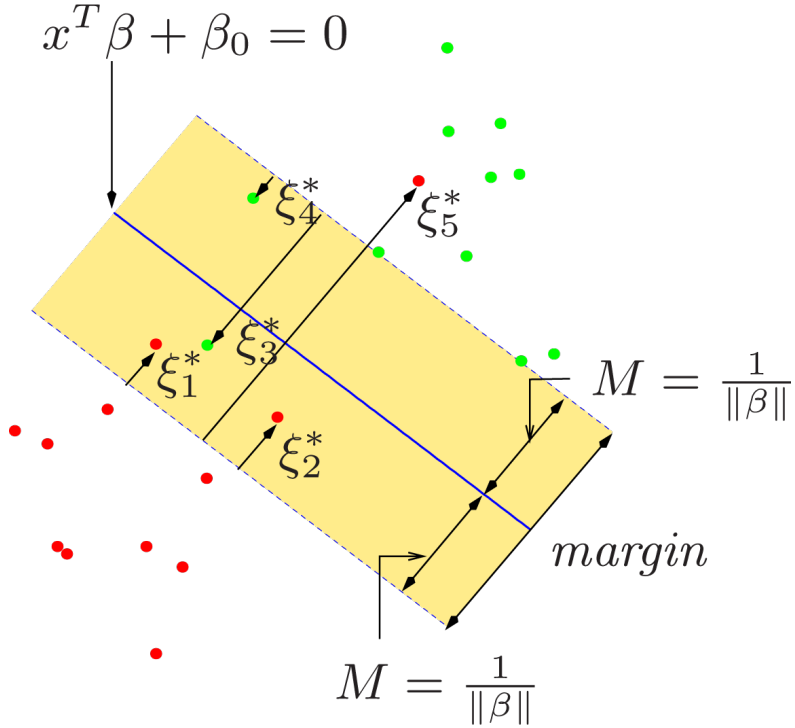
# Binary classification by Support Vector Machines (SVM)[1]

Find **maximal margin classifier**

$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i\boldsymbol{w}\cdot\boldsymbol{x} > 1-\xi_i \qquad \forall i = 1..n$$

$$\xi_i \geq 0 \qquad \forall i = 1..n$$




[1] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Conference on Learning Theory (COLT)*, pages 144–152, 1992

# Binary classification by Support Vector Machines (SVM)[1]

Find **maximal margin classifier**

$$\min_{\boldsymbol{w}} \quad \|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

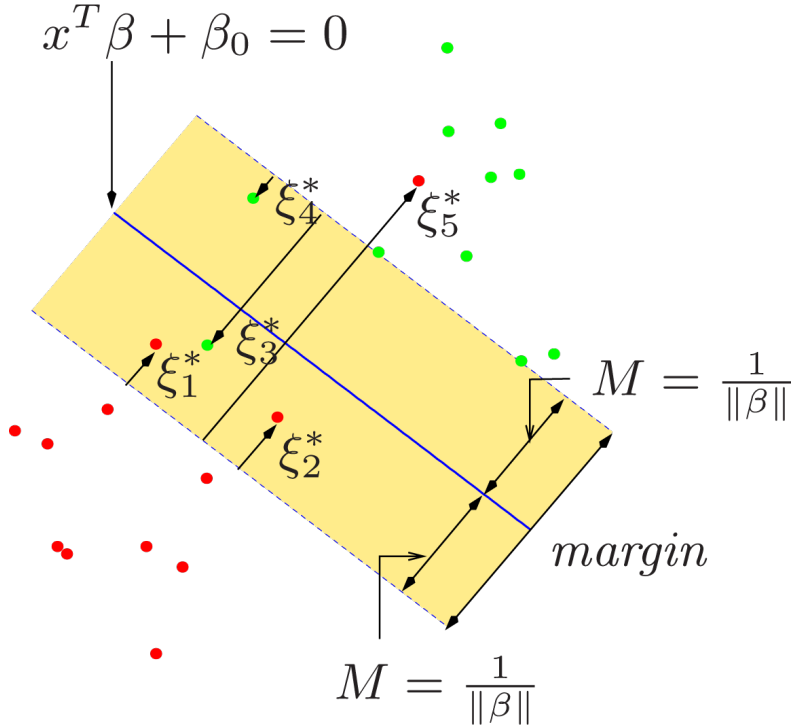
$$\text{s.t.} \quad y_i\boldsymbol{w}\cdot\boldsymbol{x} > 1-\xi_i \qquad \forall i = 1..n$$

$$\xi_i \geq 0 \qquad \forall i = 1..n$$

$$\Updownarrow$$

$$\min_{\boldsymbol{w}} \quad \sum_{i=1}^{n}\max\{0, 1-y_i\boldsymbol{w}\cdot\boldsymbol{x}_i\}$$

$$\text{s.t.} \quad \|\boldsymbol{w}\|^2 \leq B \quad \text{for some } B.$$

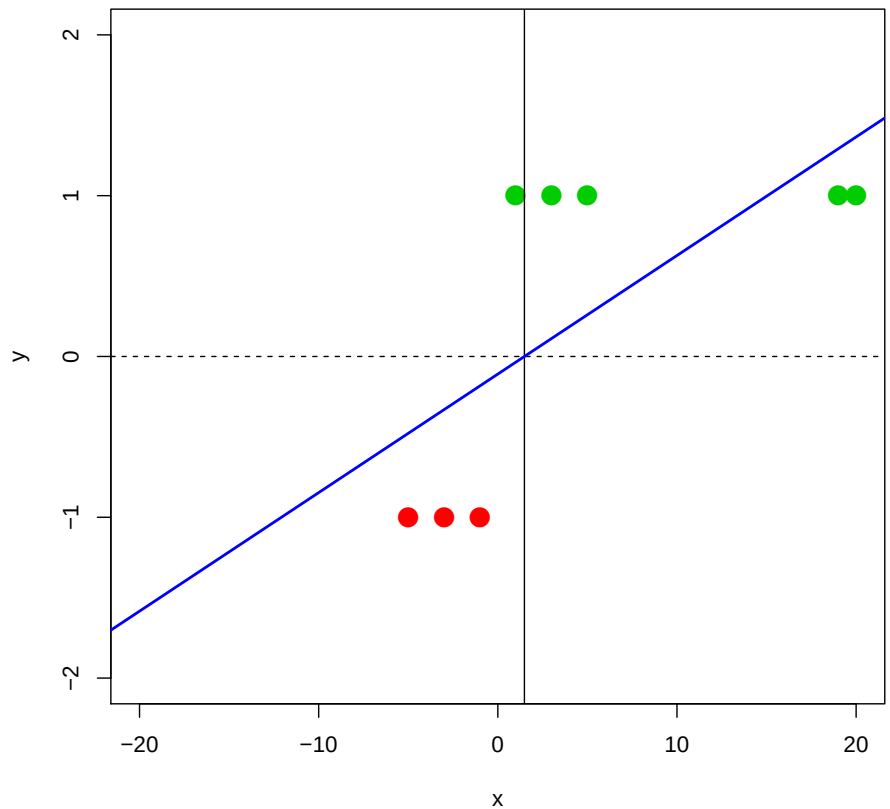



[1] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Conference on Learning Theory (COLT)*, pages 144–152, 1992

# Logistic regression

- Another option is to use a sigmoid (or logistic) transformation of the linear function:

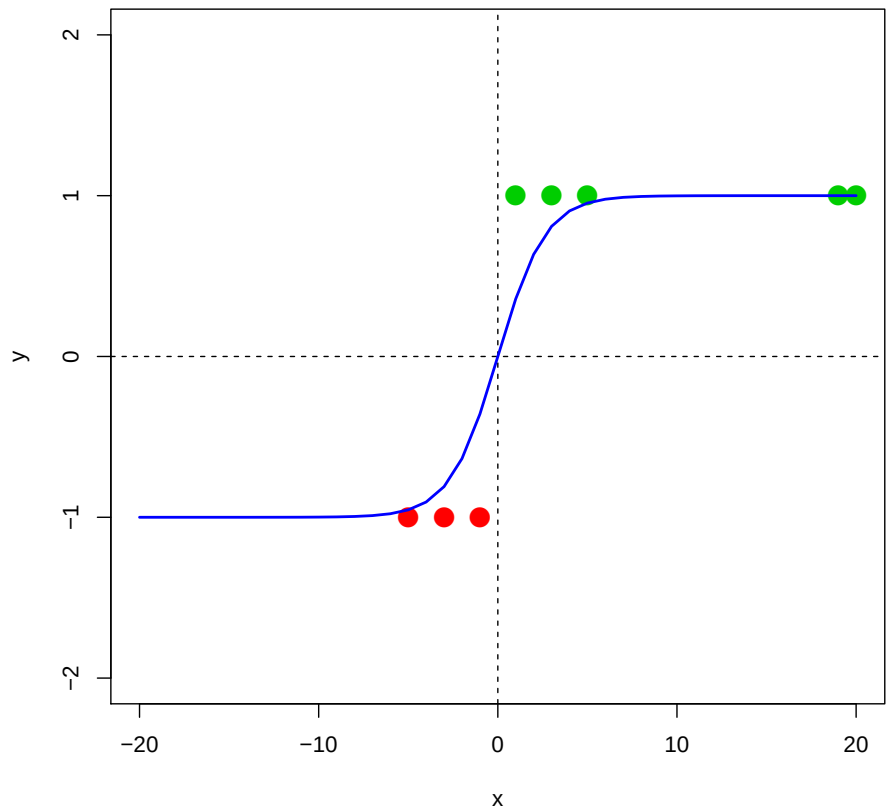
$$g(\boldsymbol{x}) = \frac{1}{1+\exp(-f(\boldsymbol{x}))} \in (0,1) \quad g(\boldsymbol{x}) = \frac{1-\exp(-f(\boldsymbol{x}))}{1+\exp(-f(\boldsymbol{x}))} \in (-1,1)$$



# Logistic regression

- Another option is to use a sigmoid (or logistic) transformation of the linear function:

$$g(\boldsymbol{x}) = \frac{1}{1+\exp(-f(\boldsymbol{x}))} \in (0,1) \quad g(\boldsymbol{x}) = \frac{1-\exp(-f(\boldsymbol{x}))}{1+\exp(-f(\boldsymbol{x}))} \in (-1,1)$$



# Logistic regression – motivation

- Get an estimate of $\eta(\boldsymbol{x}) = P(y = 1 \,|\, \boldsymbol{x})$ from $f(\boldsymbol{x})$.

# Logistic regression – motivation

- Get an estimate of $\eta(\boldsymbol{x}) = P(y = 1 \,|\, \boldsymbol{x})$ from $f(\boldsymbol{x})$.
  - $\eta(\boldsymbol{x}) \in [0, 1]$, while $f(\boldsymbol{x}) \in \mathbb{R}$.

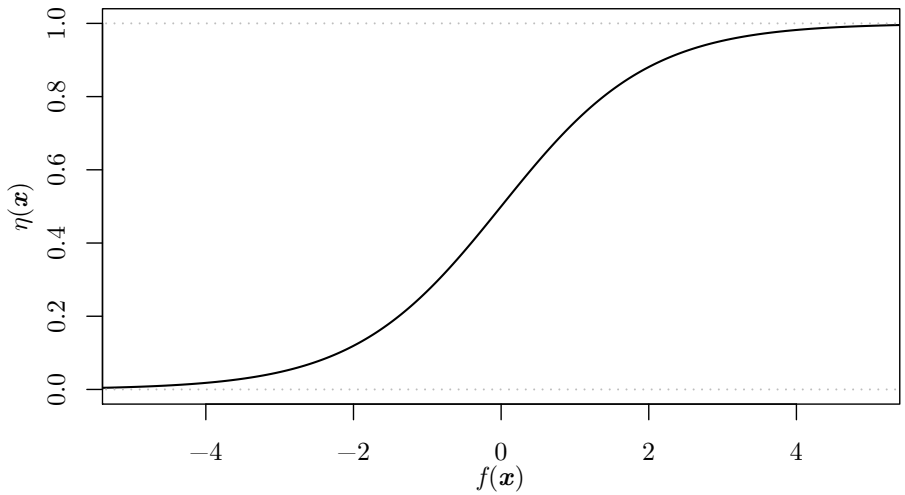
# Logistic regression – motivation

- Get an estimate of $\eta(\boldsymbol{x}) = P(y = 1 \,|\, \boldsymbol{x})$ from $f(\boldsymbol{x})$.
  - $\eta(\boldsymbol{x}) \in [0, 1]$, while $f(\boldsymbol{x}) \in \mathbb{R}$.
  - A natural candidate for function $f(\boldsymbol{x}) \mapsto \eta(\boldsymbol{x})$: **sigmoid function**

$$\eta(\boldsymbol{x}) = \frac{1}{1 + \exp(-f(\boldsymbol{x}))}$$

$$\Updownarrow$$

$$f(\boldsymbol{x}) = \log \frac{\eta(\boldsymbol{x})}{1 - \eta(\boldsymbol{x})}$$



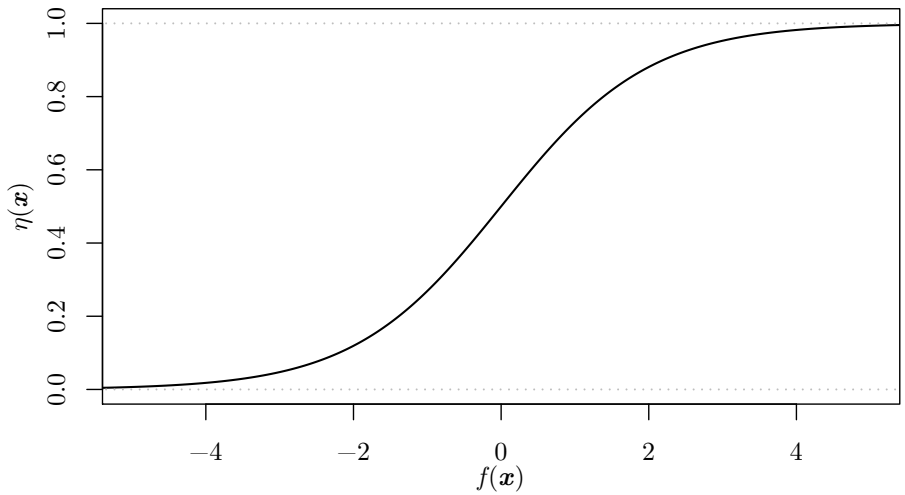
# Logistic regression – motivation

- Get an estimate of $\eta(\boldsymbol{x}) = P(y = 1 \,|\, \boldsymbol{x})$ from $f(\boldsymbol{x})$.
  - $\eta(\boldsymbol{x}) \in [0, 1]$, while $f(\boldsymbol{x}) \in \mathbb{R}$.
  - A natural candidate for function $f(\boldsymbol{x}) \mapsto \eta(\boldsymbol{x})$: **sigmoid function**

$$\eta(\boldsymbol{x}) = \frac{1}{1 + \exp(-f(\boldsymbol{x}))}$$

$$\Updownarrow$$

$$f(\boldsymbol{x}) = \log \frac{\eta(\boldsymbol{x})}{1 - \eta(\boldsymbol{x})}$$




## Question

Show that if examples are generated by $y \sim P(y)$ and $\boldsymbol{x}|y \sim N(\mu_y|\Sigma)$ (with shared covariance matrix), then $\log \frac{\eta(\boldsymbol{x})}{1-\eta(\boldsymbol{x})}$ is a linear function of $\boldsymbol{x}$.

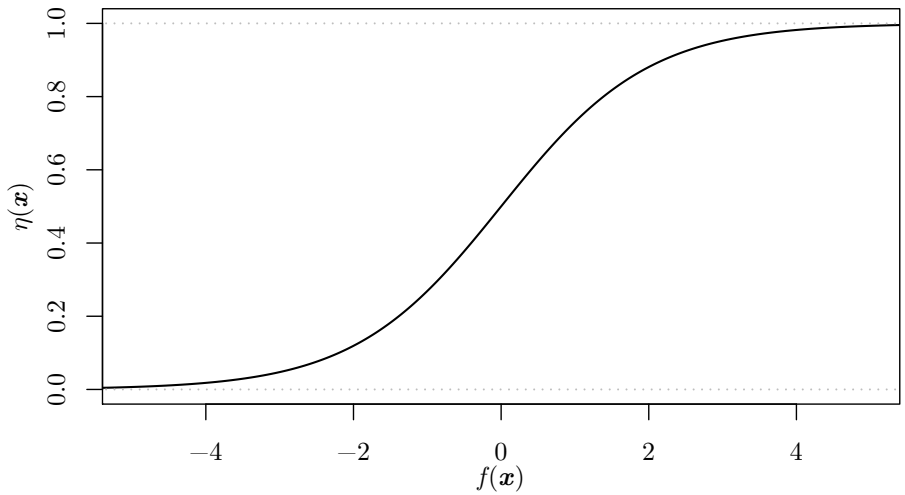
# Logistic regression – motivation

- Get an estimate of $\eta(\boldsymbol{x}) = P(y = 1 \,|\, \boldsymbol{x})$ from $f(\boldsymbol{x})$.
  - $\eta(\boldsymbol{x}) \in [0, 1]$, while $f(\boldsymbol{x}) \in \mathbb{R}$.
  - A natural candidate for function $f(\boldsymbol{x}) \mapsto \eta(\boldsymbol{x})$: **sigmoid function**

$$\eta(\boldsymbol{x}) = \frac{1}{1 + \exp(-f(\boldsymbol{x}))}$$

$$\Updownarrow$$

$$f(\boldsymbol{x}) = \log \frac{\eta(\boldsymbol{x})}{1 - \eta(\boldsymbol{x})}$$




- Solved by the method of **Maximum Likelihood**.

$$\begin{aligned}
\widehat{f} &= \arg\max_{f \in \mathcal{F}} P_f(y_1, \ldots, y_n | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \\
&= \arg\min_{f \in \mathcal{F}} - \log P_f(y_1, \ldots, y_n | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \\
&= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} - \log P_f(y_i \,|\, \boldsymbol{x}_i)
\end{aligned}$$

# Logistic regression

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} -\log P_f(y_i \mid \boldsymbol{x}_i)$$

# Logistic regression

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} - \log P_f(y_i \mid \boldsymbol{x}_i)$$

$$= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \Big( - [\![ y_i = 1 ]\!] \log \eta(\boldsymbol{x}_i) - [\![ y_i = -1 ]\!] \log(1 - \eta(\boldsymbol{x}_i)) \Big)$$

$\eta(\boldsymbol{x}) = P(y = 1 | \boldsymbol{x})$

# Logistic regression

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} - \log P_f(y_i \mid \boldsymbol{x}_i)$$

$$\eta(\boldsymbol{x}) = P(y = 1 | \boldsymbol{x})$$

$$= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \Big( - [\![ y_i = 1 ]\!] \log \eta(\boldsymbol{x}_i) - [\![ y_i = -1 ]\!] \log(1 - \eta(\boldsymbol{x}_i)) \Big)$$

$$= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \left( [\![ y_i = 1 ]\!] \log \left( 1 + e^{-f(\boldsymbol{x}_i)} \right) + [\![ y_i = -1 ]\!] \log \left( 1 + e^{f(\boldsymbol{x}_i)} \right) \right)$$

$$\eta(\boldsymbol{x}) = \left(1 + e^{-f(\boldsymbol{x})}\right)^{-1}$$

$$1 - \eta(\boldsymbol{x}) = 1 - \frac{1}{1+e^{-f(\boldsymbol{x})}} = \frac{e^{-f(\boldsymbol{x})}}{1+e^{-f(\boldsymbol{x})}} = \left(1 + e^{f(\boldsymbol{x})}\right)^{-1}$$

# Logistic regression

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} -\log P_f(y_i \mid \boldsymbol{x}_i)$$

$$\eta(\boldsymbol{x}) = P(y = 1 | \boldsymbol{x})$$

$$= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \Big( -[\![y_i = 1]\!] \log \eta(\boldsymbol{x}_i) - [\![y_i = -1]\!] \log(1 - \eta(\boldsymbol{x}_i)) \Big)$$

$$= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \left( [\![y_i = 1]\!] \log \left( 1 + e^{-f(\boldsymbol{x}_i)} \right) + [\![y_i = -1]\!] \log \left( 1 + e^{f(\boldsymbol{x}_i)} \right) \right)$$

$$= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \log \left( 1 + e^{-y_i f(\boldsymbol{x}_i)} \right).$$

$$\eta(\boldsymbol{x}) = \left(1 + e^{-f(\boldsymbol{x})}\right)^{-1}$$

$$1 - \eta(\boldsymbol{x}) = 1 - \frac{1}{1+e^{-f(\boldsymbol{x})}} = \frac{e^{-f(\boldsymbol{x})}}{1+e^{-f(\boldsymbol{x})}} = \left(1 + e^{f(\boldsymbol{x})}\right)^{-1}$$

# Logistic regression

$$\begin{aligned}\widehat{f} &= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right) \\ &= \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right).\end{aligned}$$

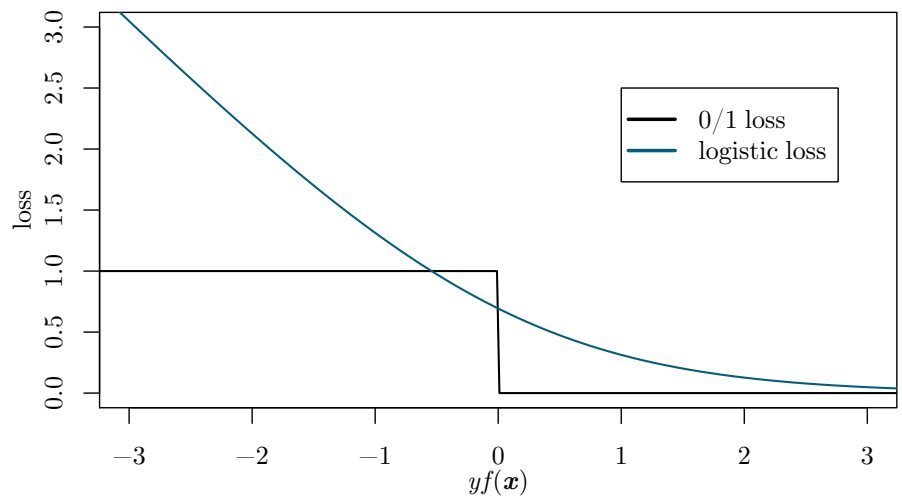
# Logistic regression

$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right)$$

$$= \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right).$$

- Effectively, we replaced $0/1$ loss with **logistic loss**:

$$\ell_{\log}(y, f(\boldsymbol{x})) = \log\left(1 + e^{-y f(\boldsymbol{x})}\right).$$



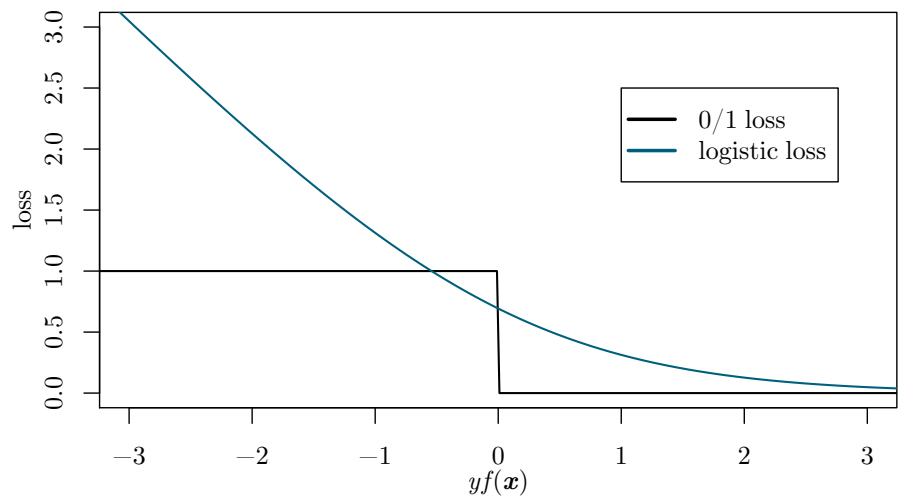
# Logistic regression

$$\widehat{f} = \underset{f \in \mathcal{F}}{\arg\min} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right)$$

$$= \underset{f \in \mathcal{F}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right).$$

- Effectively, we replaced $0/1$ loss with **logistic loss**:

$$\ell_{\log}(y, f(\boldsymbol{x})) = \log\left(1 + e^{-yf(\boldsymbol{x})}\right).$$




- Commonly used, better than least squares in practice.

## Learning algorithm for logistic regression

- Let $f(\boldsymbol{x})$ be a linear function of the input attributes:

$$f(\boldsymbol{x}) = w_0 + \sum_{j=1}^{m} w_j x_j = \boldsymbol{w} \cdot \boldsymbol{x}.$$

- The task of the learning algorithm is to solve:

$$\begin{aligned} \widehat{\boldsymbol{w}} &= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \ell_{\log}\left(y_i, \boldsymbol{w} \cdot \boldsymbol{x}_i\right) \\ &= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i\right)\right). \end{aligned}$$

- This problem is usually solved using iterative convex optimization algorithms.

# Logistic regression

- Consider a simple gradient descent algorithm.
- Total loss over the training examples:

$$\widehat{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \boldsymbol{w} \cdot \boldsymbol{x}\right)\right)$$

- The gradient descent algorithm:
  - Initialize $\boldsymbol{w}^0$, e.g. by $\boldsymbol{w}^0 = \mathbf{0}$
  - Repeat until convergence:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \alpha \frac{\partial \widehat{L}(\boldsymbol{w}^t)}{\partial \boldsymbol{w}^t}$$

where $\alpha$ is the step size (learning rate) and

$$\frac{\partial \widehat{L}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \left(\frac{\partial \widehat{L}}{\partial w_1}, \ldots, \frac{\partial \widehat{L}}{\partial w_m}\right)$$

# Logistic regression

- Compute the partial derivative of the loss with respect to $w_j$:

$$\frac{\partial \widehat{L}}{\partial w_j} =$$

# Logistic regression

- Compute the partial derivative of the loss with respect to $w_j$:

$$\frac{\partial \widehat{L}}{\partial w_j} = -\sum_{i=1}^{n} \frac{\exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i) y_i x_{ij}}{1 + \exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)}$$

# Logistic regression

- Compute the partial derivative of the loss with respect to $w_j$:

$$\begin{aligned}\frac{\partial \widehat{L}}{\partial w_j} &= -\sum_{i=1}^{n} \frac{\exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i) y_i x_{ij}}{1 + \exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)} \\ &= -\sum_{i=1}^{n} \frac{y_i x_{ij}}{1 + \exp(y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)}\end{aligned}$$

# Logistic regression

- Compute the partial derivative of the loss with respect to $w_j$:

$$\begin{aligned}
\frac{\partial \widehat{L}}{\partial w_j} &= -\sum_{i=1}^{n} \frac{\exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i) y_i x_{ij}}{1 + \exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)} \\
&= -\sum_{i=1}^{n} \frac{y_i x_{ij}}{1 + \exp(y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)} \\
&= -\sum_{i=1}^{n} \beta_i y_i x_{ij}, \qquad \text{where} \quad \beta_i = \frac{1}{1 + \exp(y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)}
\end{aligned}$$

# Logistic regression

- Compute the partial derivative of the loss with respect to $w_j$:

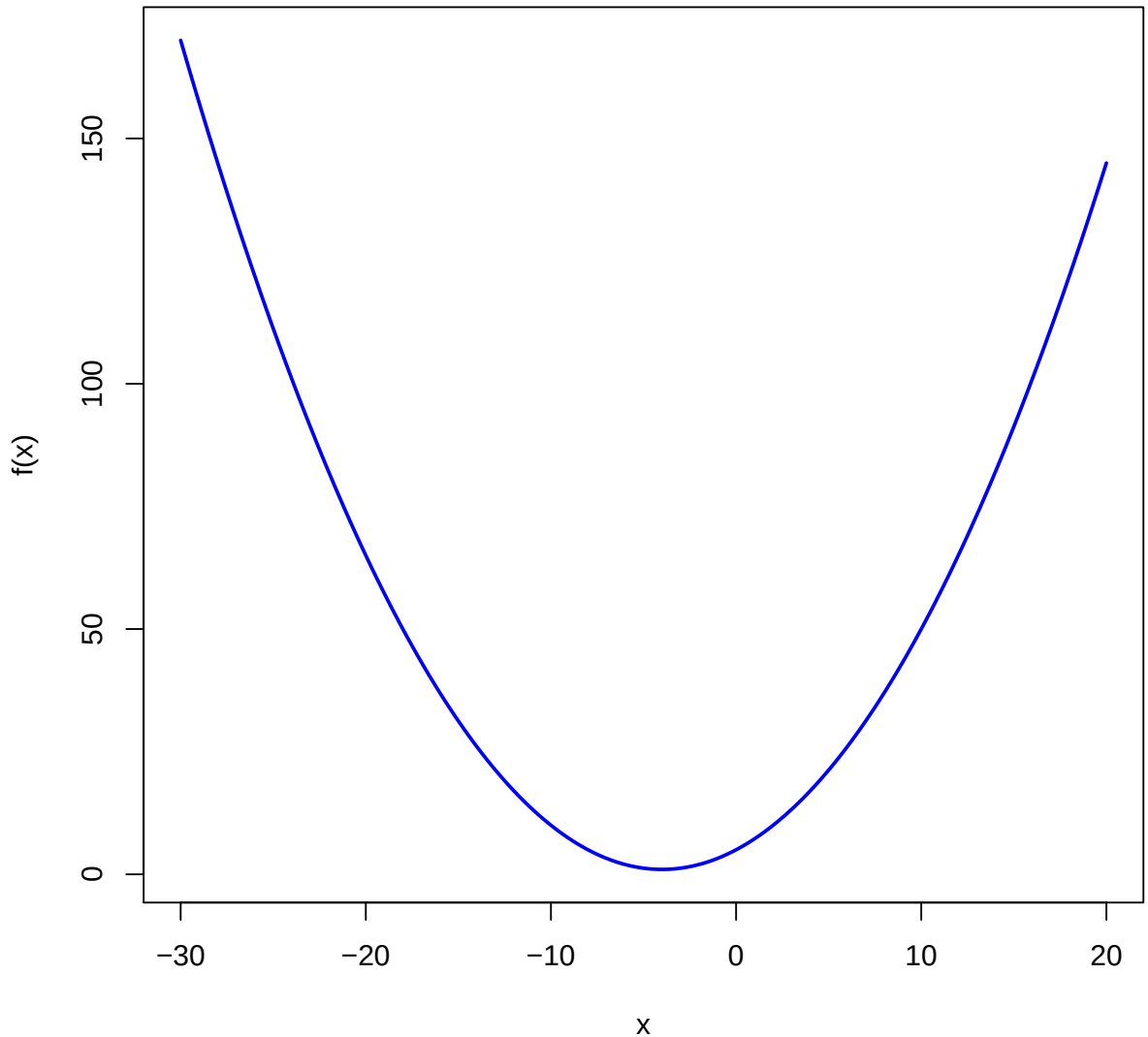
$$\begin{aligned}
\frac{\partial \widehat{L}}{\partial w_j} &= -\sum_{i=1}^{n} \frac{\exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i) y_i x_{ij}}{1 + \exp(-y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)} \\
&= -\sum_{i=1}^{n} \frac{y_i x_{ij}}{1 + \exp(y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)} \\
&= -\sum_{i=1}^{n} \beta_i y_i x_{ij}, \qquad \text{where} \quad \beta_i = \frac{1}{1 + \exp(y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)}
\end{aligned}$$

- Gradient:

$$\frac{\partial \widehat{L}(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\sum_{i=1}^{n} \beta_i y_i \boldsymbol{x}_i, \qquad \text{where} \quad \beta_i = \frac{1}{1 + \exp(y_i \boldsymbol{w} \cdot \boldsymbol{x}_i)}$$

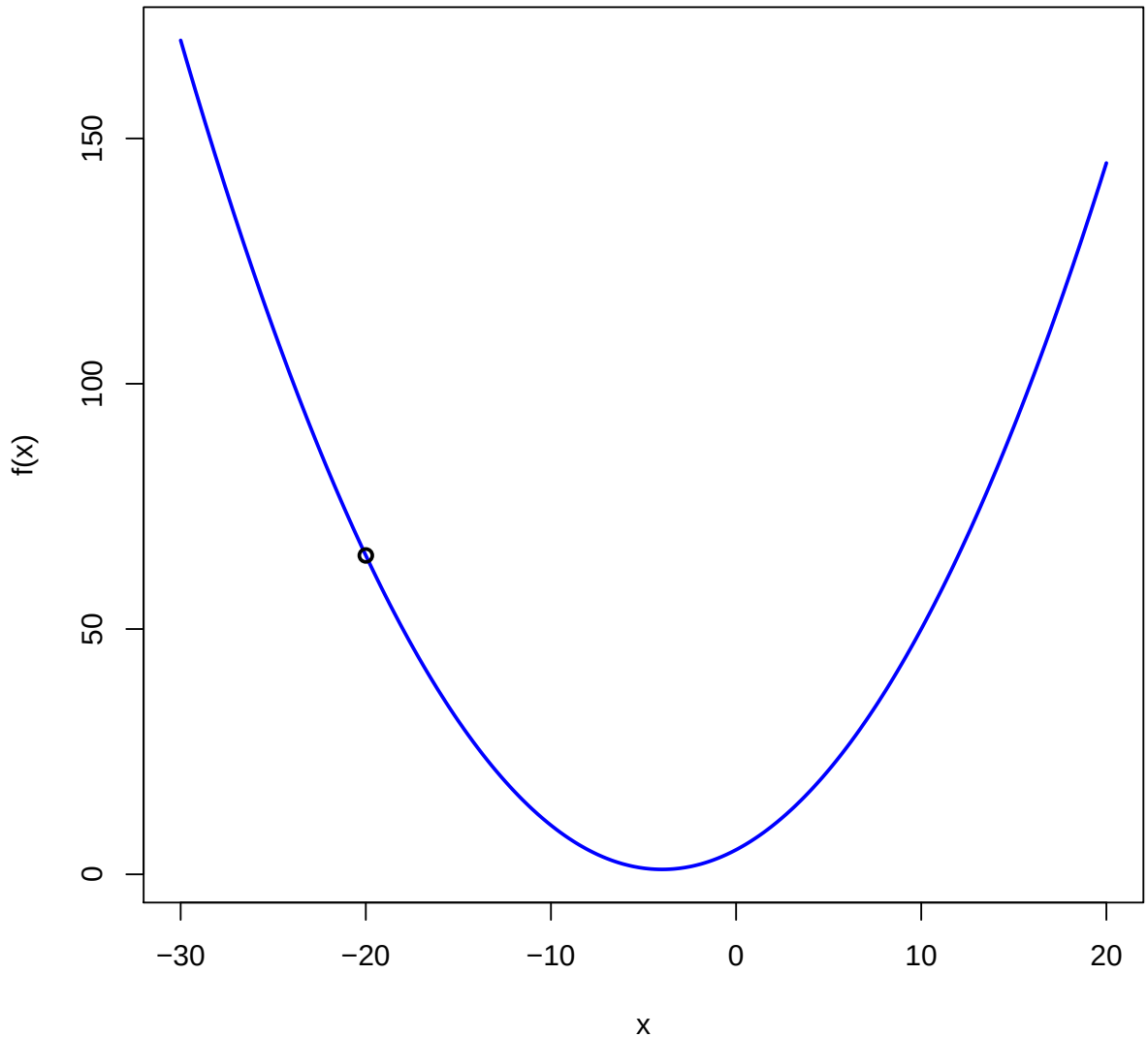
# Logistic regression

- Gradient descent example:



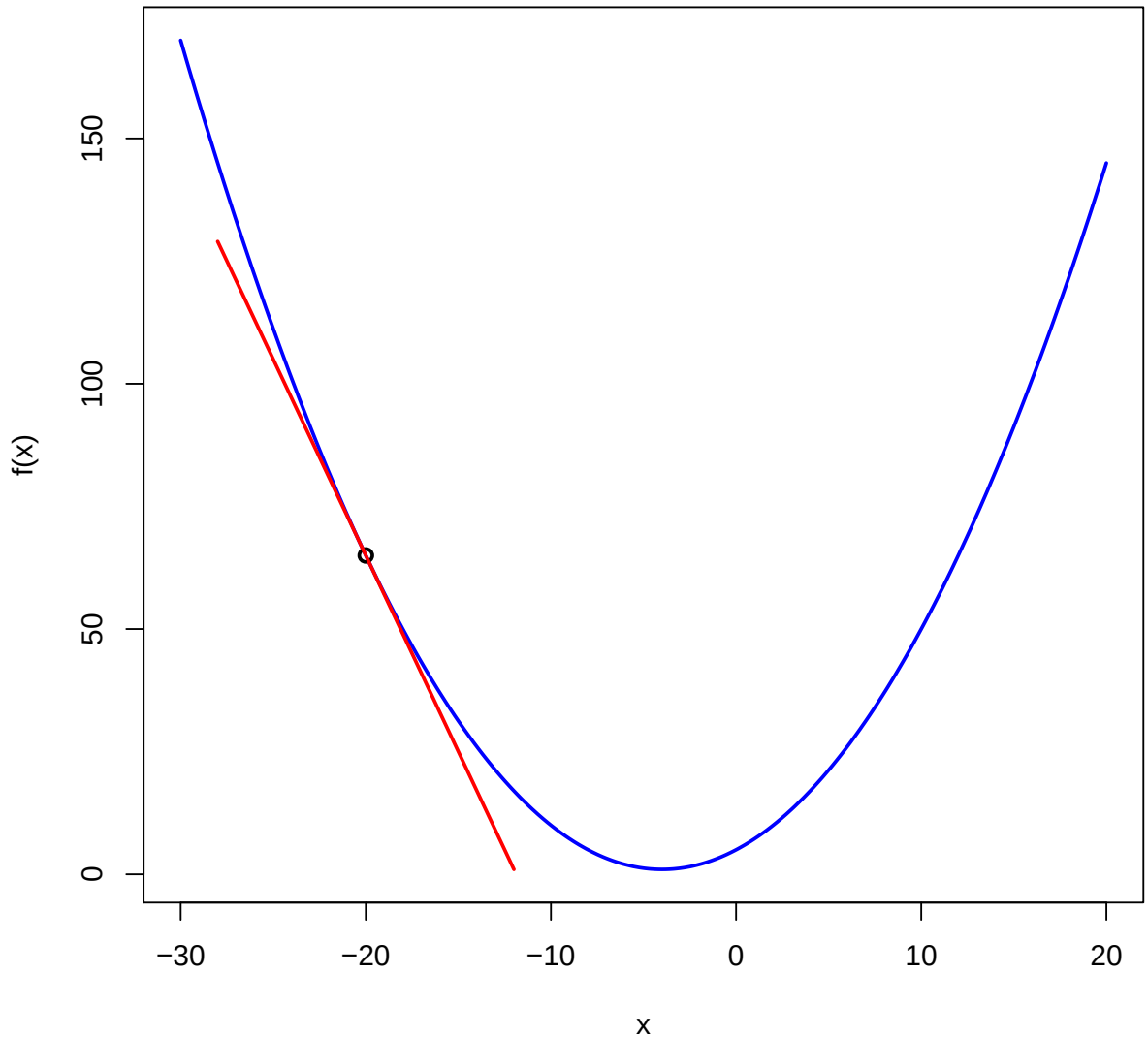
# Logistic regression

- Gradient descent example:



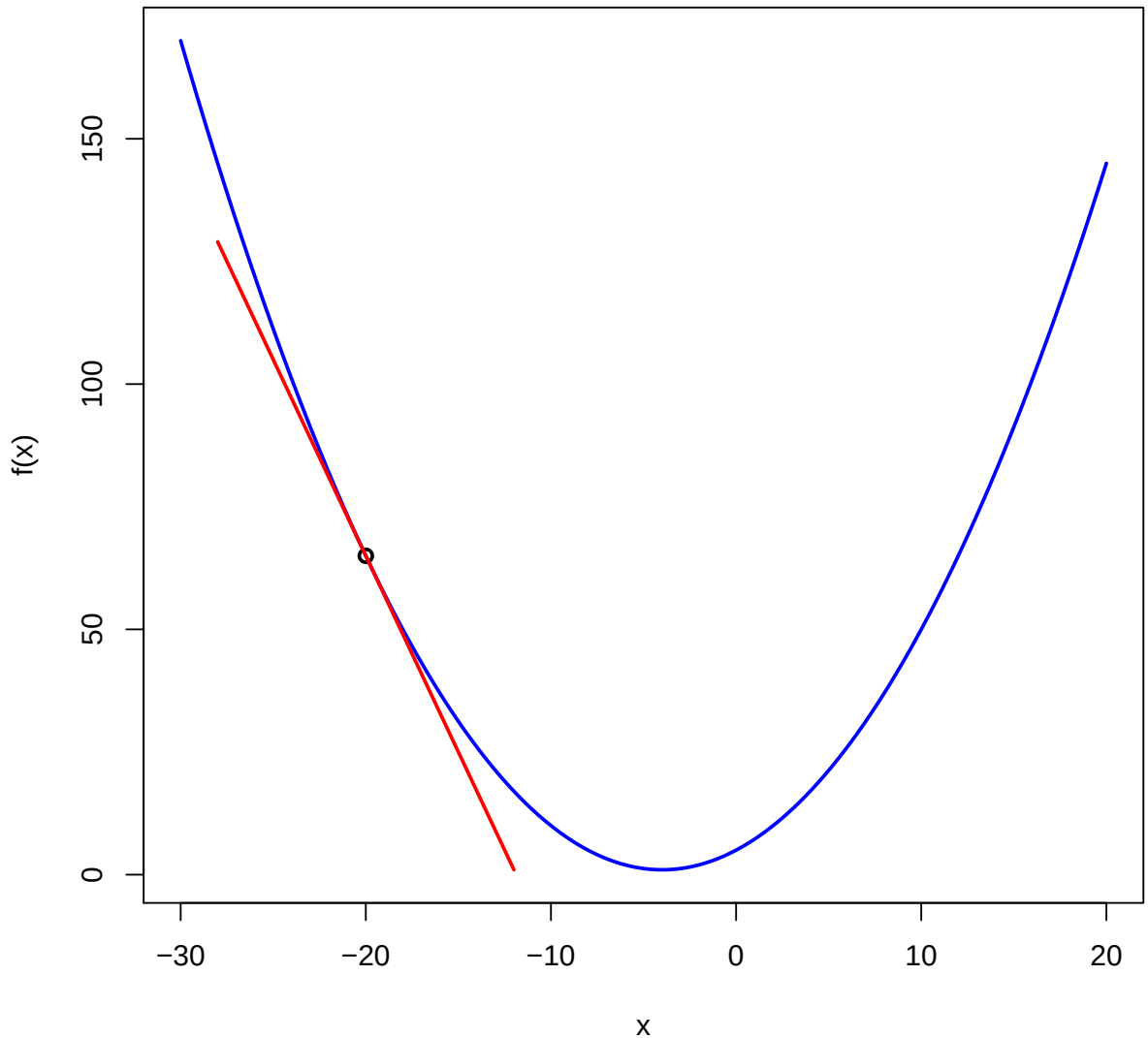
# Logistic regression

- Gradient descent example:



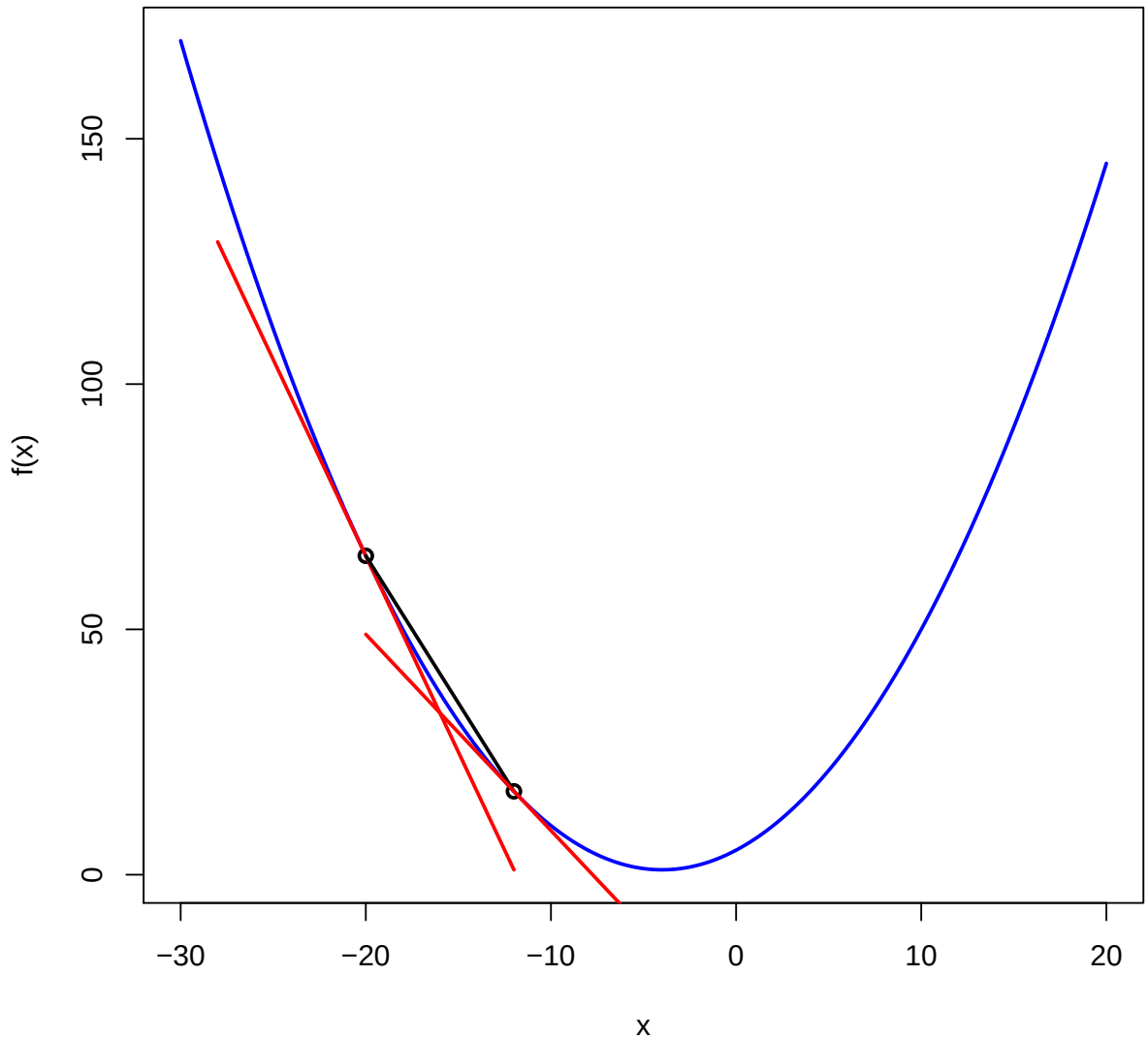
# Logistic regression

- Gradient descent example ($\alpha = 1$):



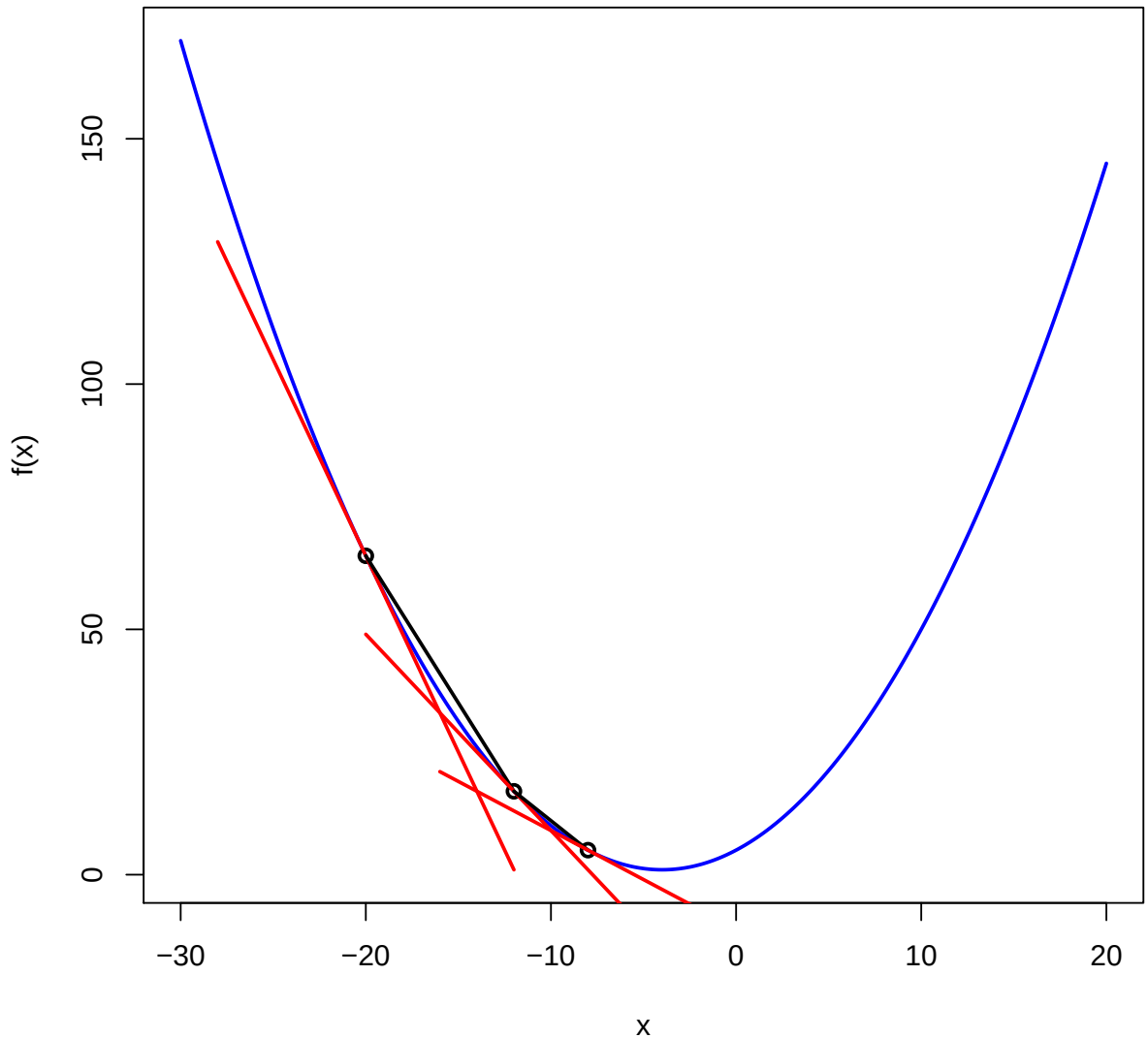
# Logistic regression

- Gradient descent example ($\alpha = 1$):



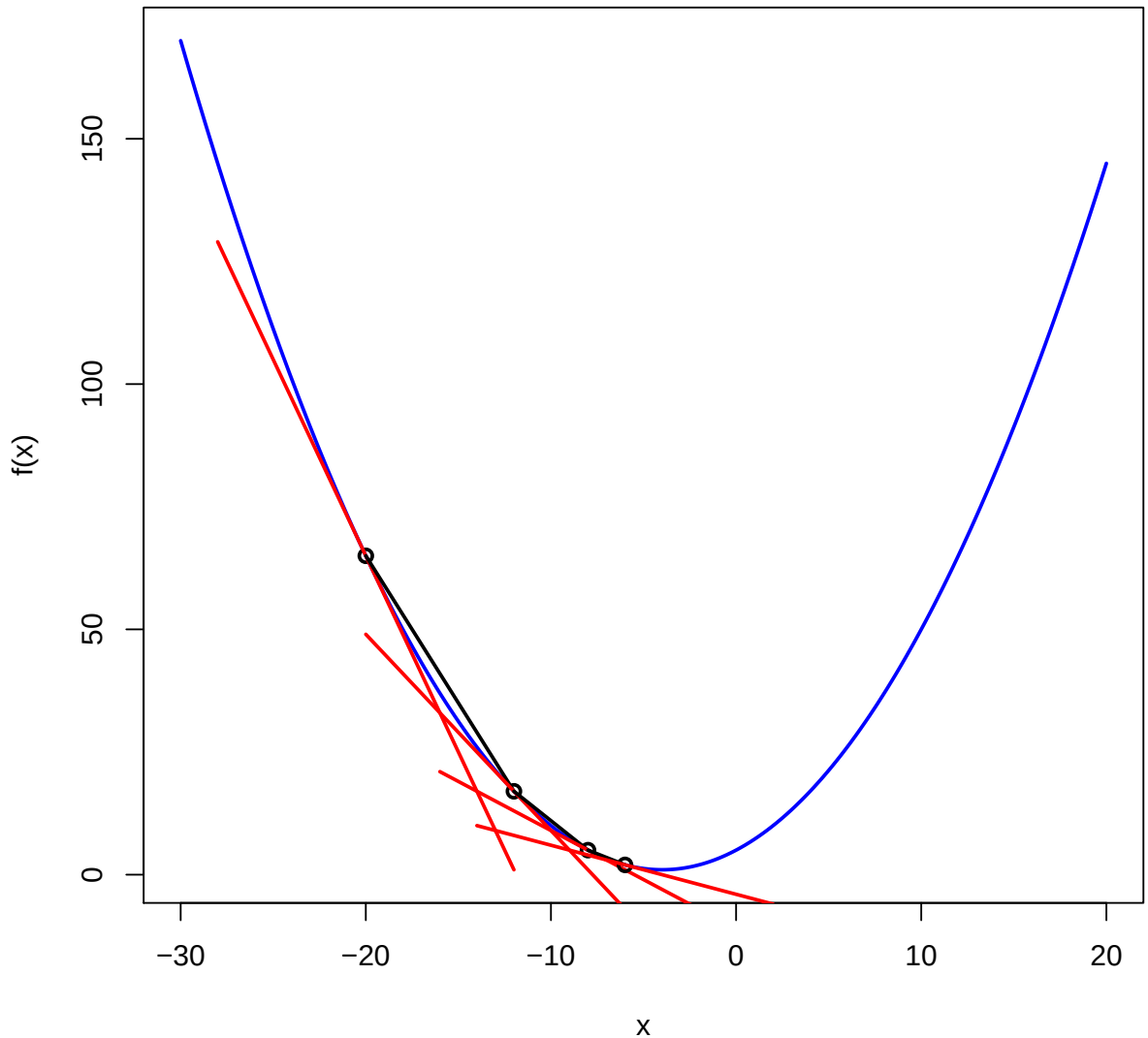
# Logistic regression

- Gradient descent example ($\alpha = 1$):



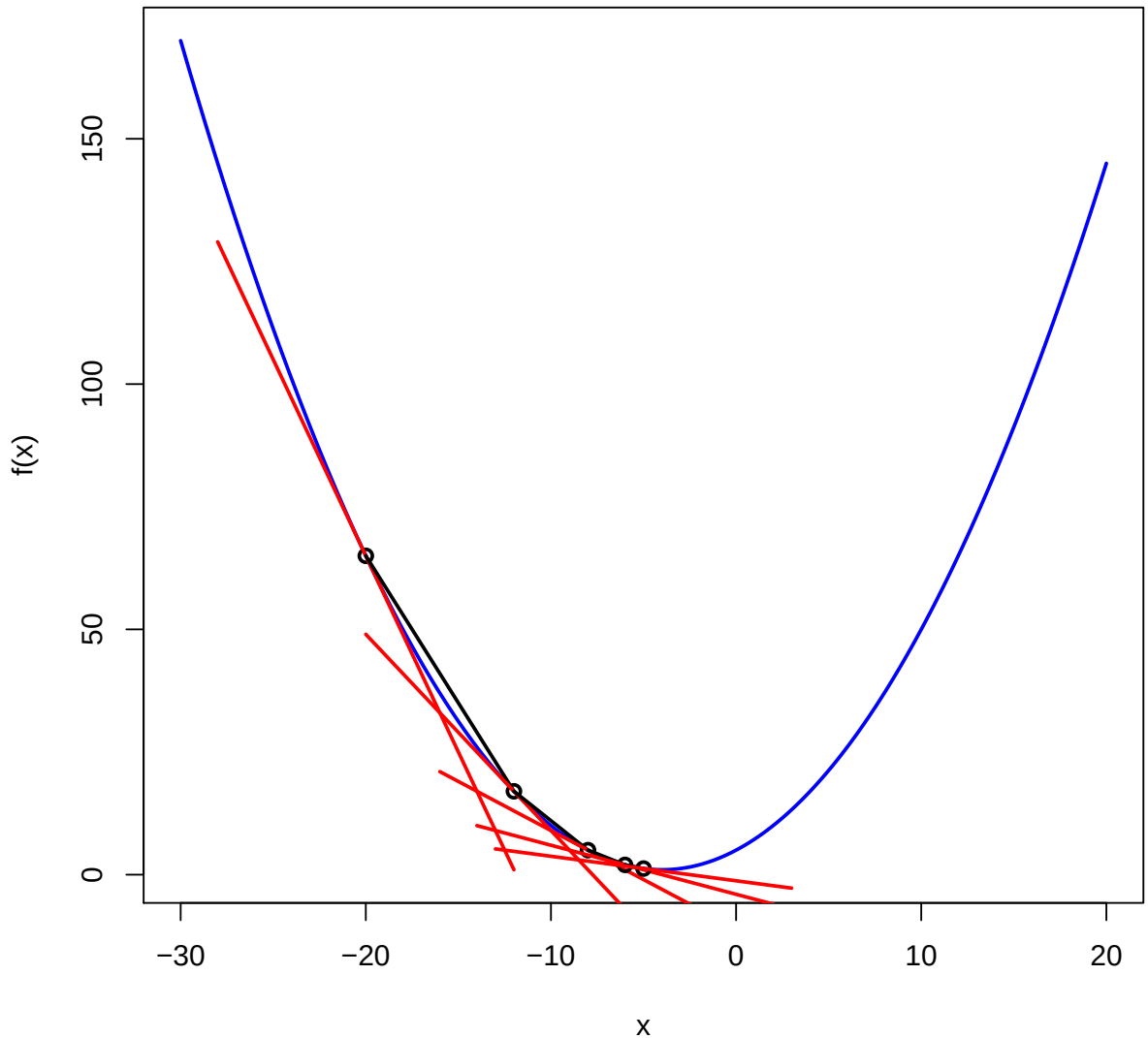
# Logistic regression

- Gradient descent example ($\alpha = 1$):



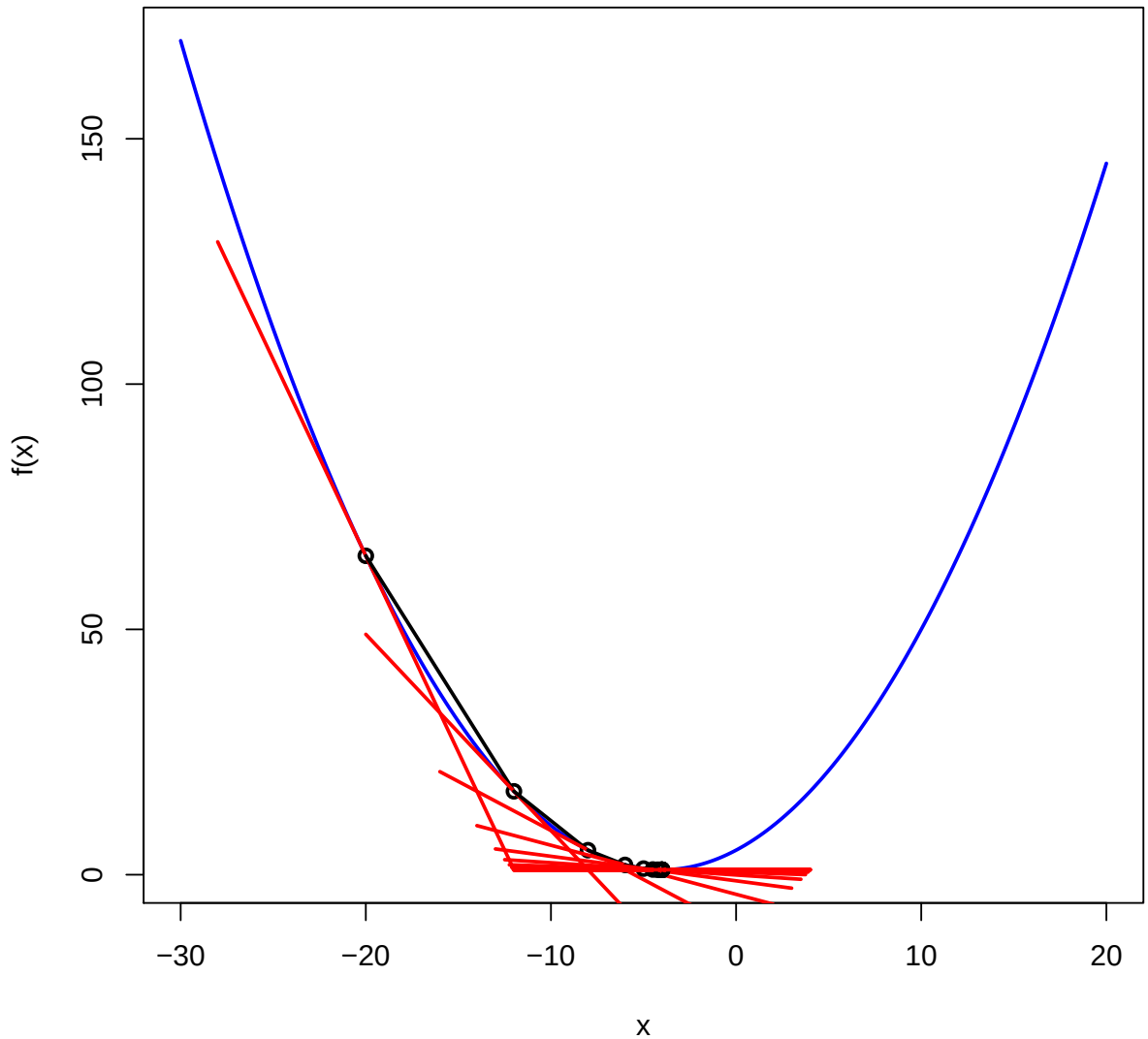
# Logistic regression

- Gradient descent example ($\alpha = 1$):



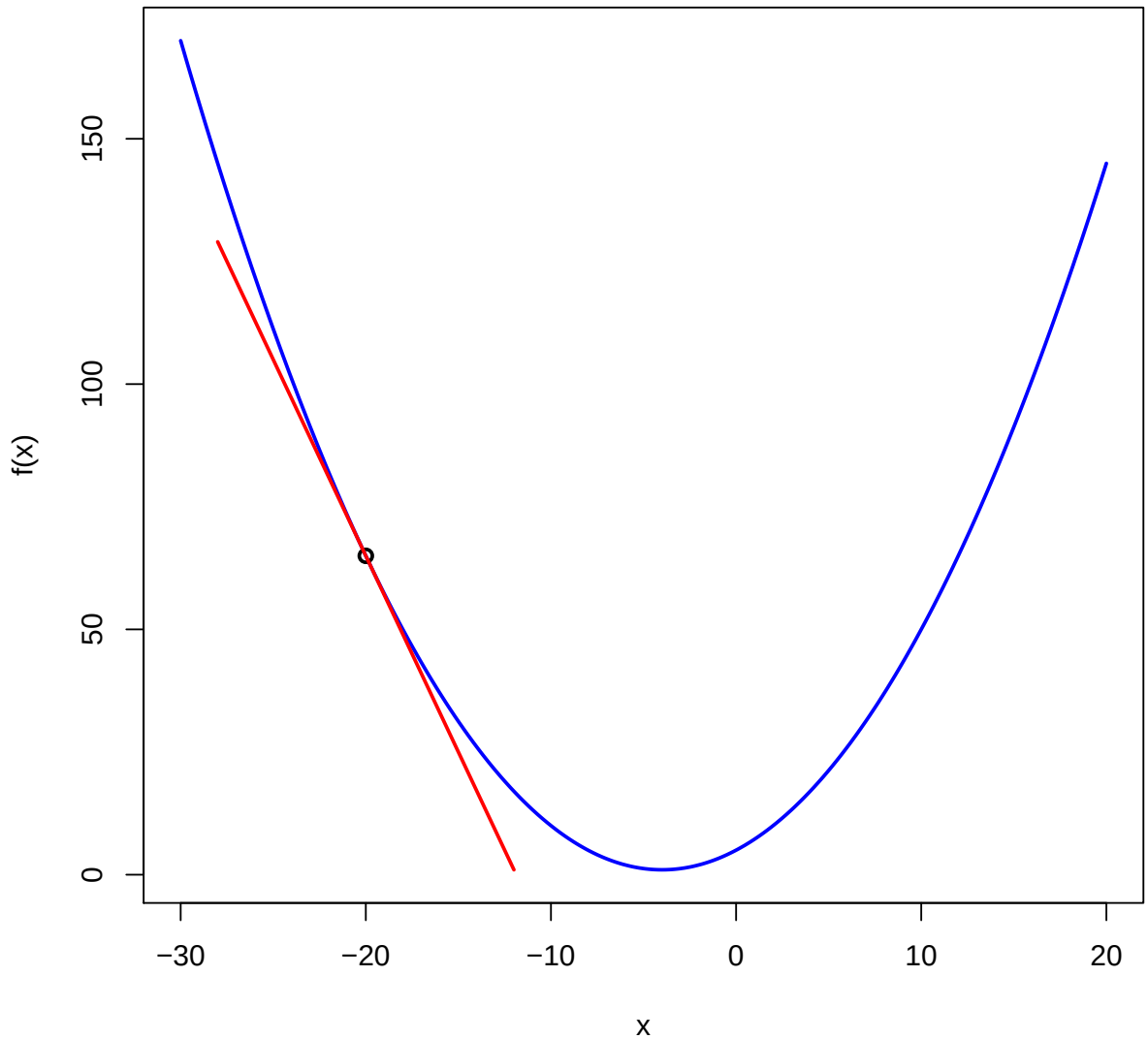
# Logistic regression

- Gradient descent example ($\alpha = 1$):



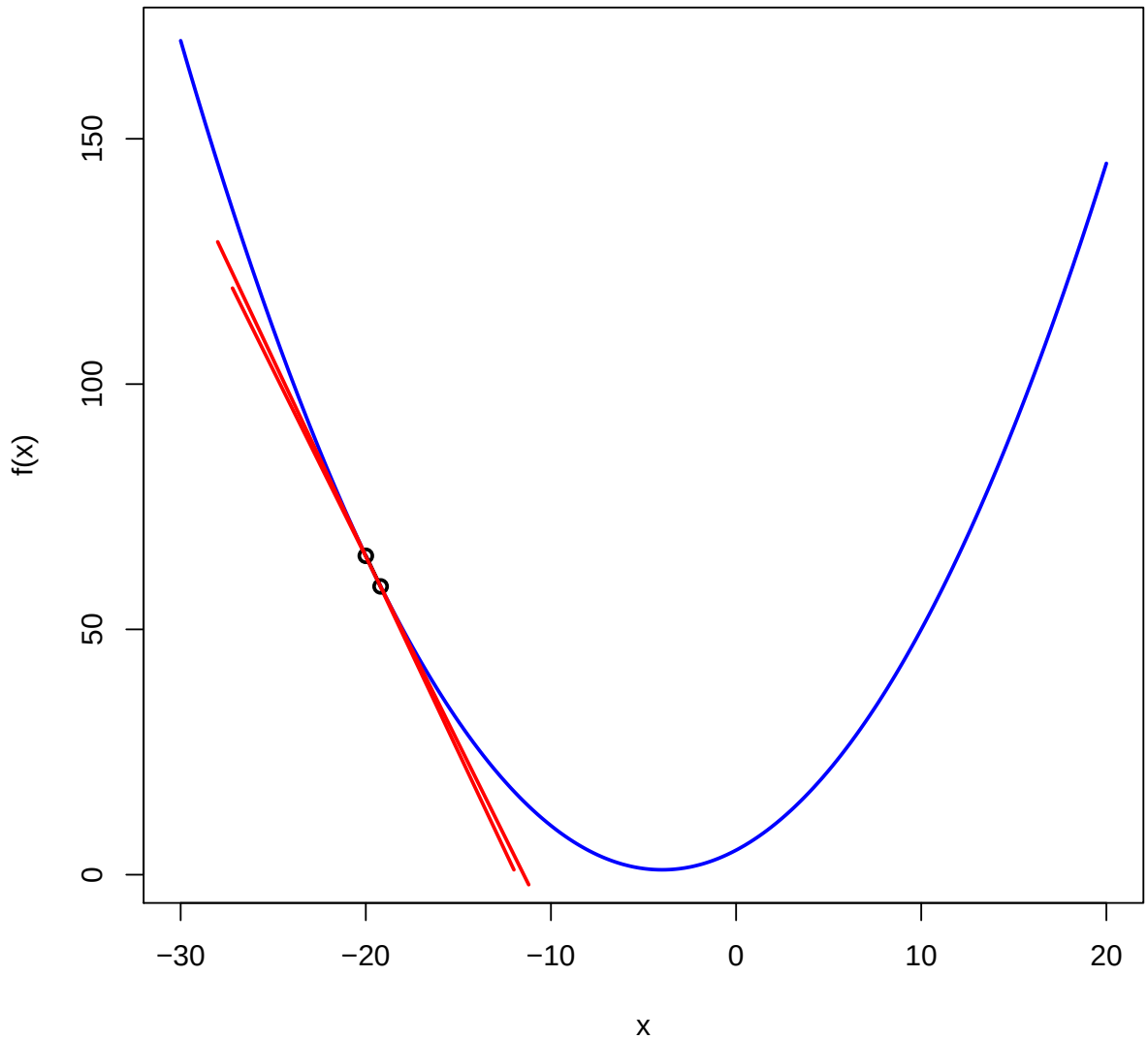
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



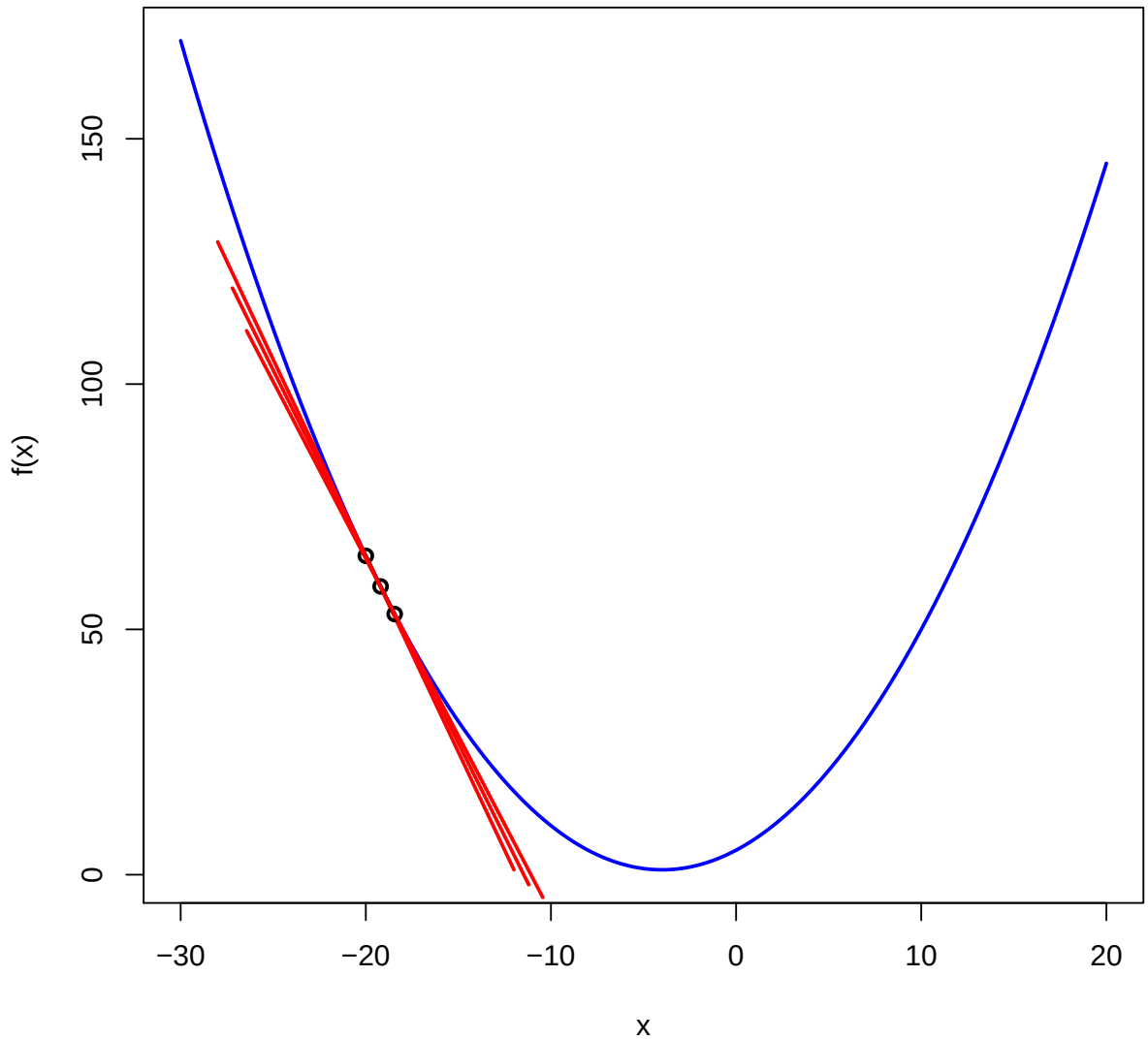
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



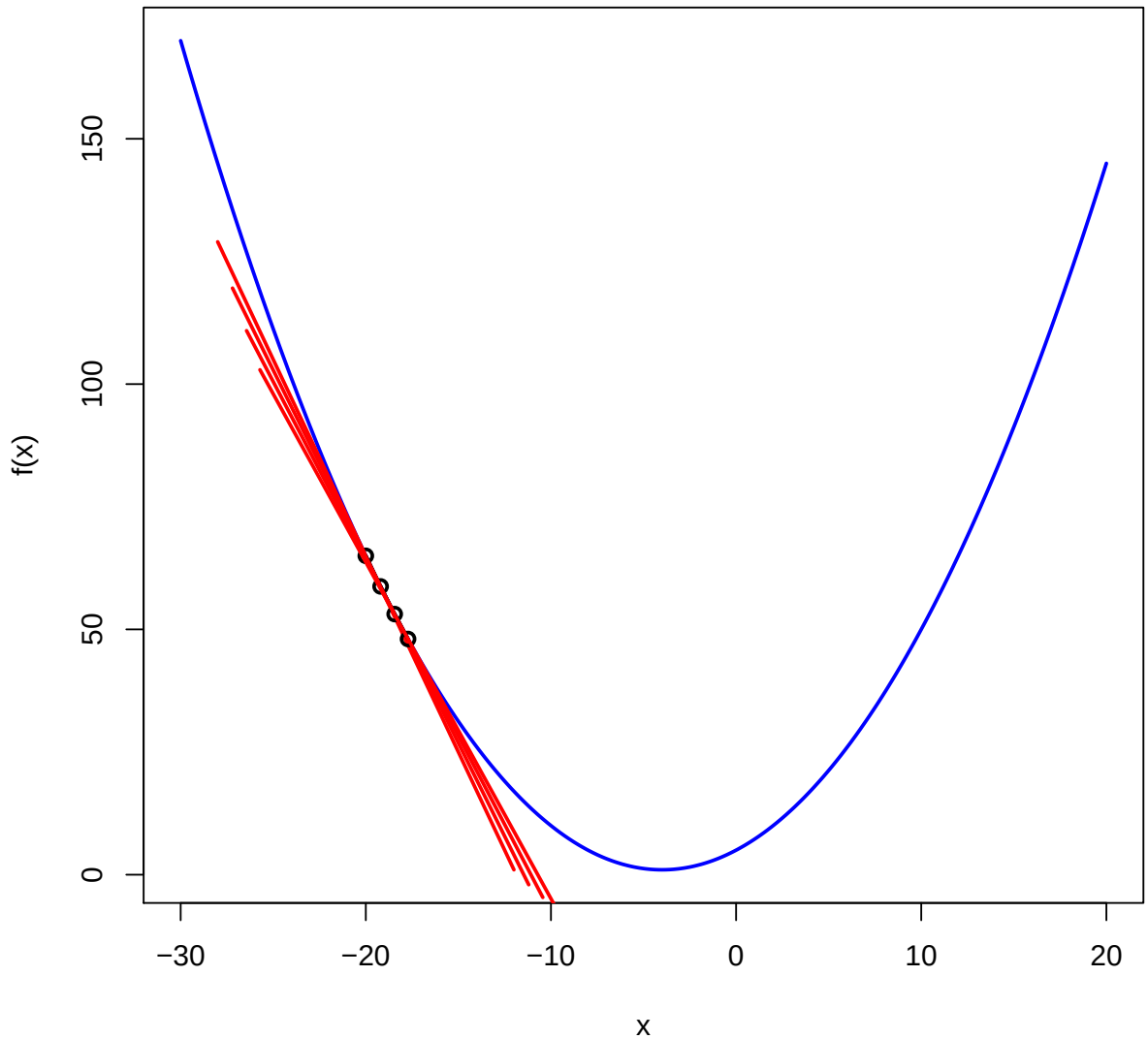
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



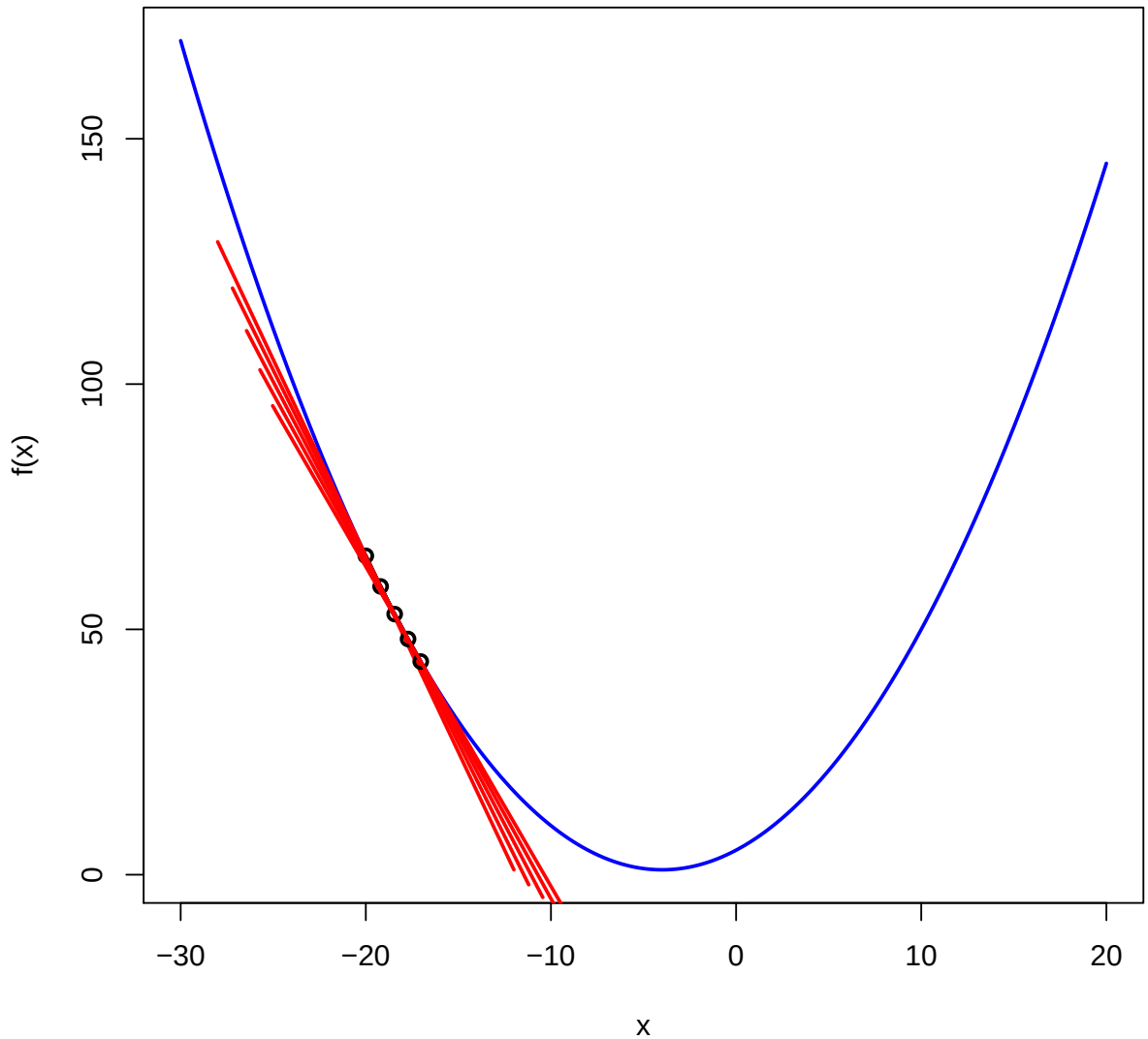
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



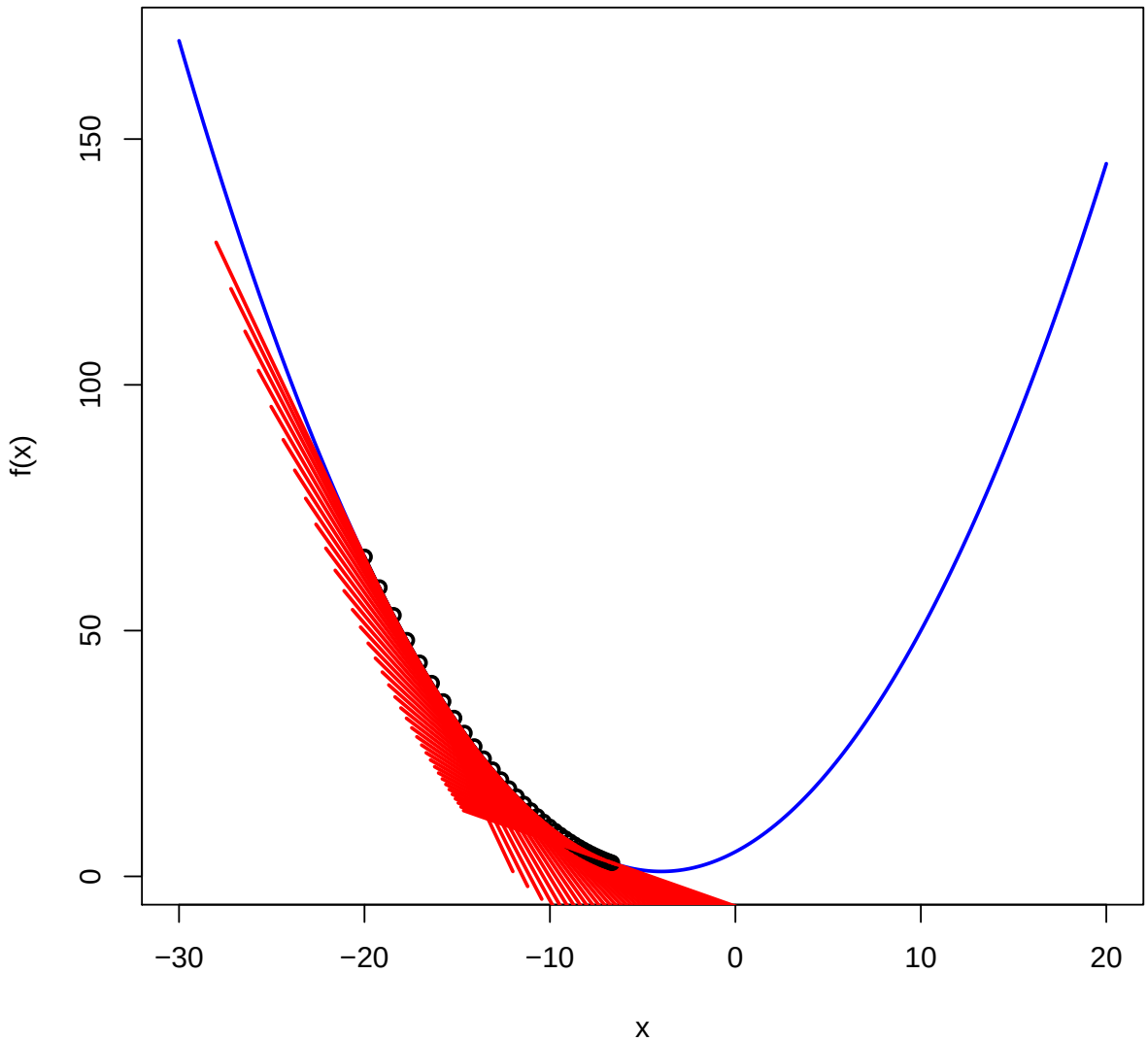
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



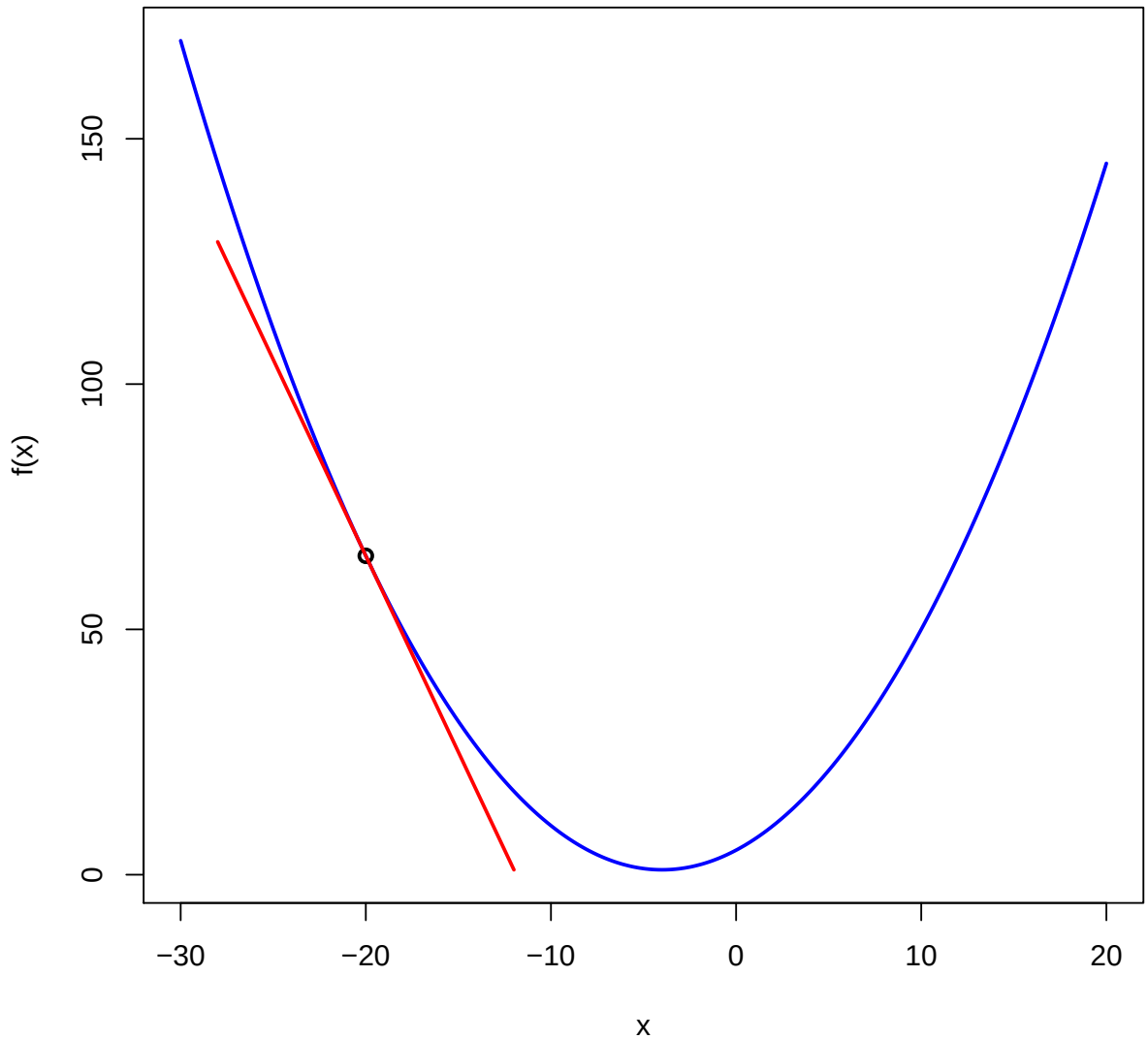
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



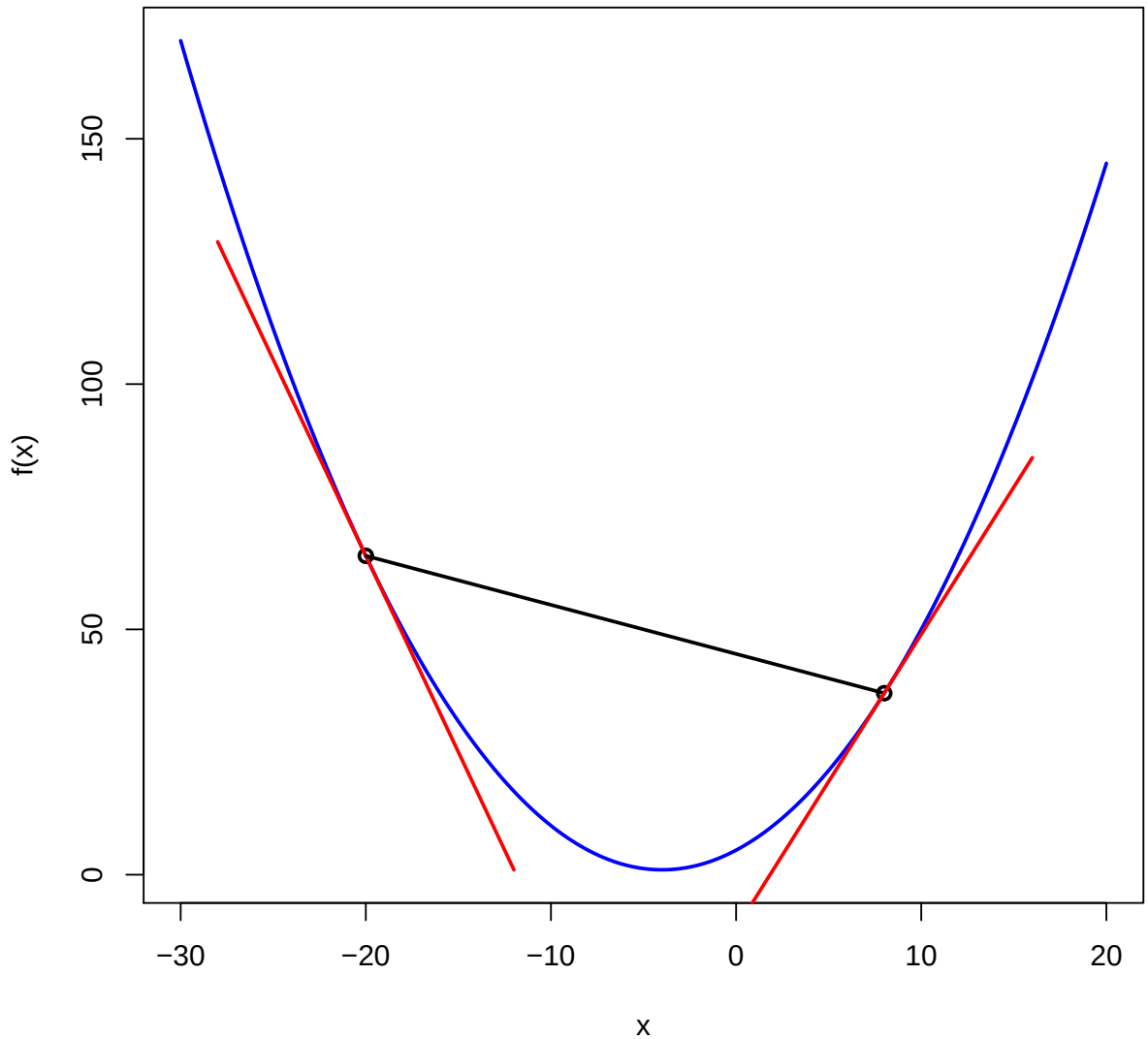
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



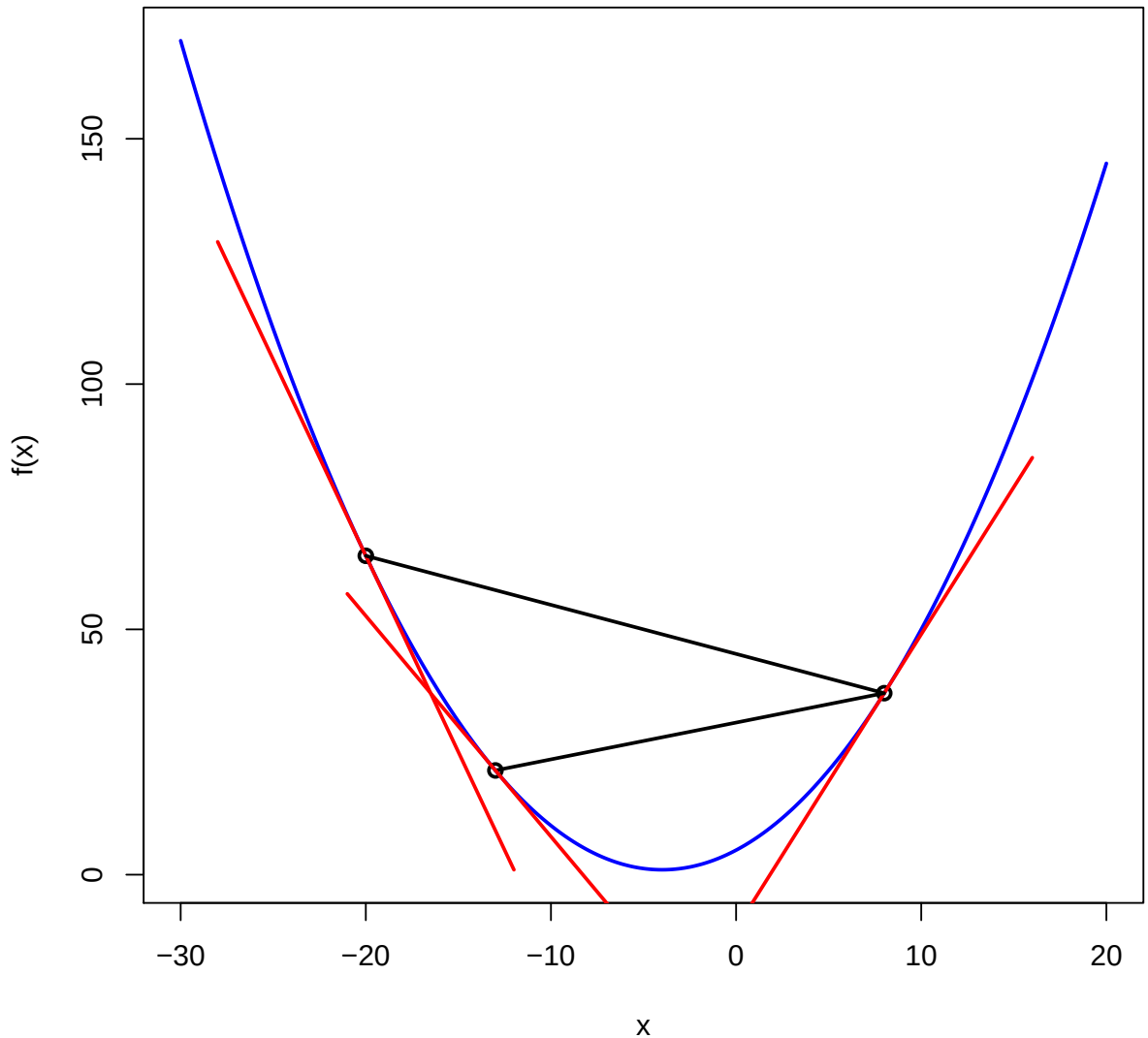
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



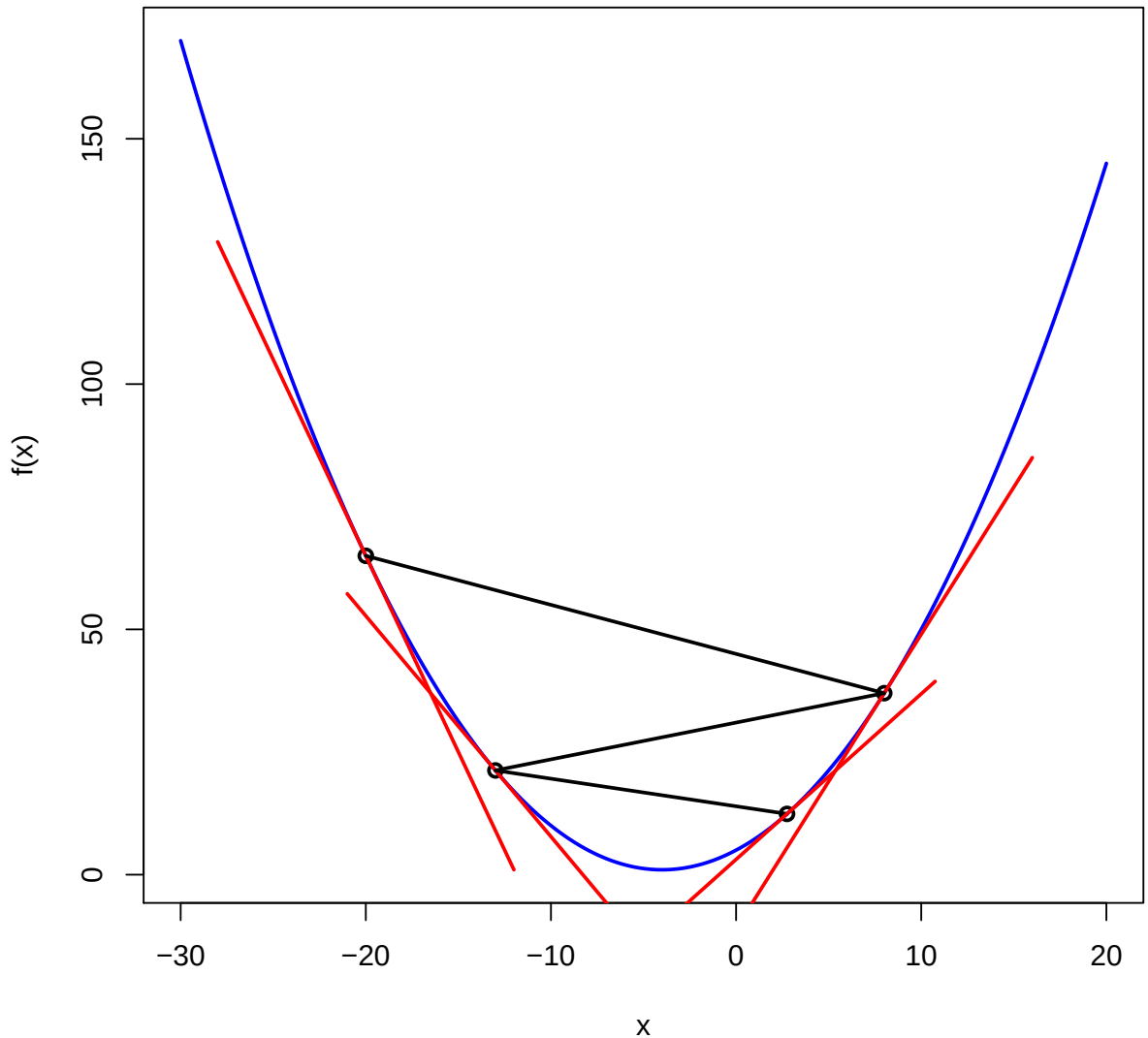
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



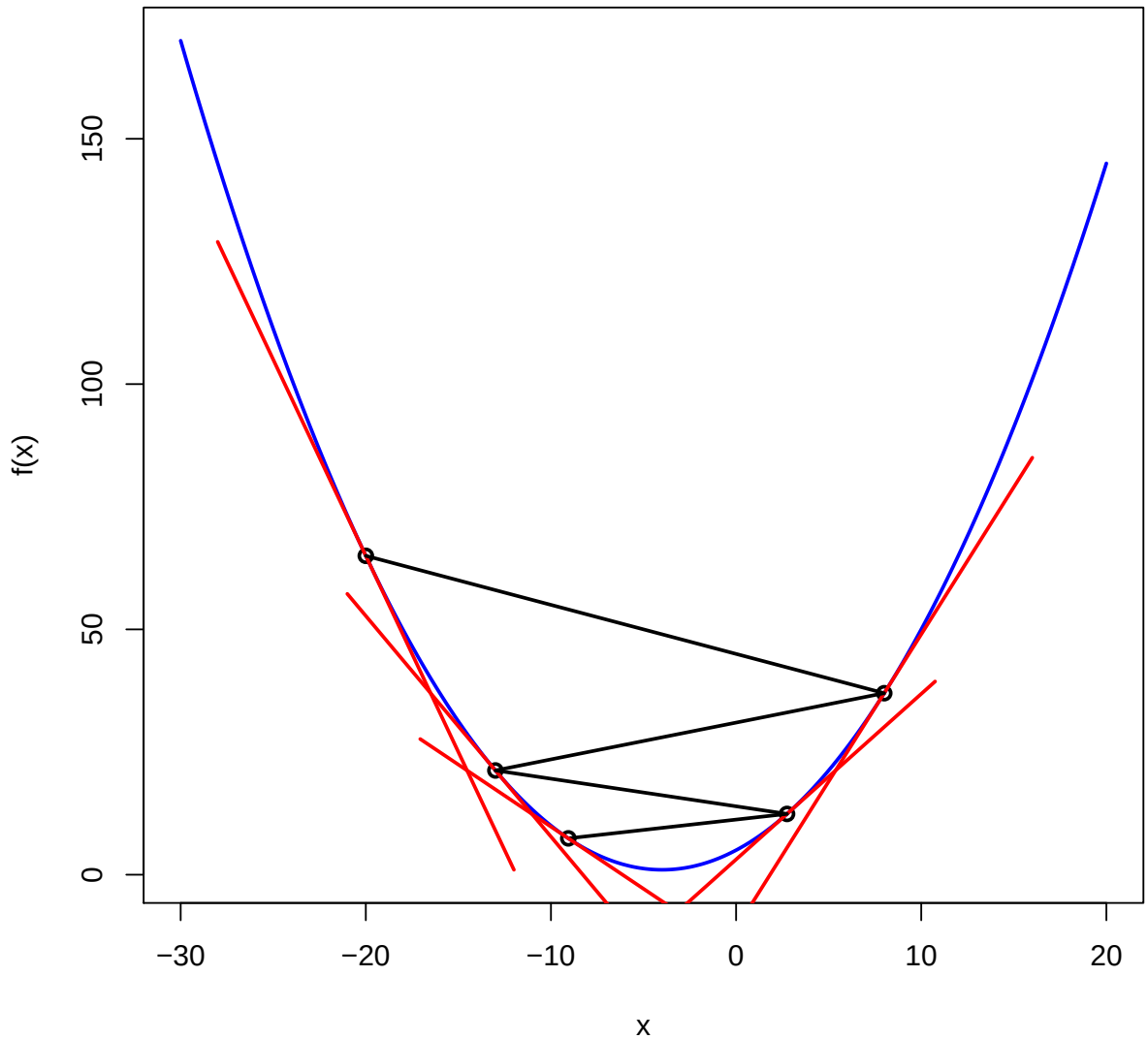
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



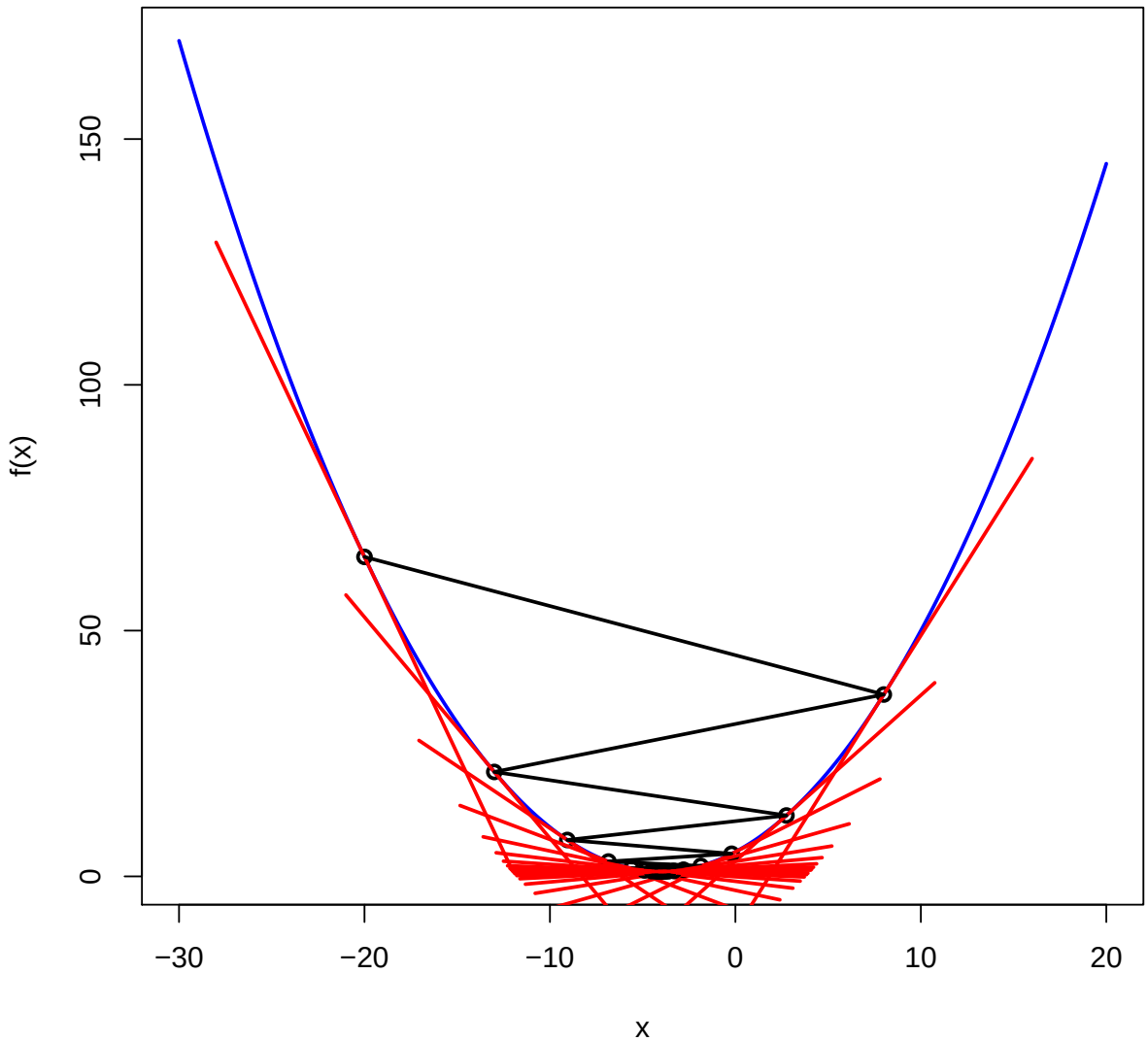
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



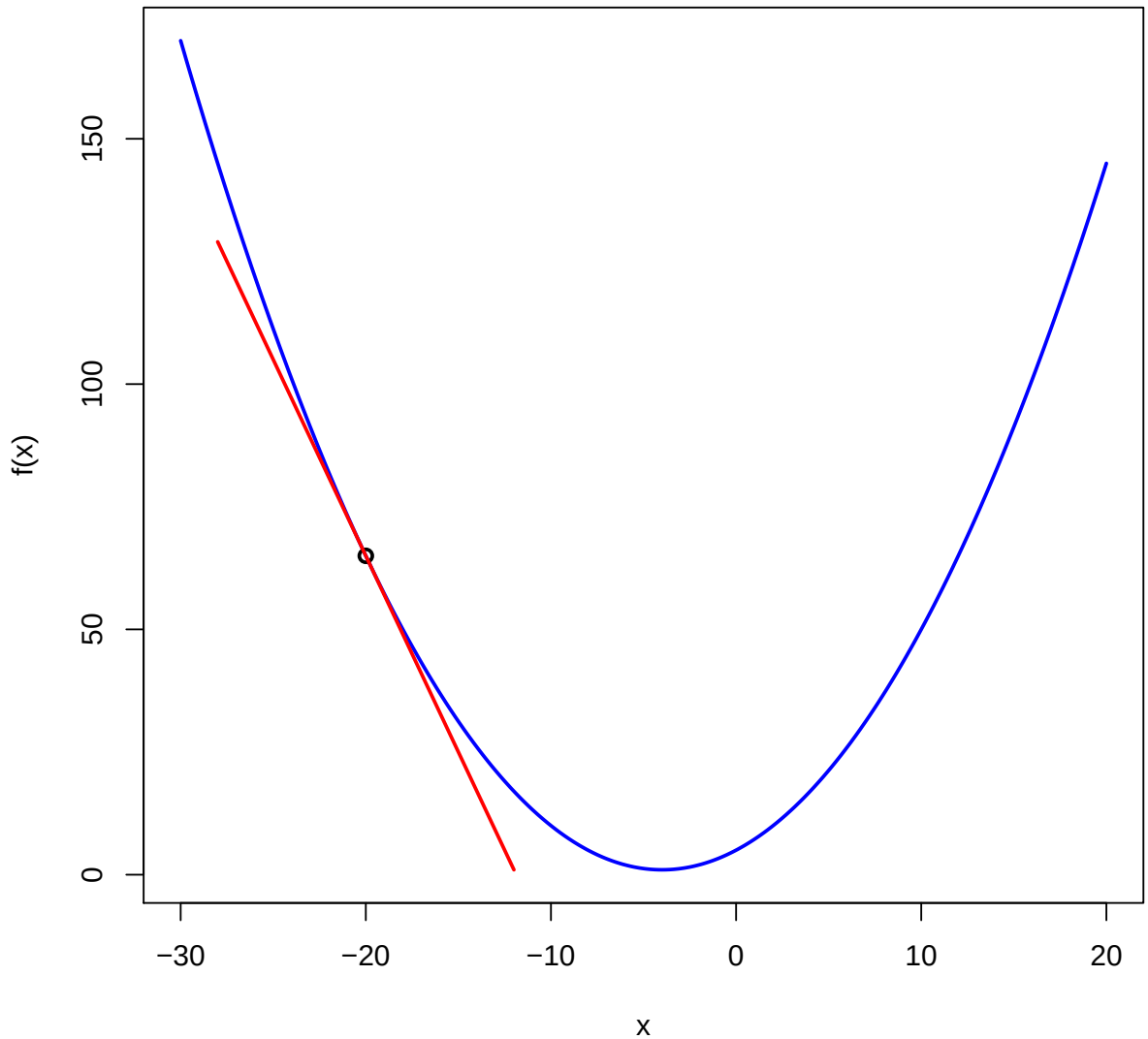
# Logistic regression

- Gradient descent example ($\alpha = 0.1$):



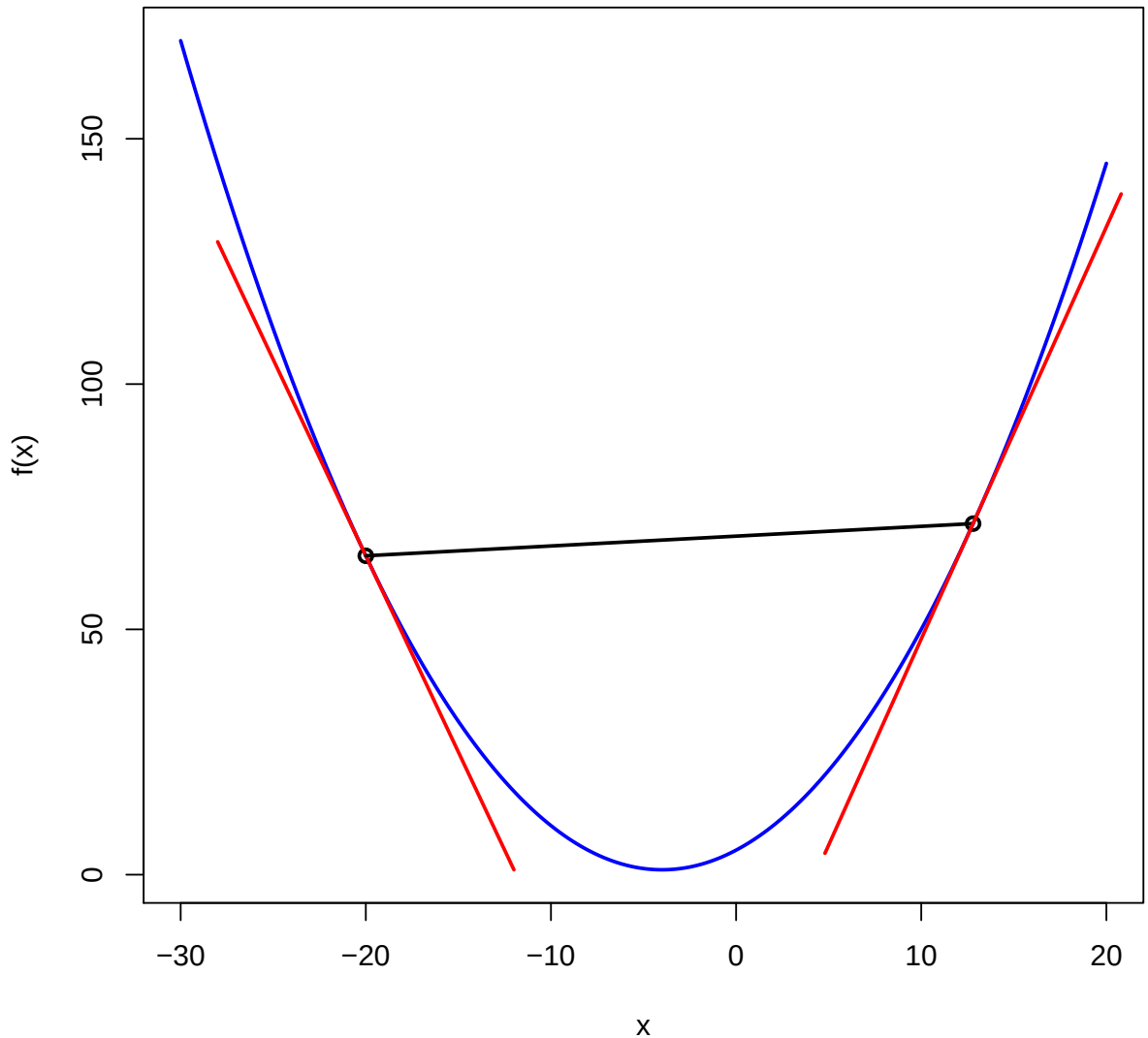
# Logistic regression

- Gradient descent example ($\alpha = 4.1$):



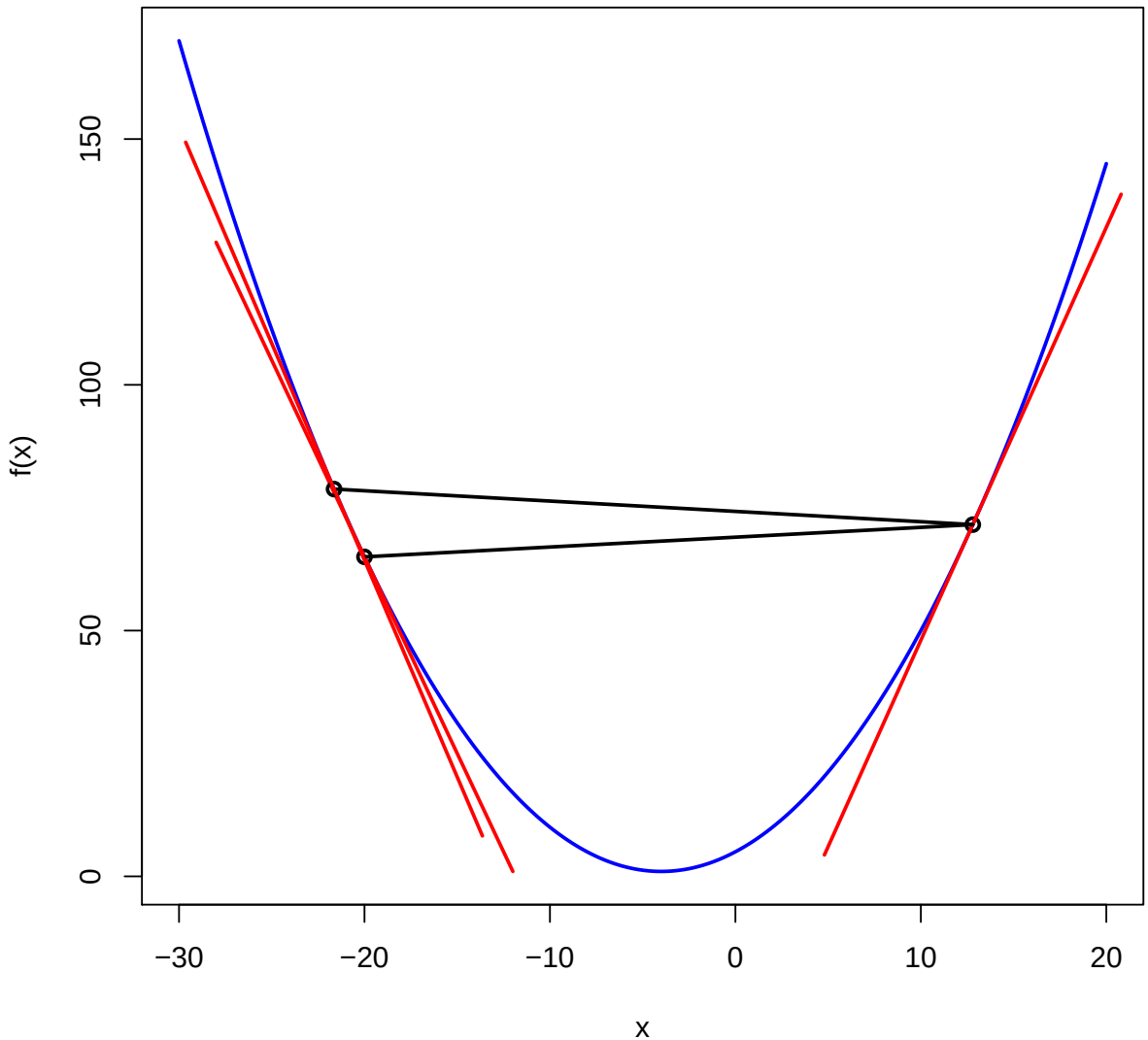
# Logistic regression

- Gradient descent example ($\alpha = 4.1$):



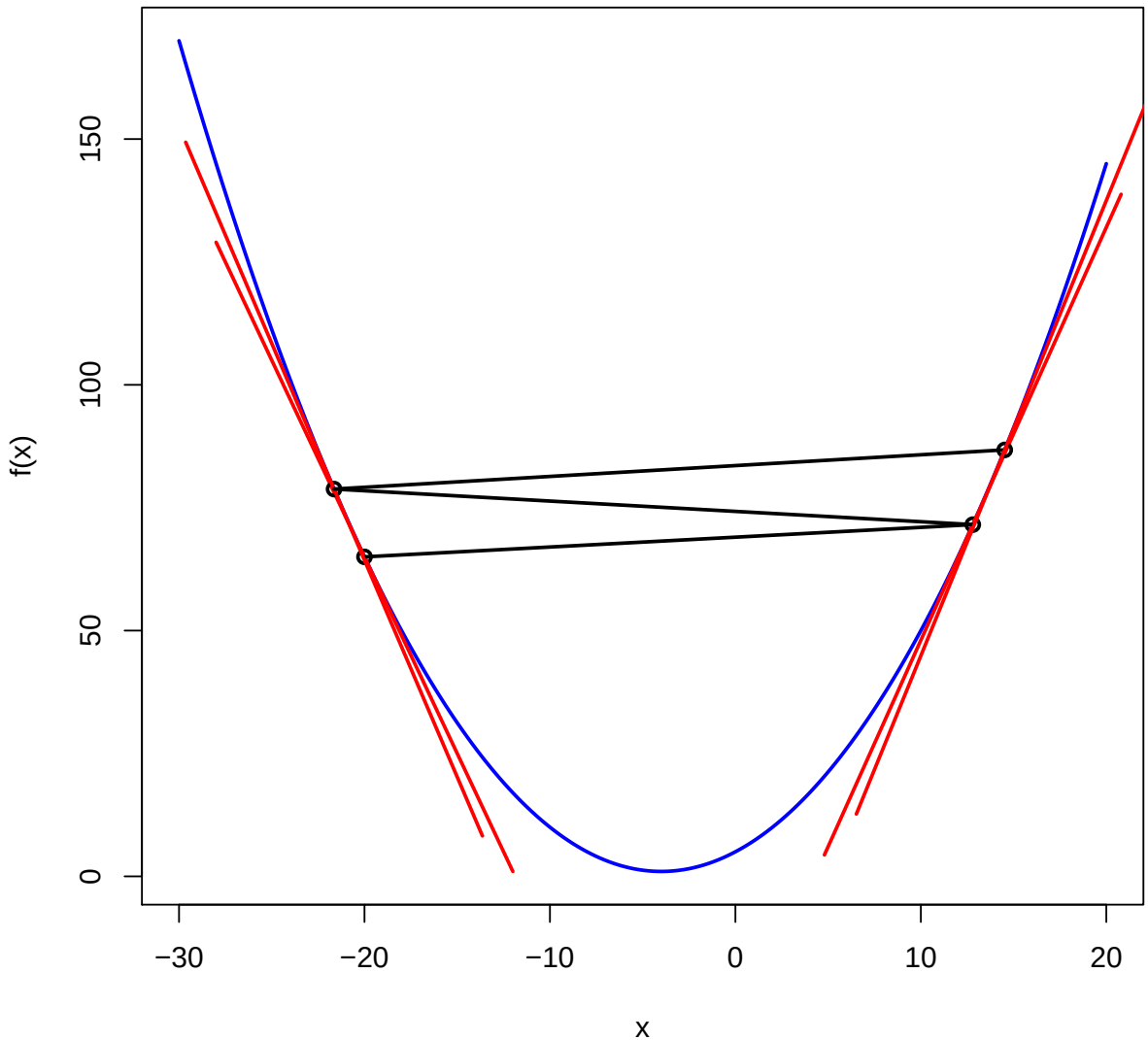
# Logistic regression

- Gradient descent example ($\alpha = 4.1$):



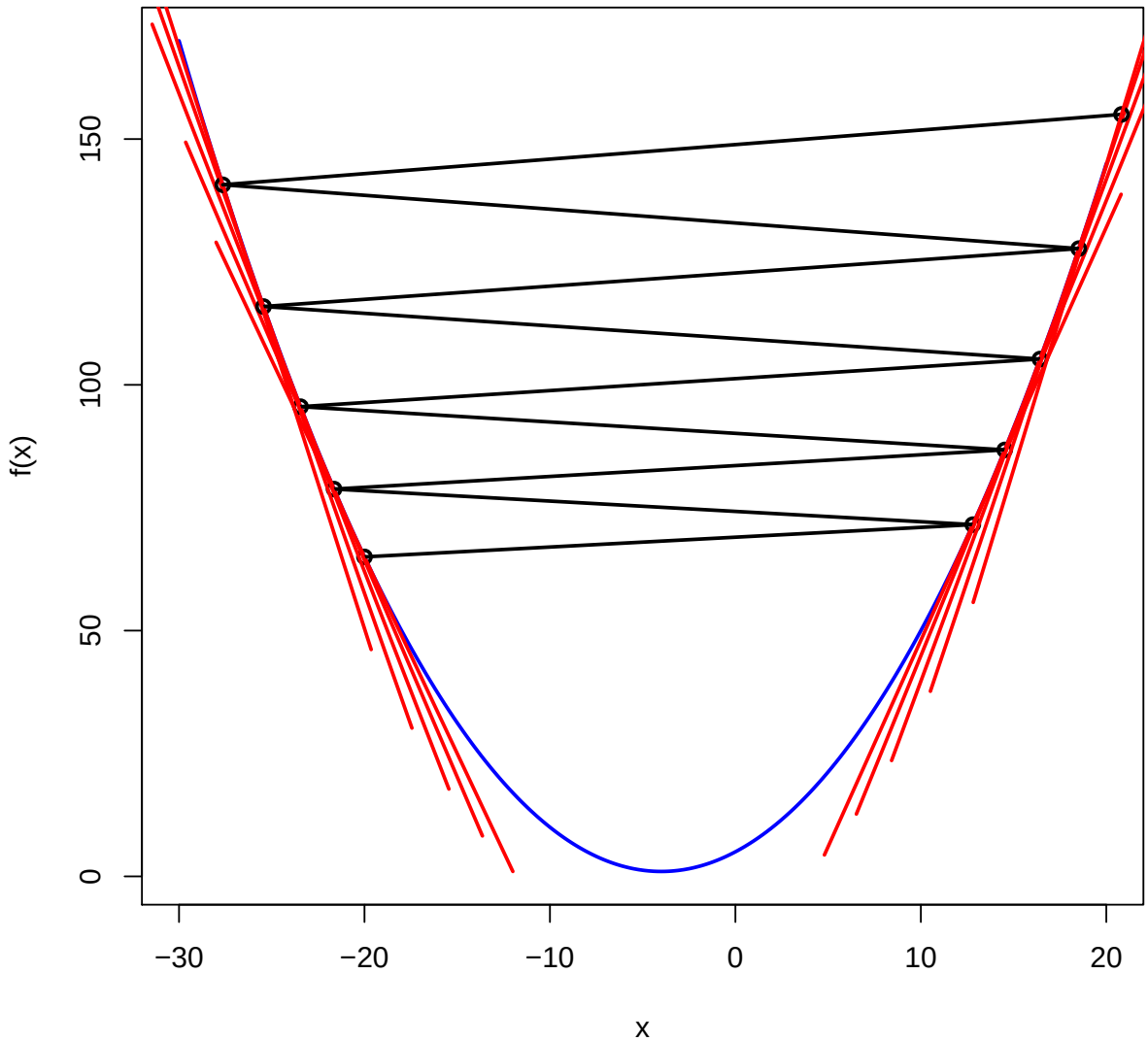
# Logistic regression

- Gradient descent example ($\alpha = 4.1$):



# Logistic regression

- Gradient descent example ($\alpha = 4.1$):



# Stochastic gradient descent

- Computing the gradient can be costly:
  - Sum over $n$ components.
  - The number of training examples $n$ can be large.
- We can approximate the gradient by sampling from the training set.
- **Stochastic gradient descent**:
  - Randomly draw an example from the training set.
  - Compute the gradient based on this single example:

$$\frac{\partial \widehat{L}_i}{\partial w_j} = -\beta_i y_i x_{ij} \,,$$

    where $\widehat{L}_i = \ell_{\log}(y_i, \boldsymbol{w} \cdot \boldsymbol{x}_i)$.
  - Update parameters.
  - Repeat until convergence.

# Stochastic gradient descent

Input: learning rate $\alpha$
$\boldsymbol{w} = \mathbf{0}$; *//(or use random values)*
**while** (approximate minimum is obtained) {
    Randomly shuffle examples in the training set
    **for** i=1 to n {
        $\boldsymbol{w} := \boldsymbol{w} - \alpha \frac{\partial \widehat{L}_i(\boldsymbol{w})}{\partial \boldsymbol{w}}$
    }
}

# Binary classification by boosting

- Iteratively learning base classifier on the data reweighted according to the error of previous base classifiers.

[2] Y. Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997
J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2), 2000

# Binary classification by boosting

- Iteratively learning base classifier on the data reweighted according to the error of previous base classifiers.
- **AdaBoost**[2]

1. $F_0 := 0$.
2. For $k = 1, 2, \ldots, K$:
$$f_k = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} e^{-y_i(F_{k-1}(\boldsymbol{x}_i) + f(\boldsymbol{x}_i))},$$
$$F_k := F_{k-1} + f_k.$$
3. Return $F_K$.

[2] Y. Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997
J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2), 2000

# Binary classification by boosting

- Effectively, we solve by the **coordinate descent method**:

$$\widehat{f} = \underset{f \in \mathrm{lin}(\mathcal{F})}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(\boldsymbol{x}_i)}$$

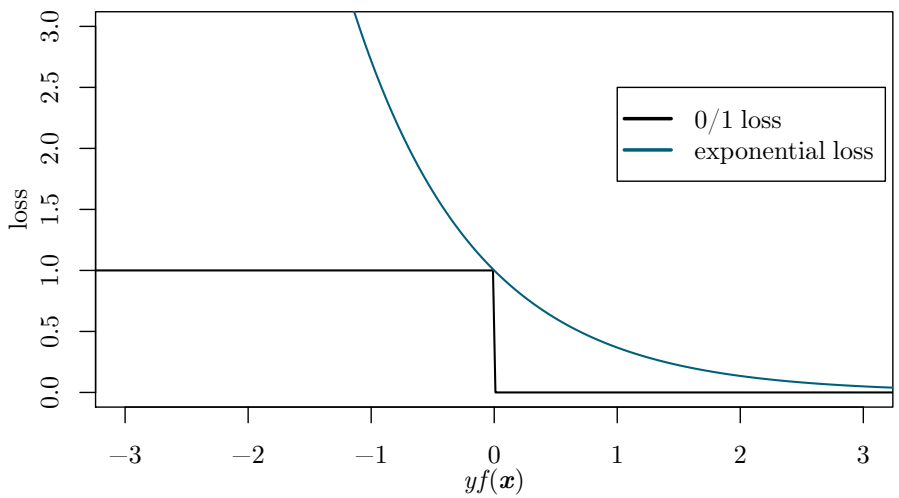
# Binary classification by boosting

- Effectively, we solve by the **coordinate descent method**:

$$\widehat{f} = \underset{f \in \mathrm{lin}(\mathcal{F})}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(\boldsymbol{x}_i)}$$

- We replaced $0/1$ loss with **exponential loss**:

$$\ell_{\exp}(y, f(\boldsymbol{x})) = e^{-yf(\boldsymbol{x})}.$$



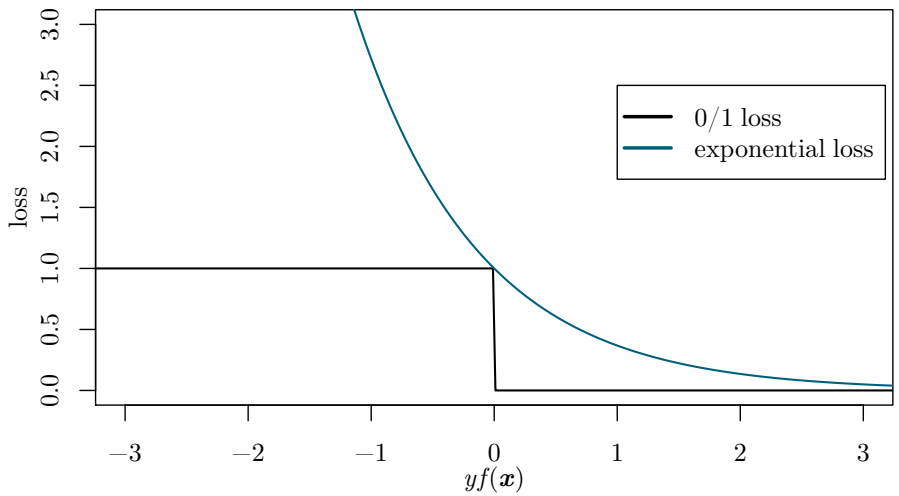
# Binary classification by boosting

- Effectively, we solve by the **coordinate descent method**:

$$\widehat{f} = \underset{f \in \mathrm{lin}(\mathcal{F})}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(\boldsymbol{x}_i)}$$

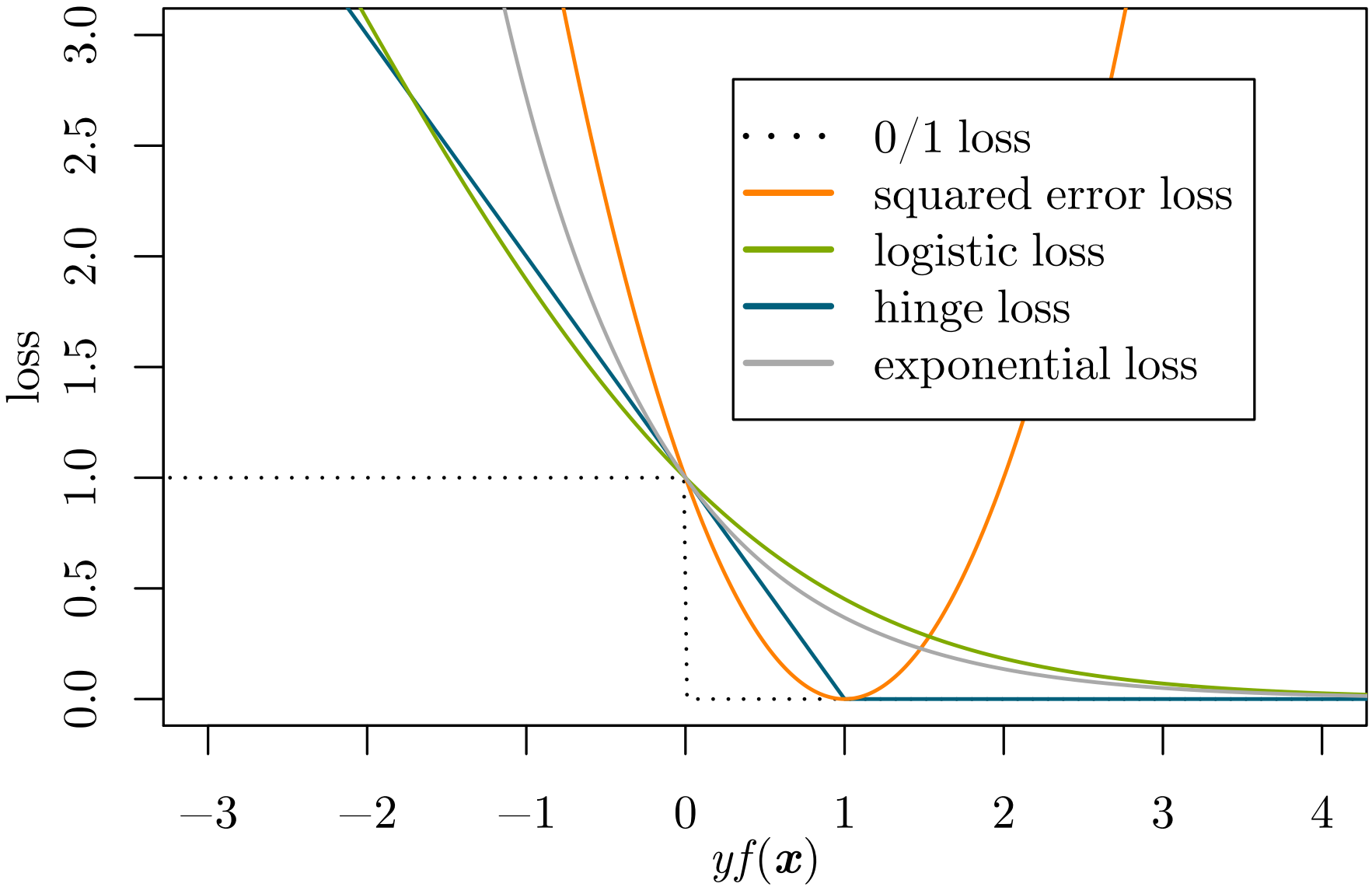
- We replaced $0/1$ loss with **exponential loss**:

$$\ell_{\exp}(y, f(\boldsymbol{x})) = e^{-yf(\boldsymbol{x})}.$$




- Once called "best off-the-shelf method for classification." (Breiman)

# Surrogate convex losses



# Outline

# Statistical learning framework

# Statistical learning framework

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $P(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.

# Statistical learning framework

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $P(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $y \in \mathcal{Y}$ drawn from a distribution $P(y \mid \boldsymbol{x})$.
  - target of our prediction: class label, real value, label vector, etc.,
  - alternative view: **example** $(\boldsymbol{x}, y)$ drawn from $P(\boldsymbol{x}, y)$.

# Statistical learning framework

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $P(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $y \in \mathcal{Y}$ drawn from a distribution $P(y \,|\, \boldsymbol{x})$.
  - target of our prediction: class label, real value, label vector, etc.,
  - alternative view: **example** $(\boldsymbol{x}, y)$ drawn from $P(\boldsymbol{x}, y)$.
- **Prediction** $\hat{y} = h(\boldsymbol{x})$ by means of **prediction function** $h \colon \mathcal{X} \to \mathcal{Y}$.
  - $h$ returns prediction $\hat{y} = h(\boldsymbol{x})$ for every input $\boldsymbol{x}$.

# Statistical learning framework

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $P(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $y \in \mathcal{Y}$ drawn from a distribution $P(y \mid \boldsymbol{x})$.
  - target of our prediction: class label, real value, label vector, etc.,
  - alternative view: **example** $(\boldsymbol{x}, y)$ drawn from $P(\boldsymbol{x}, y)$.
- **Prediction** $\hat{y} = h(\boldsymbol{x})$ by means of **prediction function** $h \colon \mathcal{X} \to \mathcal{Y}$.
  - $h$ returns prediction $\hat{y} = h(\boldsymbol{x})$ for every input $\boldsymbol{x}$.
- **Loss** of our prediction: $\ell(y, \hat{y})$.
  - $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a problem-specific **loss function**.

# Statistical learning framework

- **Input** $\boldsymbol{x} \in \mathcal{X}$ drawn from a distribution $P(\boldsymbol{x})$.
  - usually a feature vector, $\mathcal{X} \subseteq \mathbb{R}^d$.
- **Outcome** $y \in \mathcal{Y}$ drawn from a distribution $P(y \,|\, \boldsymbol{x})$.
  - target of our prediction: class label, real value, label vector, etc.,
  - alternative view: **example** $(\boldsymbol{x}, y)$ drawn from $P(\boldsymbol{x}, y)$.
- **Prediction** $\hat{y} = h(\boldsymbol{x})$ by means of **prediction function** $h \colon \mathcal{X} \to \mathcal{Y}$.
  - $h$ returns prediction $\hat{y} = h(\boldsymbol{x})$ for every input $\boldsymbol{x}$.
- **Loss** of our prediction: $\ell(y, \hat{y})$.
  - $\ell \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a problem-specific **loss function**.
- **Goal**: find a prediction function with small loss.

# Risk

- **Goal**: minimize the **expected** loss over all examples (**risk**):

$$L_\ell(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\ell(y, h(\boldsymbol{x}))\right].$$

# Risk

- **Goal**: minimize the **expected** loss over all examples (**risk**):

$$L_\ell(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\ell(y, h(\boldsymbol{x}))\right].$$

- The **optimal** prediction function over all possible functions:

$$h^* = \arg\min_h L_\ell(h),$$

(so called **Bayes prediction function**).

# Risk

- **Goal**: minimize the **expected** loss over all examples (**risk**):

$$L_\ell(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P}\left[\ell(y, h(\boldsymbol{x}))\right].$$

- The **optimal** prediction function over all possible functions:

$$h^* = \arg\min_h L_\ell(h),$$

  (so called **Bayes prediction function**).

- The smallest achievable risk (**Bayes risk**):

$$L_\ell^* = L_\ell(h^*).$$

# Decomposition of risk

$$L_\ell(h) \quad = \quad \mathbb{E}_{(\boldsymbol{x},y)}\left[\ell(y, h(\boldsymbol{x}))\right]$$

# Decomposition of risk

$$
\begin{aligned}
L_\ell(h) &= \mathbb{E}_{(\boldsymbol{x},y)}\left[\ell(y,h(\boldsymbol{x}))\right] \\
&= \int_{\mathcal{X}\times\mathcal{Y}} \ell(y,h(\boldsymbol{x}))P(\boldsymbol{x},y)\mathrm{d}\boldsymbol{x}\mathrm{d}y
\end{aligned}
$$

# Decomposition of risk

$$
\begin{aligned}
L_\ell(h) &= \mathbb{E}_{(\boldsymbol{x},y)}\left[\ell(y,h(\boldsymbol{x}))\right] \\
&= \int_{\mathcal{X}\times\mathcal{Y}} \ell(y,h(\boldsymbol{x}))P(\boldsymbol{x},y)\mathrm{d}\boldsymbol{x}\mathrm{d}y \\
&= \int_{\mathcal{X}} \left( \int_{\mathcal{Y}} \ell(y,h(\boldsymbol{x}))P(y\,|\,\boldsymbol{x})\mathrm{d}y \right) P(\boldsymbol{x})\mathrm{d}\boldsymbol{x}
\end{aligned}
$$

# Decomposition of risk

$$
\begin{aligned}
L_\ell(h) &= \mathbb{E}_{(\boldsymbol{x},y)}\left[\ell(y,h(\boldsymbol{x}))\right] \\
&= \int_{\mathcal{X}\times\mathcal{Y}} \ell(y,h(\boldsymbol{x}))P(\boldsymbol{x},y)\mathrm{d}\boldsymbol{x}\mathrm{d}y \\
&= \int_{\mathcal{X}} \left( \int_{\mathcal{Y}} \ell(y,h(\boldsymbol{x}))P(y\,|\,\boldsymbol{x})\mathrm{d}y \right) P(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{\,\boldsymbol{x}}\left[L_\ell(h\,|\,\boldsymbol{x})\right].
\end{aligned}
$$

# Decomposition of risk

$$
\begin{aligned}
L_\ell(h) &= \mathbb{E}_{(\boldsymbol{x},y)}\left[\ell(y,h(\boldsymbol{x}))\right] \\
&= \int_{\mathcal{X}\times\mathcal{Y}} \ell(y,h(\boldsymbol{x}))P(\boldsymbol{x},y)\mathrm{d}\boldsymbol{x}\mathrm{d}y \\
&= \int_{\mathcal{X}} \left( \int_{\mathcal{Y}} \ell(y,h(\boldsymbol{x}))P(y\,|\,\boldsymbol{x})\mathrm{d}y \right) P(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{\,\boldsymbol{x}}\left[L_\ell(h\,|\,\boldsymbol{x})\right].
\end{aligned}
$$

- $L_\ell(h\,|\,\boldsymbol{x})$ is the **conditional risk** of $\hat{y} = h(\boldsymbol{x})$ at $\boldsymbol{x}$.

# Decomposition of risk

$$
\begin{aligned}
L_\ell(h) &= \mathbb{E}_{(\boldsymbol{x},y)}\left[\ell(y, h(\boldsymbol{x}))\right] \\
&= \int_{\mathcal{X}\times\mathcal{Y}} \ell(y, h(\boldsymbol{x}))P(\boldsymbol{x}, y)\mathrm{d}\boldsymbol{x}\mathrm{d}y \\
&= \int_{\mathcal{X}} \left( \int_{\mathcal{Y}} \ell(y, h(\boldsymbol{x}))P(y \,|\, \boldsymbol{x})\mathrm{d}y \right) P(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\
&= \mathbb{E}_{\,\boldsymbol{x}}\left[L_\ell(h \,|\, \boldsymbol{x})\right].
\end{aligned}
$$

- $L_\ell(h \,|\, \boldsymbol{x})$ is the **conditional risk** of $\hat{y} = h(\boldsymbol{x})$ at $\boldsymbol{x}$.
- Bayes prediction **minimizes the conditional risk** for every $\boldsymbol{x}$:

$$h^*(\boldsymbol{x}) = \arg\min_h L_\ell(h \,|\, \boldsymbol{x}).$$

## Example — binary classification and $0/1$ loss

- The outcome $y$ is **binary**, $y \in \{-1, 1\}$.

## Example — binary classification and $0/1$ loss

- The outcome $y$ is **binary**, $y \in \{-1, 1\}$.
- **0/1** loss function:

$$\ell_{0/1}(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y}\,, \\ 1, & \text{otherwise}\,. \end{cases}$$

## Example — binary classification and $0/1$ loss

- The outcome $y$ is **binary**, $y \in \{-1, 1\}$.
- **0/1** loss function:

$$\ell_{0/1}(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y}\,, \\ 1, & \text{otherwise}\,. \end{cases}$$

- Define $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$. The conditional 0/1 risk at $\boldsymbol{x}$ is:

$$L_{0/1}(h|\boldsymbol{x}) = \eta(\boldsymbol{x})[\![h(\boldsymbol{x}) = -1]\!] + (1 - \eta(\boldsymbol{x}))[\![h(\boldsymbol{x}) = 1]\!].$$

## Example — binary classification and $0/1$ loss

- The outcome $y$ is **binary**, $y \in \{-1, 1\}$.
- **0/1** loss function:

$$\ell_{0/1}(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y}\,, \\ 1, & \text{otherwise}\,. \end{cases}$$

- Define $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$. The conditional 0/1 risk at $\boldsymbol{x}$ is:

$$L_{0/1}(h|\boldsymbol{x}) = \eta(\boldsymbol{x})[\![h(\boldsymbol{x}) = -1]\!] + (1 - \eta(\boldsymbol{x}))[\![h(\boldsymbol{x}) = 1]\!].$$

- The **Bayes classifier**:

$$h^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \eta(\boldsymbol{x}) > 1 - \eta(\boldsymbol{x}) \\ -1 & \text{if } \eta(\boldsymbol{x}) < 1 - \eta(\boldsymbol{x}) \end{cases} = \operatorname{sgn}\left(\eta(\boldsymbol{x}) - 1/2\right),$$

  and the **Bayes conditional risk**:

$$L_\ell(h^* \,|\, \boldsymbol{x}) = \min\{\eta(\boldsymbol{x}), 1 - \eta(\boldsymbol{x})\}.$$

# Regret

- Bayes risk measures the **difficulty** of the problems.
- Difficult data $\Longrightarrow$ high Bayes risk $\Longrightarrow$ every $h$ has high risk.

# Regret

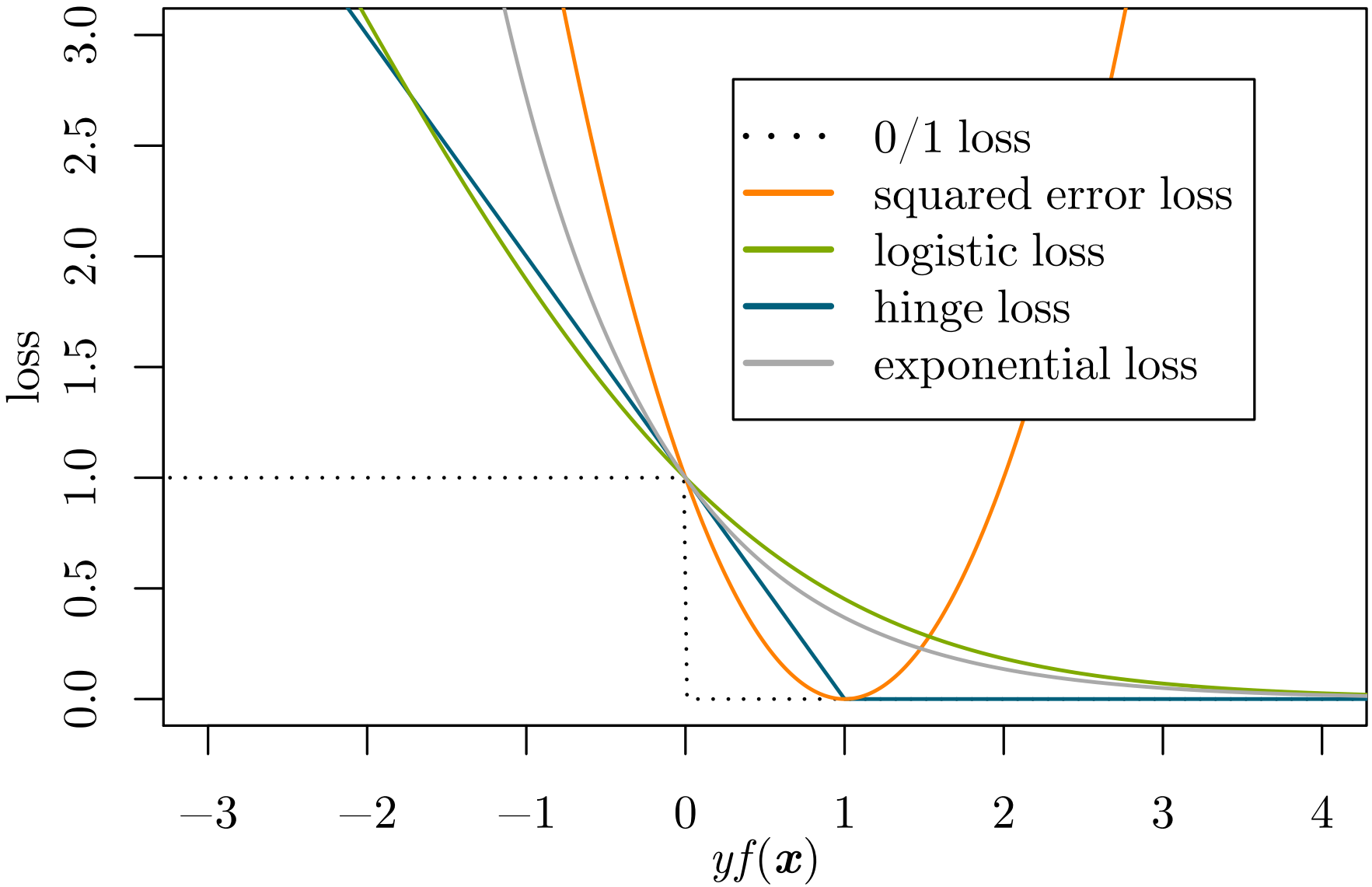
- Bayes risk measures the **difficulty** of the problems.
- Difficult data $\Longrightarrow$ high Bayes risk $\Longrightarrow$ every $h$ has high risk.
- It is not the risk what captures the prediction accuracy of $h$, but rather **how close the risk of $h$ is to the risk of $h^*$**.

# Regret

- Bayes risk measures the **difficulty** of the problems.
- Difficult data $\Longrightarrow$ high Bayes risk $\Longrightarrow$ every $h$ has high risk.
- It is not the risk what captures the prediction accuracy of $h$, but rather **how close the risk of $h$ is to the risk of $h^*$**.
- The **regret** (**excess risk**) of a prediction function $h$:

$$\mathrm{Reg}_\ell(h) = L_\ell(h) - L_\ell^*$$

# Surrogate convex losses

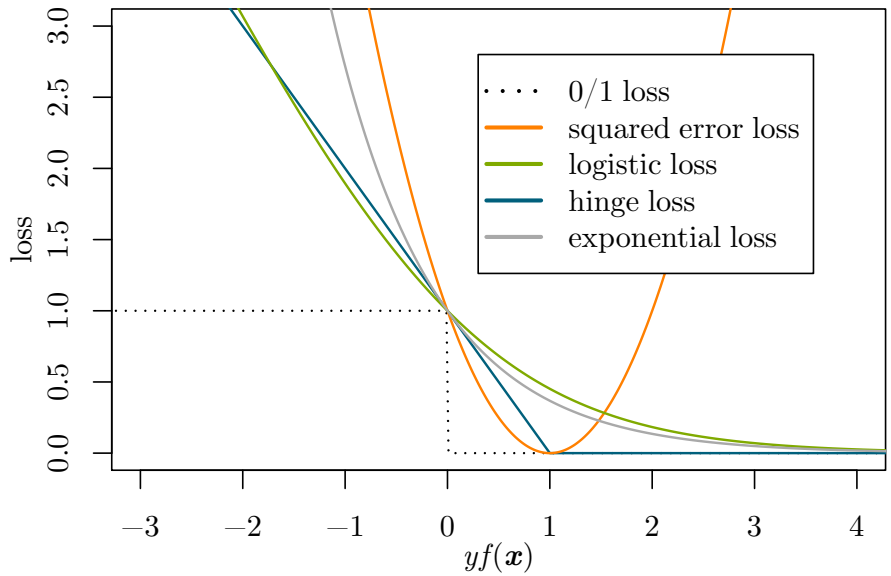


# Surrogate convex losses

In each case, $\ell(y, f(\boldsymbol{x}))$ is a function of a **single** variable:

$$\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x})).$$

We call $yf(\boldsymbol{x})$ the **margin**, and such $\ell$ – **margin-based loss function**.

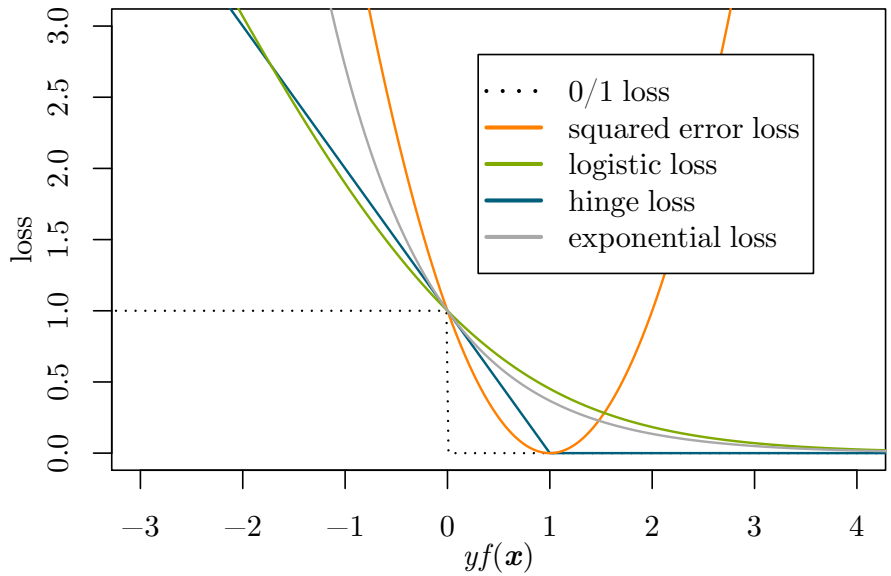


# Surrogate convex losses

In each case, $\ell(y, f(\boldsymbol{x}))$ is a function of a **single** variable:

$$\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x})).$$

We call $yf(\boldsymbol{x})$ the **margin**, and such $\ell$ – **margin-based loss function**.



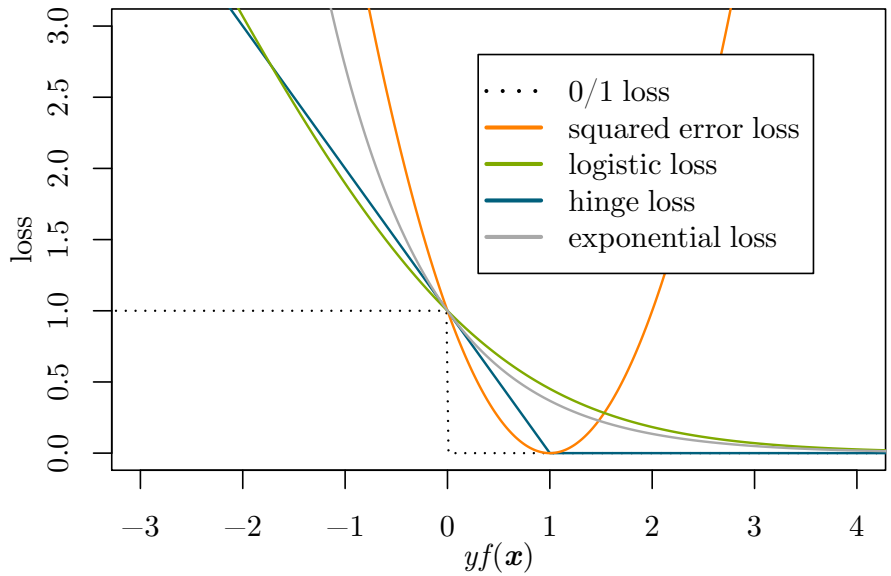

- All losses are **convex** and can be scaled to **upperbound** $0/1$ loss.

# Surrogate convex losses

In each case, $\ell(y, f(\boldsymbol{x}))$ is a function of a **single** variable:

$$\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x})).$$

We call $yf(\boldsymbol{x})$ the **margin**, and such $\ell$ – **margin-based loss function**.



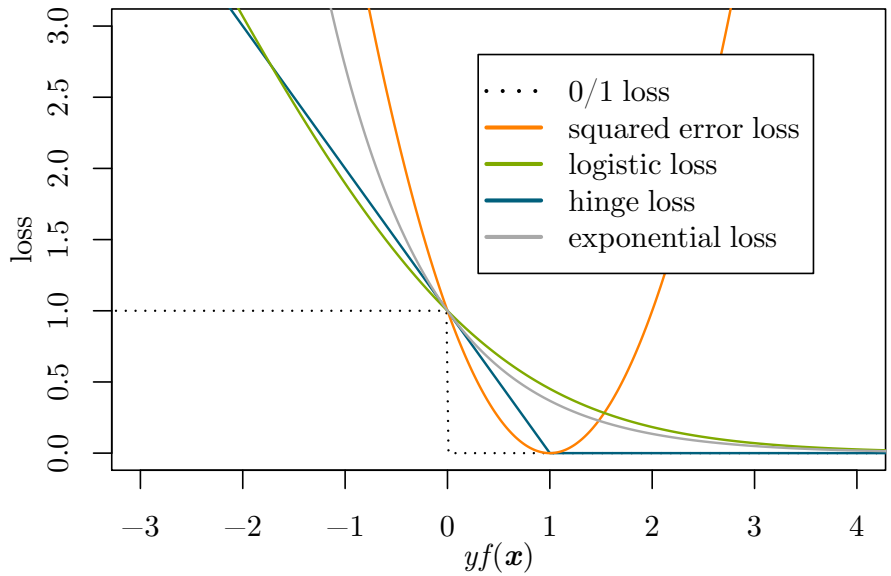

- All losses are **convex** and can be scaled to **upperbound** $0/1$ loss.
- Minimizing such an upper bound gives a bound on $0/1$ **risk**, but **not** on the $0/1$ **regret**!

# Surrogate convex losses

In each case, $\ell(y, f(\boldsymbol{x}))$ is a function of a **single** variable:

$$\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x})).$$

We call $yf(\boldsymbol{x})$ the **margin**, and such $\ell$ – **margin-based loss function**.



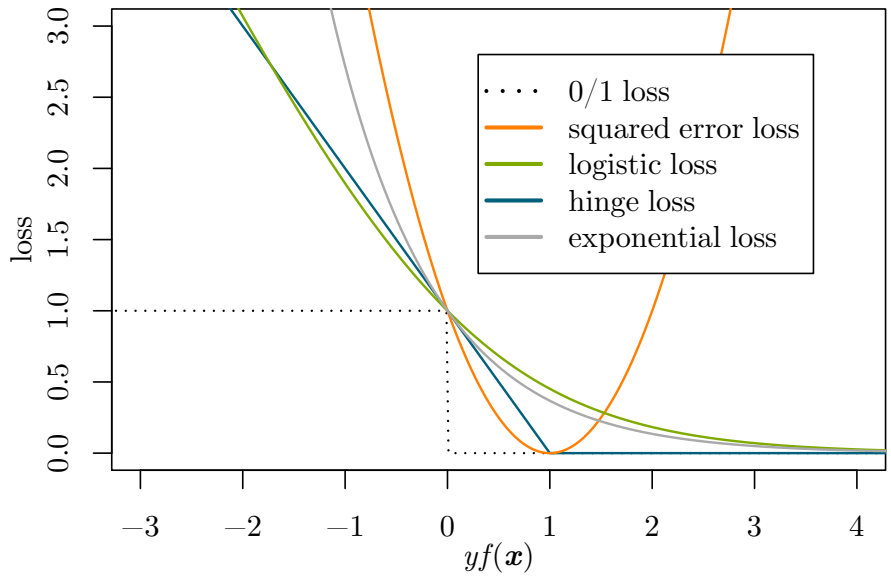

- All losses are **convex** and can be scaled to **upperbound** $0/1$ loss.
- Minimizing such an upper bound gives a bound on $0/1$ **risk**, but **not** on the $0/1$ **regret**!
- Do we actually solve the original classification problem by minimizing the surrogate loss?

# Surrogate convex losses

In each case, $\ell(y, f(\boldsymbol{x}))$ is a function of a **single** variable:

$$\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x})).$$

We call $yf(\boldsymbol{x})$ the **margin**, and such $\ell$ – **margin-based loss function**.




- All losses are **convex** and can be scaled to **upperbound** $0/1$ loss.
- Minimizing such an upper bound gives a bound on $0/1$ **risk**, but **not** on the $0/1$ **regret**!
- Do we actually solve the original classification problem by minimizing the surrogate loss?
- Need a **theory** for classification with **surrogate convex losses**.

# Outline

# Calibration

- **Task loss** $\ell_{\mathrm{T}}$, the one to be actually minimized.
  - e.g., $0/1$ loss in binary classification.

# Calibration

- **Task loss** $\ell_{\mathrm{T}}$, the one to be actually minimized.
  - e.g., $0/1$ loss in binary classification.
- **Surrogate (proxy) loss** $\ell_{\mathrm{S}}$, the one which we use in learning.
  - e.g., logistic loss in binary classification.

# Calibration

- **Task loss** $\ell_{\mathrm{T}}$, the one to be actually minimized.
  - e.g., $0/1$ loss in binary classification.
- **Surrogate (proxy) loss** $\ell_{\mathrm{S}}$, the one which we use in learning.
  - e.g., logistic loss in binary classification.
- We say a surrogate loss $\ell_{\mathrm{S}}$ is **calibrated (consistent)** for a task loss $\ell_{\mathrm{T}}$ if for any sequence of functions $\{f_n\}$, and any distribution $P(\boldsymbol{x}, y)$:

$$L_{\ell_{\mathrm{S}}}(f_n) \to \min_f L_{\ell_{\mathrm{S}}}(f) \Longrightarrow L_{\ell_{\mathrm{T}}}(f_n) \to \min_f L_{\ell_{\mathrm{T}}}(f).$$

  or equivalently:

$$\mathrm{Reg}_{\ell_{\mathrm{S}}}(f_n) \to 0 \Longrightarrow \mathrm{Reg}_{\ell_{\mathrm{T}}}(f_n) \to 0.$$

# Calibration

- **Task loss** $\ell_{\mathrm{T}}$, the one to be actually minimized.
  - e.g., $0/1$ loss in binary classification.
- **Surrogate (proxy) loss** $\ell_{\mathrm{S}}$, the one which we use in learning.
  - e.g., logistic loss in binary classification.
- We say a surrogate loss $\ell_{\mathrm{S}}$ is **calibrated (consistent)** for a task loss $\ell_{\mathrm{T}}$ if for any sequence of functions $\{f_n\}$, and any distribution $P(\boldsymbol{x}, y)$:

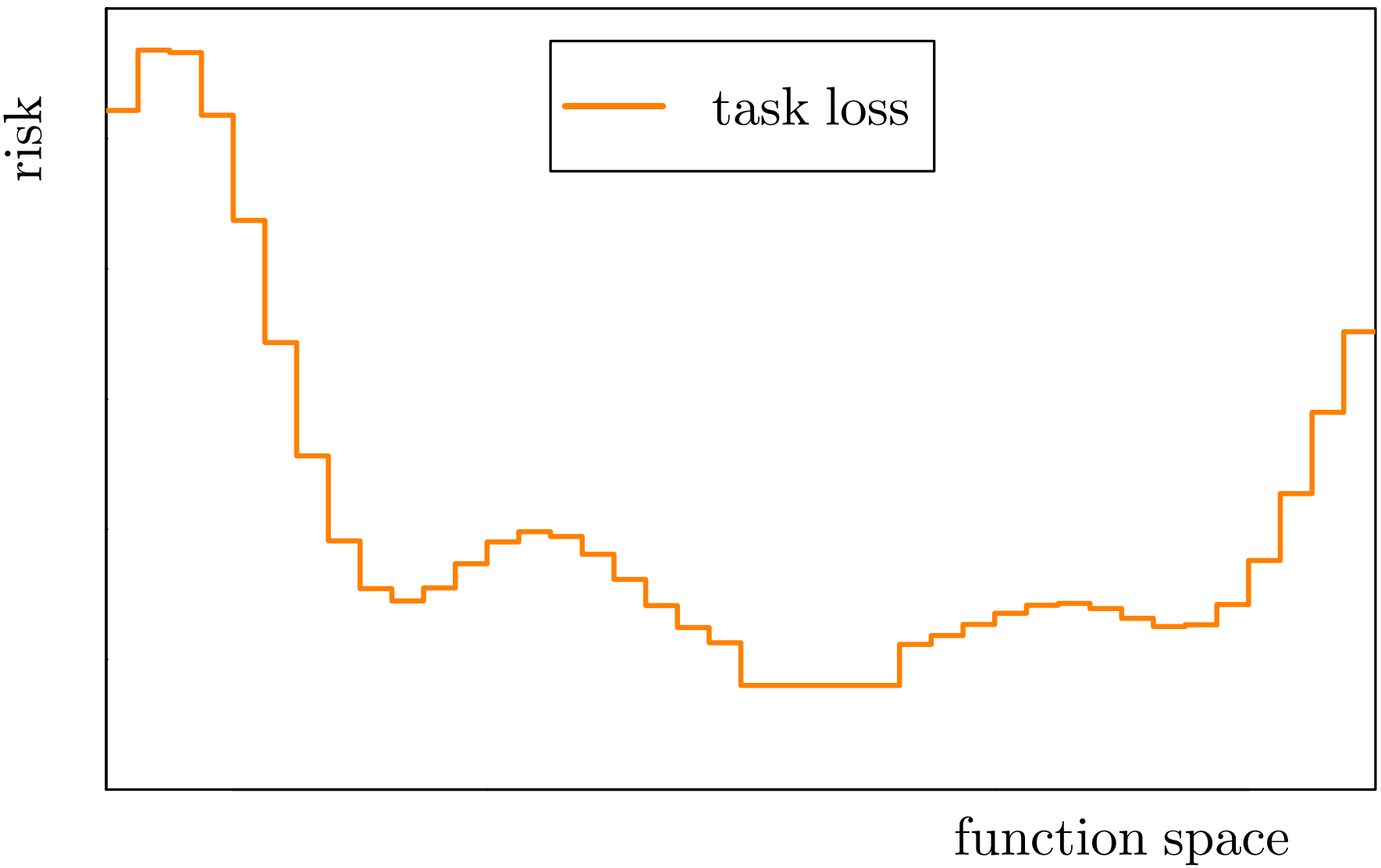
$$L_{\ell_{\mathrm{S}}}(f_n) \to \min_f L_{\ell_{\mathrm{S}}}(f) \implies L_{\ell_{\mathrm{T}}}(f_n) \to \min_f L_{\ell_{\mathrm{T}}}(f).$$

  or equivalently:

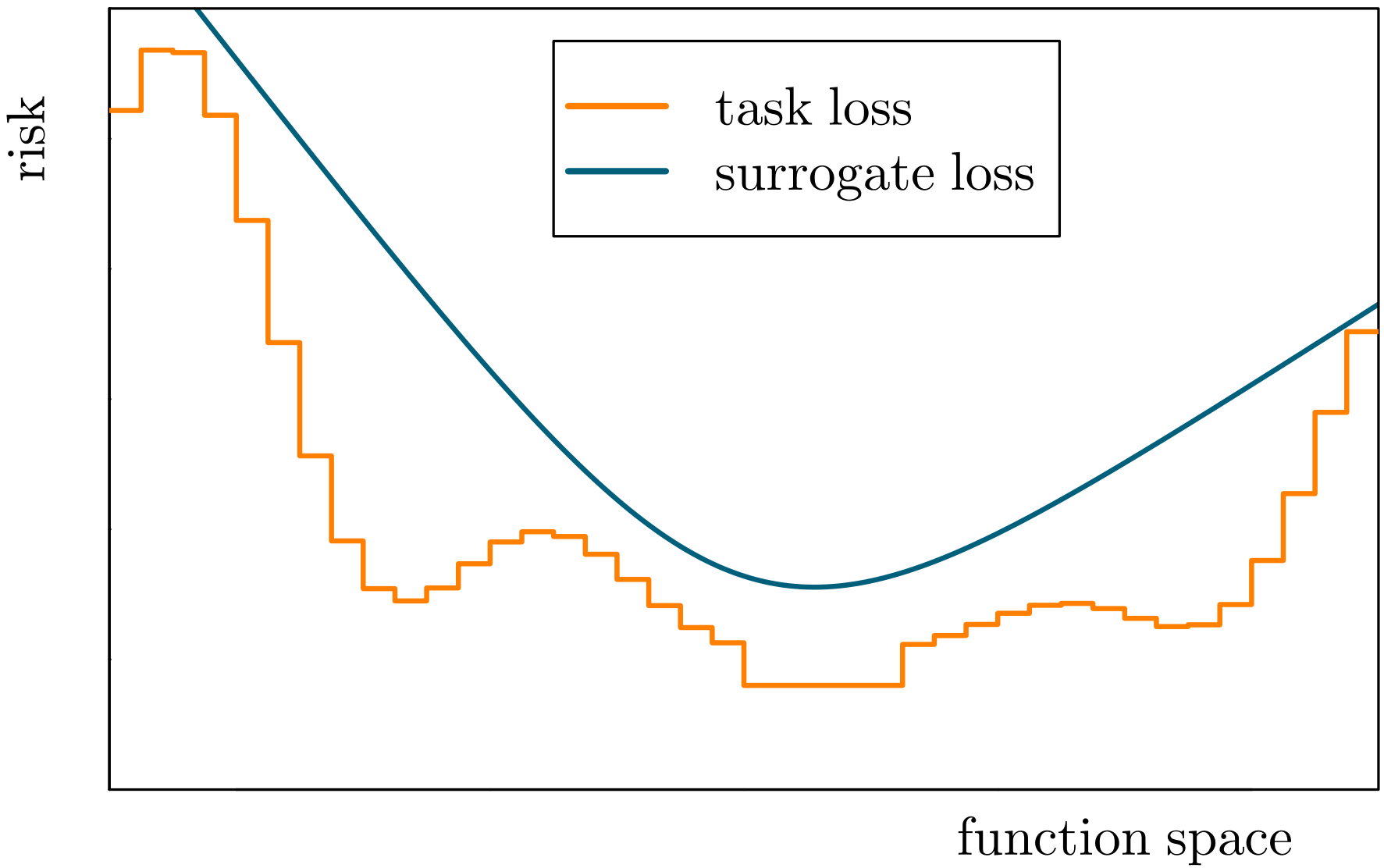
$$\mathrm{Reg}_{\ell_{\mathrm{S}}}(f_n) \to 0 \implies \mathrm{Reg}_{\ell_{\mathrm{T}}}(f_n) \to 0.$$

- Informally: if we converge to the prediction function **optimal with respect to** $\ell_{\mathrm{S}}$, then we also converge to the prediction function **optimal with respect to** $\ell_{\mathrm{T}}$.

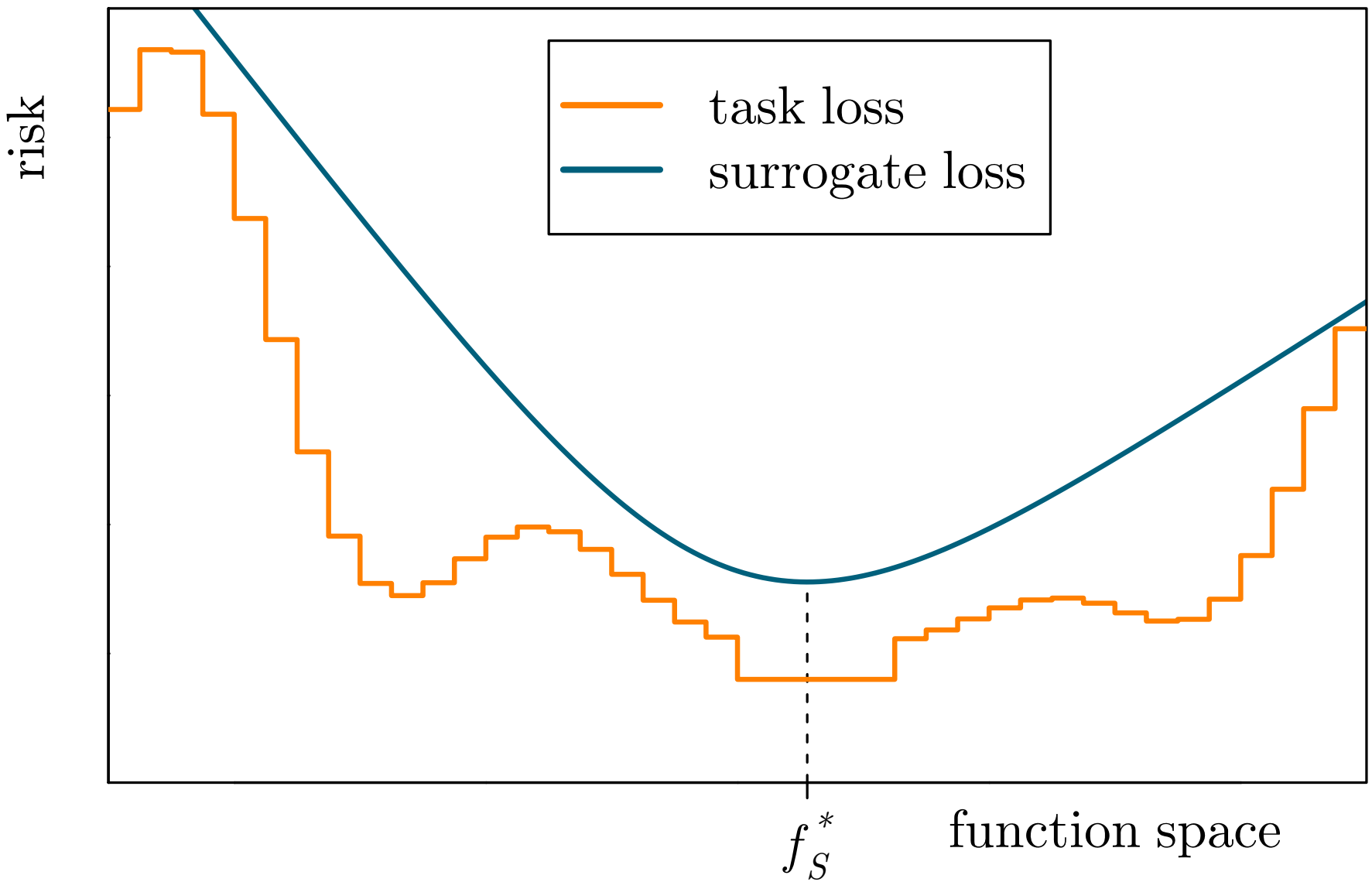
# Calibrated loss



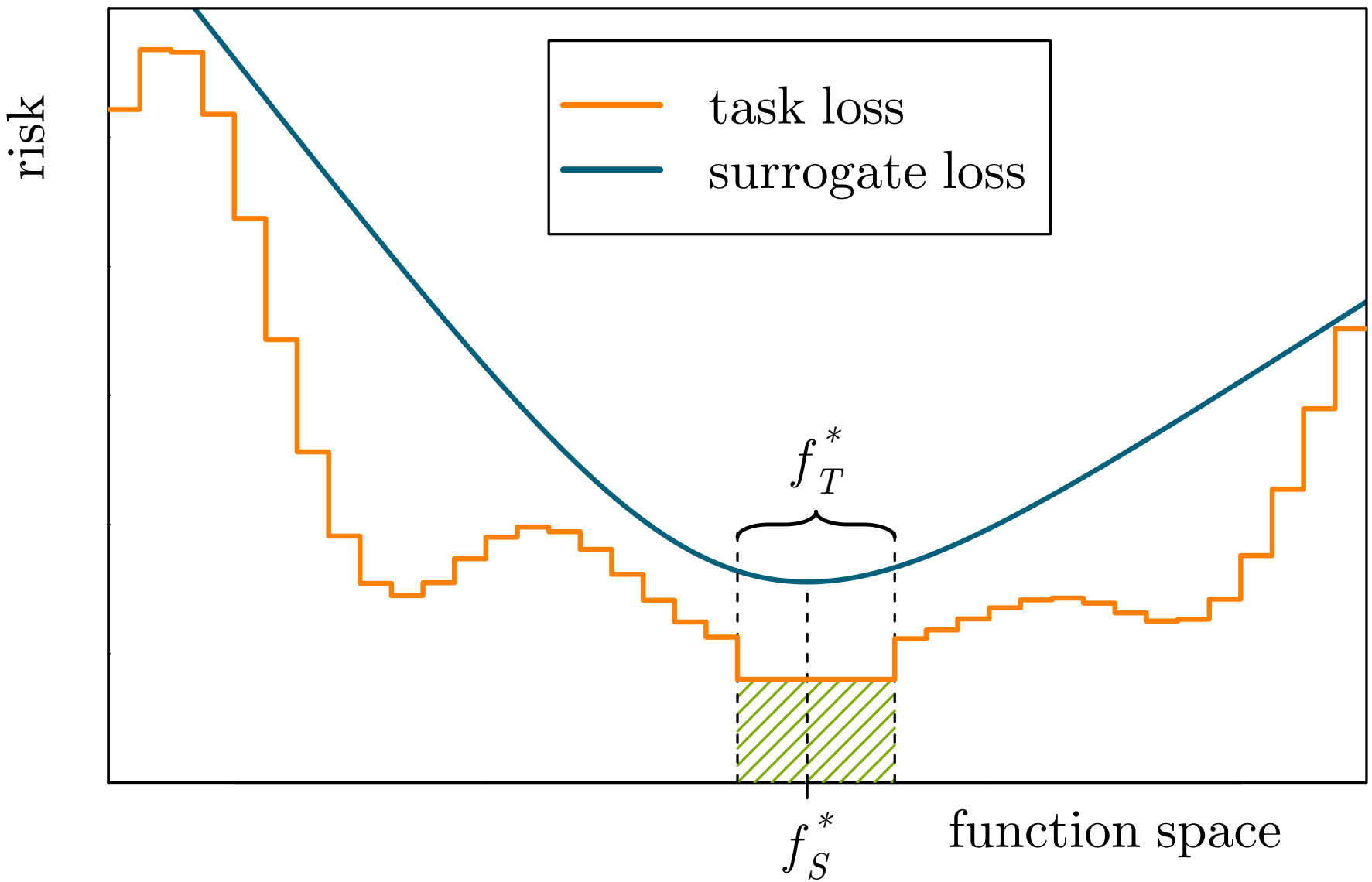
# Calibrated loss



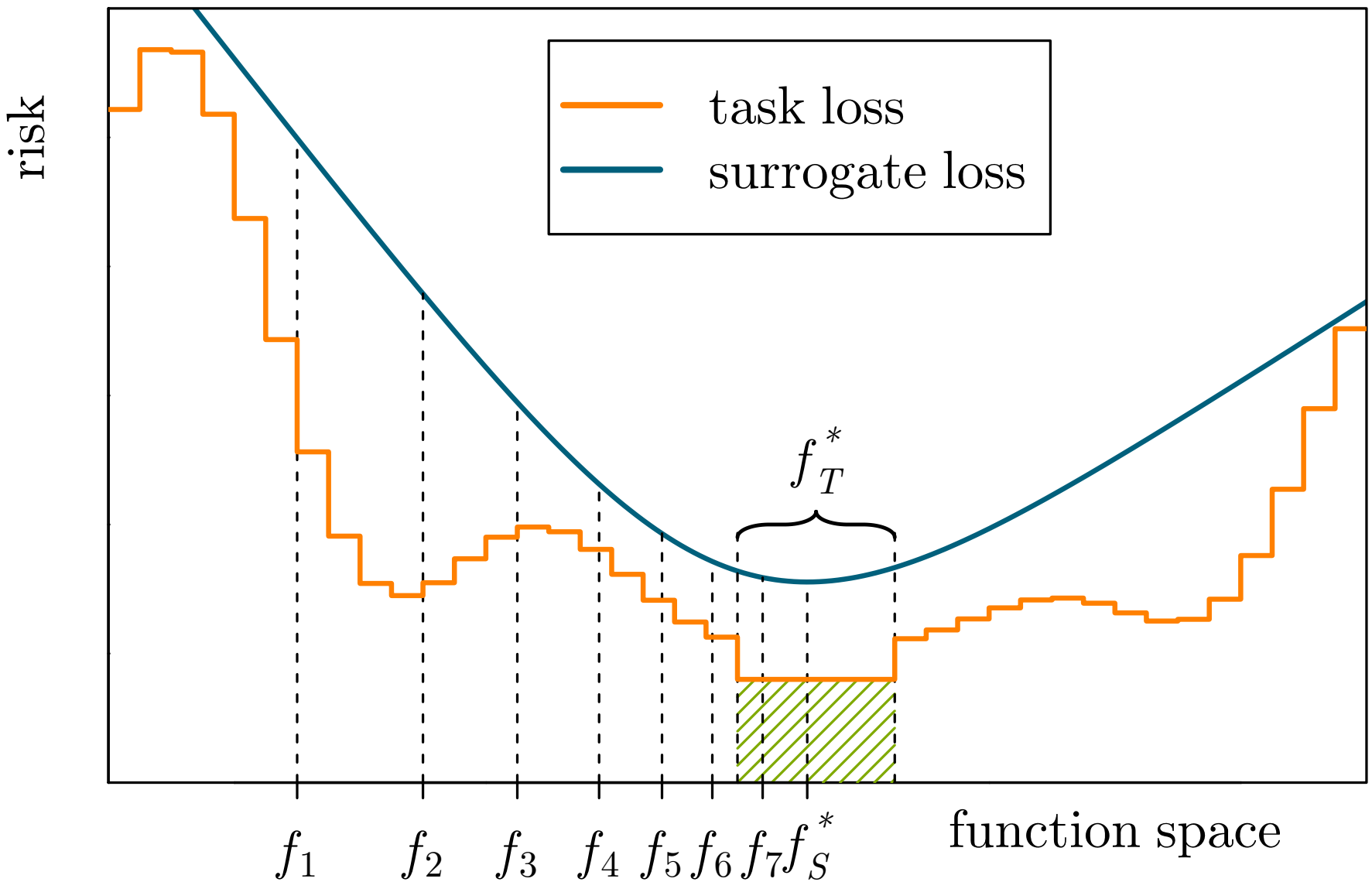
# Calibrated loss



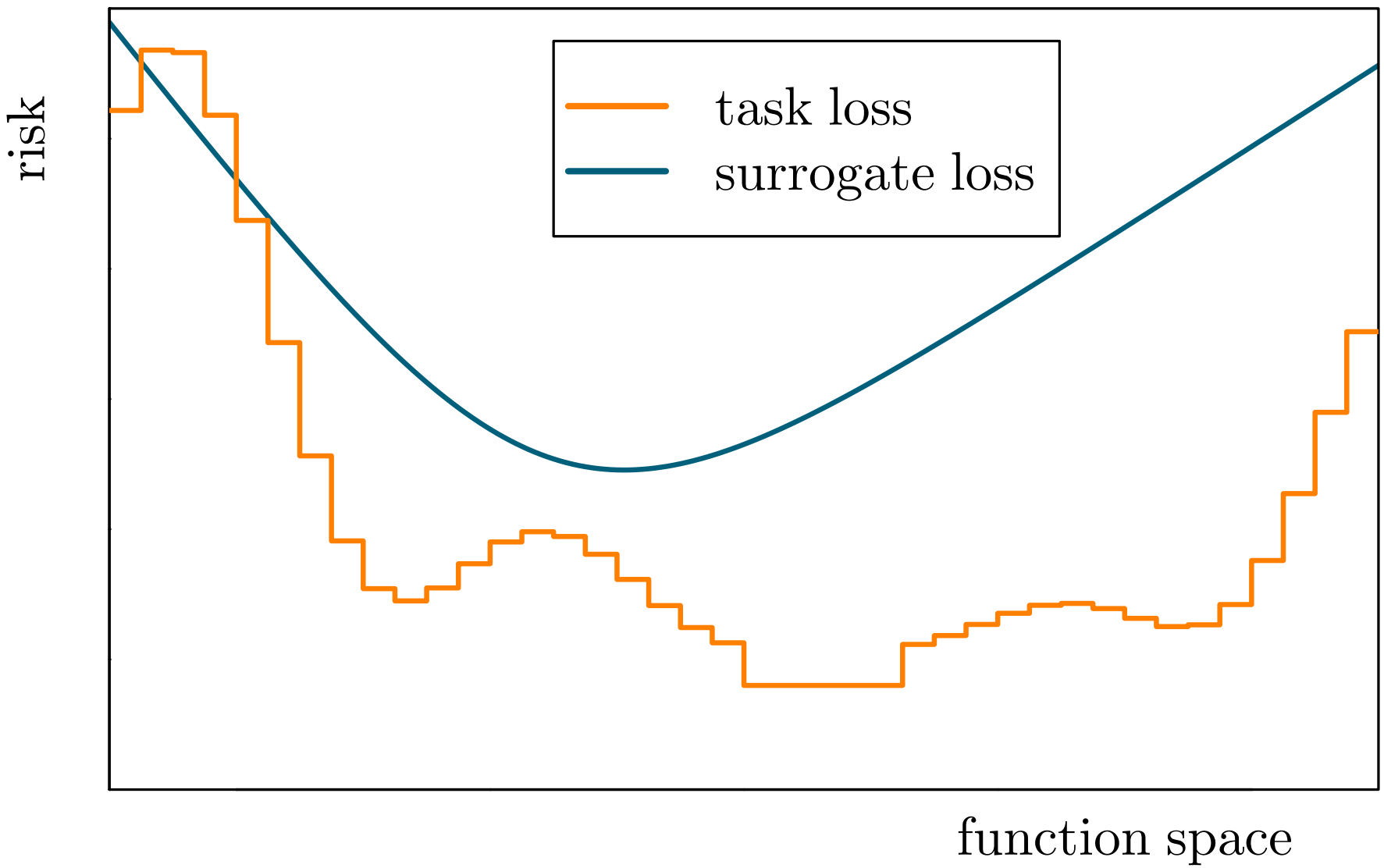
# Calibrated loss



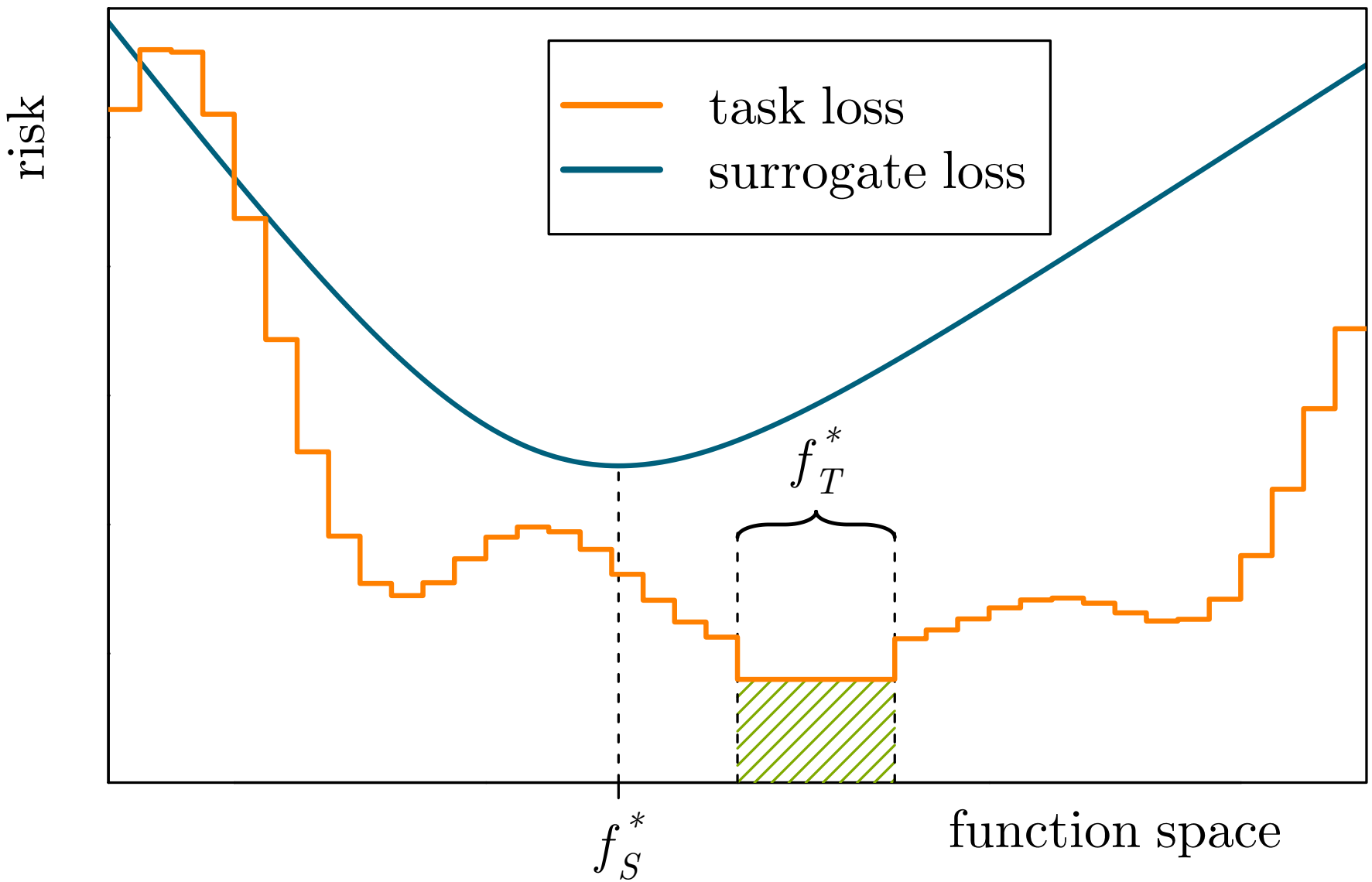
# Calibrated loss



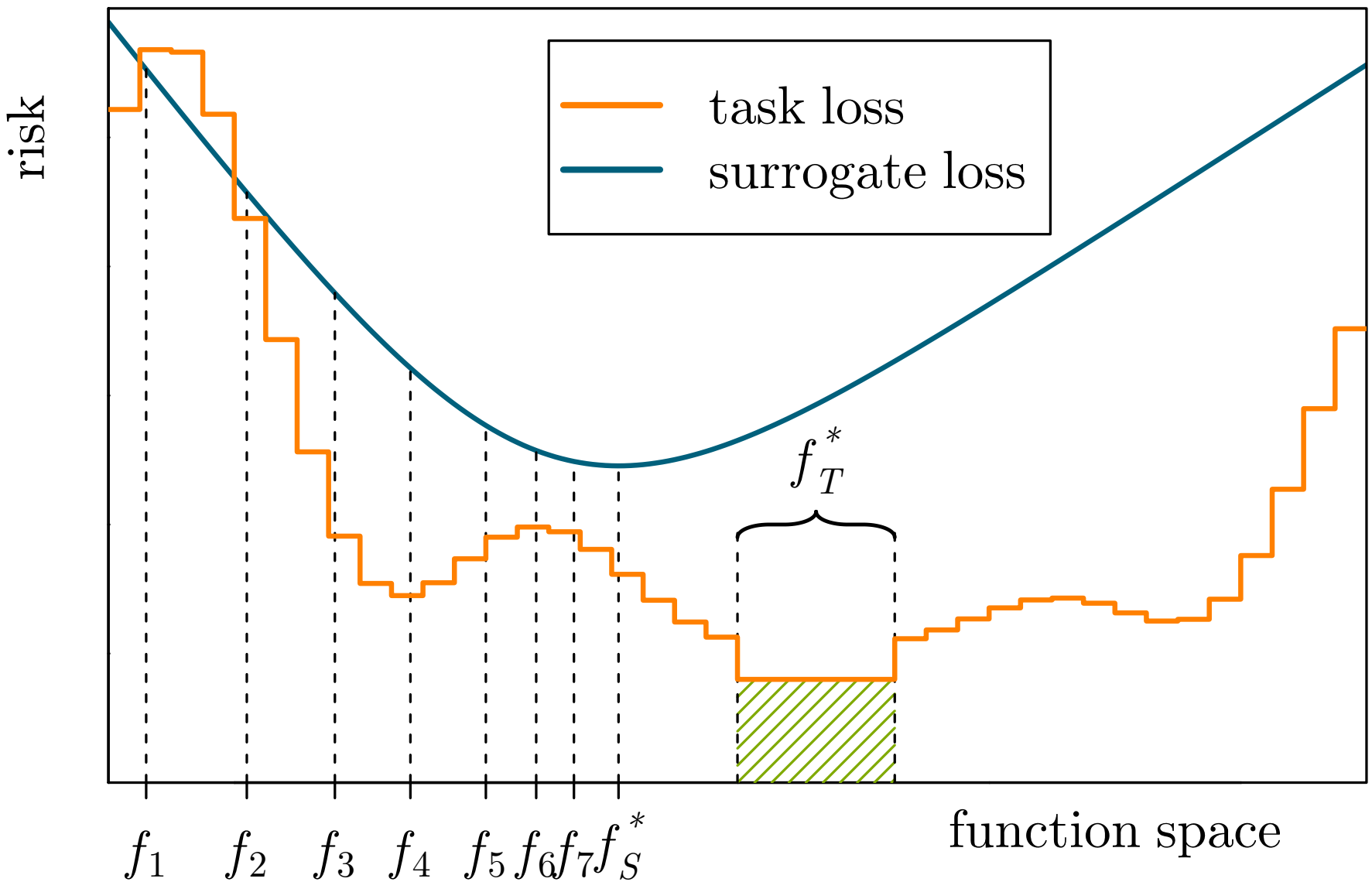
# Non-calibrated loss



# Non-calibrated loss



# Non-calibrated loss



## Classification calibration

- We say a surrogate loss $\ell$ is **classification calibrated (consistent)** if for any sequence of functions $\{f_n\}$ and any distribution $P(\boldsymbol{x}, y)$:

$$\mathrm{Reg}_{\ell}(f_n) \to 0 \Longrightarrow \mathrm{Reg}_{0/1}(f_n) \to 0.$$

## Classification calibration

- We say a surrogate loss $\ell$ is **classification calibrated (consistent)** if for any sequence of functions $\{f_n\}$ and any distribution $P(\boldsymbol{x}, y)$:

$$\mathrm{Reg}_\ell(f_n) \to 0 \Longrightarrow \mathrm{Reg}_{0/1}(f_n) \to 0.$$

- We only consider **margin-based losses**: $\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x}))$.

# Classification calibration

- We say a surrogate loss $\ell$ is **classification calibrated (consistent)** if for any sequence of functions $\{f_n\}$ and any distribution $P(\boldsymbol{x}, y)$:

$$\mathrm{Reg}_\ell(f_n) \to 0 \implies \mathrm{Reg}_{0/1}(f_n) \to 0.$$

- We only consider **margin-based losses**: $\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x}))$.
- Holds for any $P(\boldsymbol{x}, y) \Rightarrow$ must hold conditionally for any $\boldsymbol{x}$:

Question: why?

$$\mathrm{Reg}_\ell(f_n|\boldsymbol{x}) \to 0 \implies \mathrm{Reg}_{0/1}(f_n|\boldsymbol{x}) \to 0, \qquad \text{or}$$

$$L_\ell(f_n|\boldsymbol{x}) \to \min_f L_\ell(f|\boldsymbol{x}) \implies L_{0/1}(f_n|\boldsymbol{x}) \to \min_f L_{0/1}(f|\boldsymbol{x}).$$

## Classification calibration

- We say a surrogate loss $\ell$ is **classification calibrated (consistent)** if for any sequence of functions $\{f_n\}$ and any distribution $P(\boldsymbol{x}, y)$:

$$\mathrm{Reg}_\ell(f_n) \to 0 \Longrightarrow \mathrm{Reg}_{0/1}(f_n) \to 0.$$

- We only consider **margin-based losses**: $\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x}))$.
- Holds for any $P(\boldsymbol{x}, y) \Rightarrow$ must hold conditionally for any $\boldsymbol{x}$:

Question: why?

$$\mathrm{Reg}_\ell(f_n|\boldsymbol{x}) \to 0 \Longrightarrow \mathrm{Reg}_{0/1}(f_n|\boldsymbol{x}) \to 0, \qquad \text{or}$$

$$L_\ell(f_n|\boldsymbol{x}) \to \min_f L_\ell(f|\boldsymbol{x}) \Longrightarrow L_{0/1}(f_n|\boldsymbol{x}) \to \min_f L_{0/1}(f|\boldsymbol{x}).$$

- **Abstraction**: $L_\ell(f|\boldsymbol{x})$ depends **only on two numbers**: $f(\boldsymbol{x})$ and $\eta(\boldsymbol{x})$. Also, $\min_f L_\ell(f|\boldsymbol{x})$ only depends on a **single number** $\eta(\boldsymbol{x})$.

## Classification calibration

- We say a surrogate loss $\ell$ is **classification calibrated (consistent)** if for any sequence of functions $\{f_n\}$ and any distribution $P(\boldsymbol{x}, y)$:

$$\mathrm{Reg}_\ell(f_n) \to 0 \Longrightarrow \mathrm{Reg}_{0/1}(f_n) \to 0.$$

- We only consider **margin-based losses**: $\ell(y, f(\boldsymbol{x})) = \ell(yf(\boldsymbol{x}))$.
- Holds for any $P(\boldsymbol{x}, y) \Rightarrow$ must hold conditionally for any $\boldsymbol{x}$:

Question: why?

$$\mathrm{Reg}_\ell(f_n|\boldsymbol{x}) \to 0 \Longrightarrow \mathrm{Reg}_{0/1}(f_n|\boldsymbol{x}) \to 0, \qquad \text{or}$$

$$L_\ell(f_n|\boldsymbol{x}) \to \min_f L_\ell(f|\boldsymbol{x}) \Longrightarrow L_{0/1}(f_n|\boldsymbol{x}) \to \min_f L_{0/1}(f|\boldsymbol{x}).$$

- **Abstraction**: $L_\ell(f|\boldsymbol{x})$ depends **only on two numbers**: $f(\boldsymbol{x})$ and $\eta(\boldsymbol{x})$. Also, $\min_f L_\ell(f|\boldsymbol{x})$ only depends on a **single number** $\eta(\boldsymbol{x})$.
- We drop the dependence on $\boldsymbol{x}$ and test convergence for any sequence of **numbers** $\{f_n\}$ and any **number** $\eta$.

# Classification calibration[3]

- A shorthand notation for **conditional** $0/1$**-risk** $L_{0/1}(f|\boldsymbol{x})$:

$$C_\eta^{0/1}(f) = \eta[\![f < 0]\!] + (1-\eta)[\![f > 0]\!] + \frac{1}{2}[\![f = 0]\!].$$

[3] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006

# Classification calibration[3]

- A shorthand notation for **conditional** $0/1$**-risk** $L_{0/1}(f|\boldsymbol{x})$:

$$C_\eta^{0/1}(f) = \eta[\![f < 0]\!] + (1-\eta)[\![f > 0]\!] + \frac{1}{2}[\![f = 0]\!].$$

- $C_\eta^{0/1}(f)$ is **minimized** by any $f^*$, s.t. $\mathbf{sgn}\,(f^*) = \mathbf{sgn}\,(\eta - 1/2)$ for $\eta \neq 1/2$, and $f^*$ **arbitrary** for $\eta = 1/2$.
  - The **Bayes** $(0/1)$ **classifier** $f^*$ is **non unique**!

[3] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006

# Classification calibration[3]

- A shorthand notation for **conditional** $0/1$**-risk** $L_{0/1}(f|\boldsymbol{x})$:

$$C_\eta^{0/1}(f) = \eta[\![f < 0]\!] + (1-\eta)[\![f > 0]\!] + \frac{1}{2}[\![f = 0]\!].$$

- $C_\eta^{0/1}(f)$ is **minimized** by any $f^*$, s.t. $\mathbf{sgn}\,(f^*) = \mathbf{sgn}\,(\eta - 1/2)$ for $\eta \neq 1/2$, and $f^*$ **arbitrary** for $\eta = 1/2$.
  - The **Bayes** $(0/1)$ **classifier** $f^*$ is **non unique**!
- The **Bayes** $0/1$**-risk**:

$$\min_f C_\eta^{0/1}(f) = \min\{\eta, 1-\eta\},$$

[3] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006

## Classification calibration

- A shorthand notation for **conditional $\ell$-risk** $L_\ell(f|\boldsymbol{x})$:

$$C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f).$$

# Classification calibration

- A shorthand notation for **conditional $\ell$-risk** $L_\ell(f|\boldsymbol{x})$:

$$C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f).$$

- **Bayes conditional $\ell$-risk**:

$$H(\eta) := \min_f C_\eta(f).$$

## Classification calibration

- A shorthand notation for **conditional $\ell$-risk** $L_\ell(f|\boldsymbol{x})$:

$$C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f).$$

- **Bayes conditional $\ell$-risk**:

$$H(\eta) := \min_f C_\eta(f).$$

- Loss $\ell$ is **classification calibrated** if any only if for any $\{f_n\}$ and $\eta$:

$$C_\eta(f_n) \to H(\eta) \Longrightarrow C_\eta^{0/1}(f_n) \to \min\{\eta, 1-\eta\}.$$

## Classification calibration

- To have $C_\eta^{0/1}(f_n) \to \min\{\eta, 1-\eta\}$, for any $\eta \neq 1/2$, $f_n$ must **eventually** pick the **correct sign**, i.e. for large $n$,

$$\operatorname{sgn}(f_n) = \operatorname{sgn}(\eta - 1/2), \qquad \text{or} \qquad f_n(\eta - 1/2) > 0.$$

## Classification calibration

- To have $C_\eta^{0/1}(f_n) \to \min\{\eta, 1-\eta\}$, for any $\eta \neq 1/2$, $f_n$ must **eventually** pick the **correct sign**, i.e. for large $n$,

$$\operatorname{sgn}(f_n) = \operatorname{sgn}(\eta - 1/2), \qquad \text{or} \qquad f_n(\eta - 1/2) > 0.$$

- Since $C_\eta(f_n) \to \min_f C_\eta(f)$, this means any minimizer $f^*$ of $C_\eta(f)$ must satisfy $f^*(\eta - 1/2) > 0$.

## Classification calibration

- To have $C_\eta^{0/1}(f_n) \to \min\{\eta, 1-\eta\}$, for any $\eta \neq 1/2$, $f_n$ must **eventually** pick the **correct sign**, i.e. for large $n$,

$$\operatorname{sgn}(f_n) = \operatorname{sgn}(\eta - 1/2), \qquad \text{or} \qquad f_n(\eta - 1/2) > 0.$$

- Since $C_\eta(f_n) \to \min_f C_\eta(f)$, this means any minimizer $f^*$ of $C_\eta(f)$ must satisfy $f^*(\eta - 1/2) > 0.$ , i.e.:

$$H(\eta) = \min_f C_\eta(f) < \min_{f\colon f(\eta-1/2)\leq 0} C_\eta(f) =: H^-(\eta).$$

## Classification calibration

- To have $C_\eta^{0/1}(f_n) \to \min\{\eta, 1-\eta\}$, for any $\eta \neq 1/2$, $f_n$ must **eventually** pick the **correct sign**, i.e. for large $n$,

$$\operatorname{sgn}(f_n) = \operatorname{sgn}(\eta - 1/2), \qquad \text{or} \qquad f_n(\eta - 1/2) > 0.$$

- Since $C_\eta(f_n) \to \min_f C_\eta(f)$, this means any minimizer $f^*$ of $C_\eta(f)$ must satisfy $f^*(\eta - 1/2) > 0.$ , i.e.:

$$H(\eta) = \min_f C_\eta(f) < \min_{f\colon f(\eta - 1/2) \leq 0} C_\eta(f) =: H^-(\eta).$$

- **Fact**: $\ell$ is **classification calibrated if and only if** for any $\eta \neq 1/2$,

$$H(\eta) < H^-(\eta).$$

The minimum of the conditional $\ell$-risk is always achieved on the **correct side of** $0$.

## Classification calibration

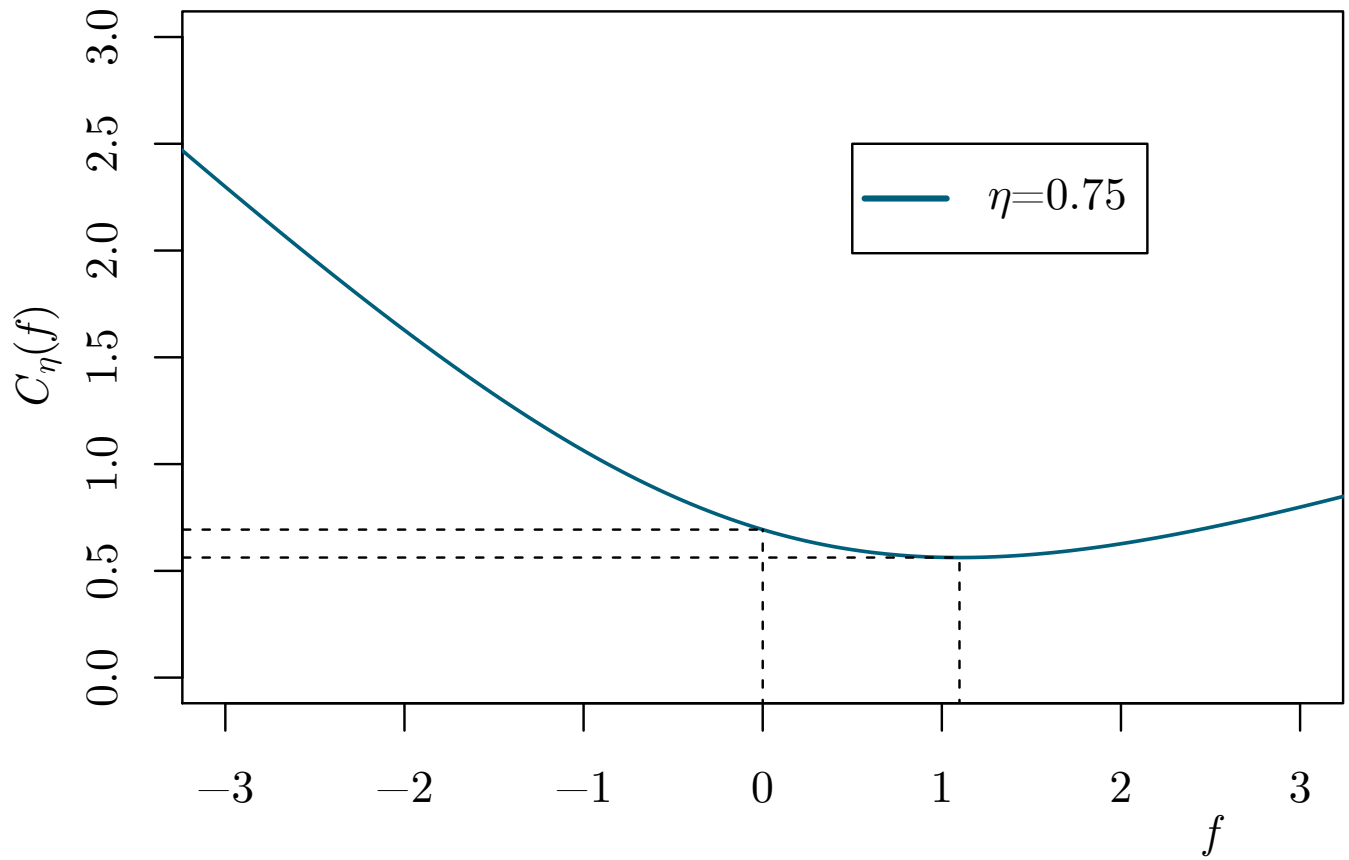
Loss $\ell$ is **classification calibrated** if and only if $H(\eta) < H^-(\eta)$, i.e.

$$\min_f C_\eta(f) < \min_{f\colon f(\eta-1/2)\leq 0} C_\eta(f).$$

# Classification calibration

Loss $\ell$ is **classification calibrated** if and only if $H(\eta) < H^-(\eta)$, i.e.

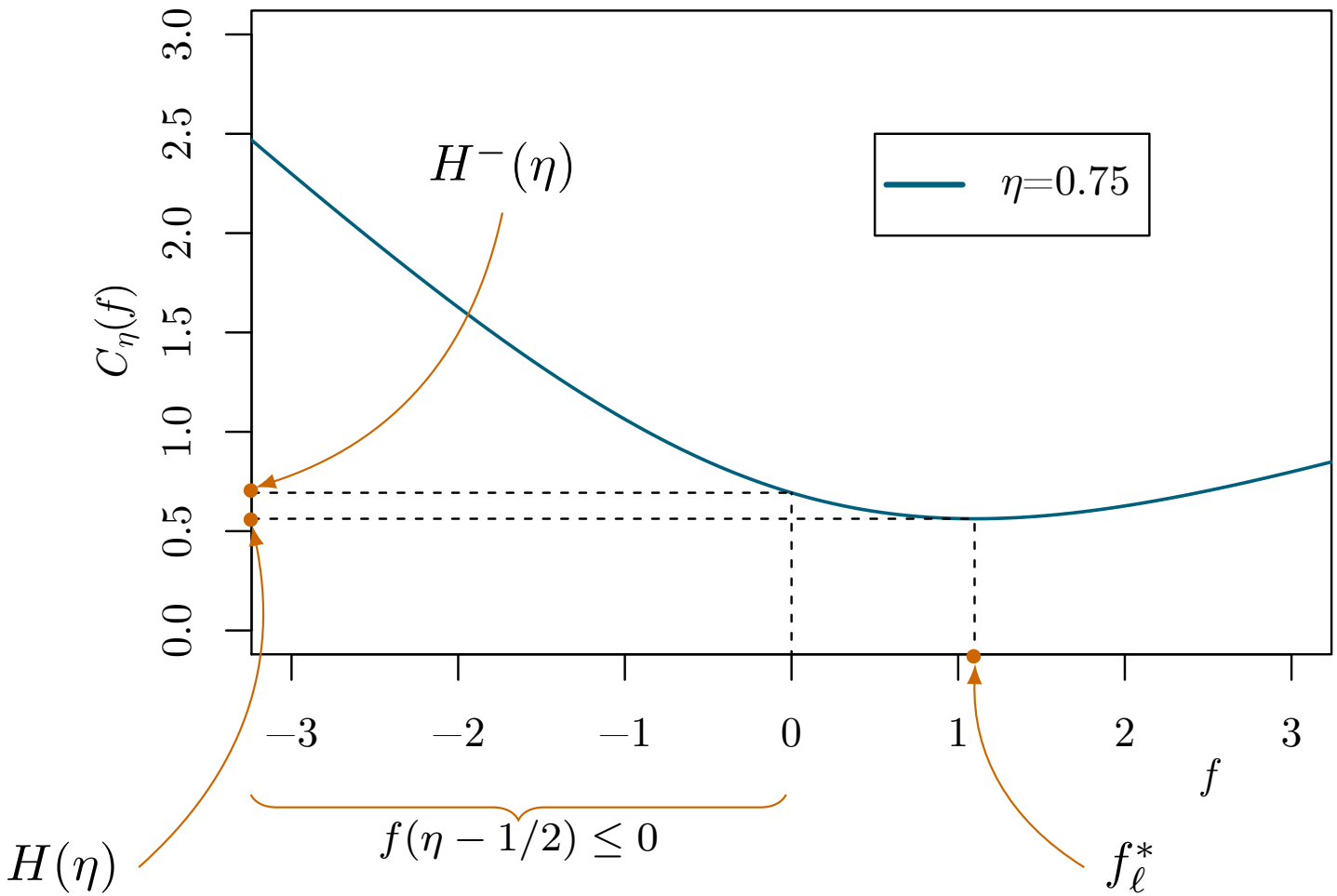
$$\min_f C_\eta(f) < \min_{f:\, f(\eta-1/2)\leq 0} C_\eta(f).$$



# Classification calibration

Loss $\ell$ is **classification calibrated** if and only if $H(\eta) < H^-(\eta)$, i.e.

$$\min_f C_\eta(f) < \min_{f\colon f(\eta-1/2)\le 0} C_\eta(f).$$



## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longleftarrow$

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longleftarrow$

First take $\eta > 1/2$ and assume $\ell'(0) < 0$.

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longleftarrow$

First take $\eta > 1/2$ and assume $\ell'(0) < 0$.

$\ell(f)$ is convex, then so is $C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f)$ (why?).

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longleftarrow$

First take $\eta > 1/2$ and assume $\ell'(0) < 0$.

$\ell(f)$ is convex, then so is $C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f)$ (why?).
Moreover,

$$C'_\eta(0) = (2\eta - 1)\ell'(0) < 0.$$

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longleftarrow$

First take $\eta > 1/2$ and assume $\ell'(0) < 0$.

$\ell(f)$ is convex, then so is $C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f)$ (why?).
Moreover,

$$C'_\eta(0) = (2\eta - 1)\ell'(0) < 0.$$

For convex function the derivative is **increasing**, so the minimum must be on the **positive side** of $0$.

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longleftarrow$

First take $\eta > 1/2$ and assume $\ell'(0) < 0$.

$\ell(f)$ is convex, then so is $C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f)$ (why?).
Moreover,

$$C'_\eta(0) = (2\eta - 1)\ell'(0) < 0.$$

For convex function the derivative is **increasing**, so the minimum must be on the **positive side** of $0$.

Similar argument for $\eta < 1/2$.

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longrightarrow$

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longrightarrow$

First take $\eta > 1/2$ and assume $\ell(f)$ is **calibrated**.

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longrightarrow$

First take $\eta > 1/2$ and assume $\ell(f)$ is **calibrated**.

Convexity of $\ell$ implies that for any $f$, $\ell(f) \geq \ell(0) + \ell'(0)f$ (tangent to the function chart at $0$ is always below the chart).

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longrightarrow$

First take $\eta > 1/2$ and assume $\ell(f)$ is **calibrated**.

Convexity of $\ell$ implies that for any $f$, $\ell(f) \geq \ell(0) + \ell'(0)f$ (tangent to the function chart at $0$ is always below the chart).

We get:

$$C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f) \geq \ell(0) + \ell'(0)(2\eta - 1).$$

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longrightarrow$

First take $\eta > 1/2$ and assume $\ell(f)$ is **calibrated**.

Convexity of $\ell$ implies that for any $f$, $\ell(f) \geq \ell(0) + \ell'(0)f$ (tangent to the function chart at $0$ is always below the chart).

We get:

$$C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f) \geq \ell(0) + \ell'(0)(2\eta - 1).$$

The r.h.s. must be **less than** $\ell(0)$ (so that $\ell'(0) < 0$). Otherwise, $C_\eta(f) \geq \ell(0) = C_\eta(0)$, which implies $f = 0$ is the minimizer of $\ell$, so $\ell(f)$ is not calibrated.

## Calibration — simplified condition

A convex loss $\ell(f)$ is classification calibrated **if and only if** it is **differentiable at** $0$ and $\ell'(0) < 0$.

**Proof (sketch):**
(only show the proof for $\ell$ differentiable at $0$)

$\Longrightarrow$

First take $\eta > 1/2$ and assume $\ell(f)$ is **calibrated**.

Convexity of $\ell$ implies that for any $f$, $\ell(f) \geq \ell(0) + \ell'(0)f$ (tangent to the function chart at $0$ is always below the chart).

We get:

$$C_\eta(f) = \eta\ell(f) + (1-\eta)\ell(-f) \geq \ell(0) + \ell'(0)(2\eta - 1).$$

The r.h.s. must be **less than** $\ell(0)$ (so that $\ell'(0) < 0$). Otherwise, $C_\eta(f) \geq \ell(0) = C_\eta(0)$, which implies $f = 0$ is the minimizer of $\ell$, so $\ell(f)$ is not calibrated. Similar argument for $\eta < 1/2$.

# Examples

## Question

**Prove** that all the following losses are calibrated and **derive** their Bayes classifiers:

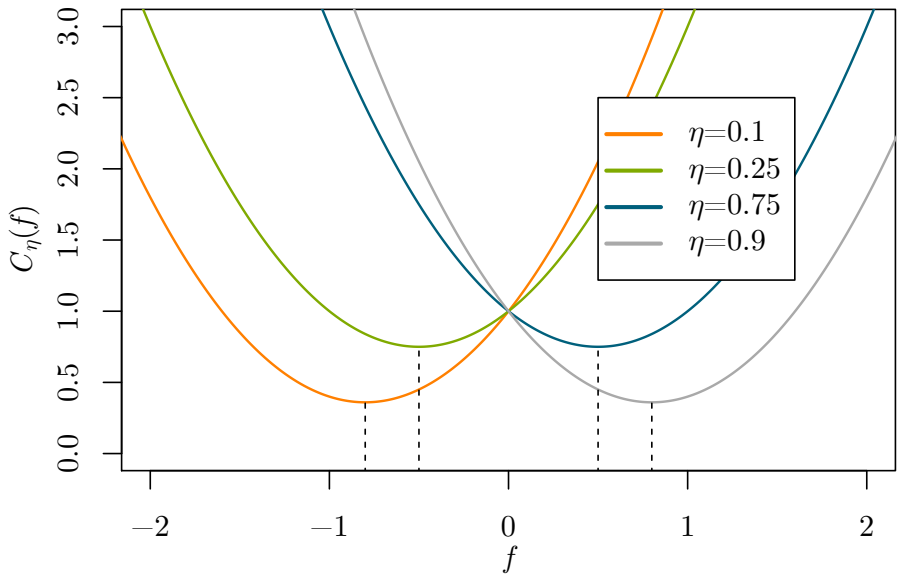
- squared error loss $\ell(f) = (1 - f)^2$,
- logistic loss $\ell(f) = \log\left(1 + e^{-f}\right)$,
- hinge loss $\ell(f) = \max\{0, 1 - f\}$,
- exponential loss $\ell(f) = e^{-f}$.

# Example — squared error loss

$$\ell(f) = (1-f)^2$$

**Conditional risk**:

$$\begin{aligned} C_\eta(f) &= \eta(1-f)^2 + (1-\eta)(1+f)^2 \\ &= 1 + f^2 + 2f(1-2\eta). \end{aligned}$$




**Minimized by**:

$$f^* = 2\eta - 1$$

**Calibration**:

$$\ell'(f) = -2(1-f) \quad \Longrightarrow \quad \ell'(0) = -2 < 0$$

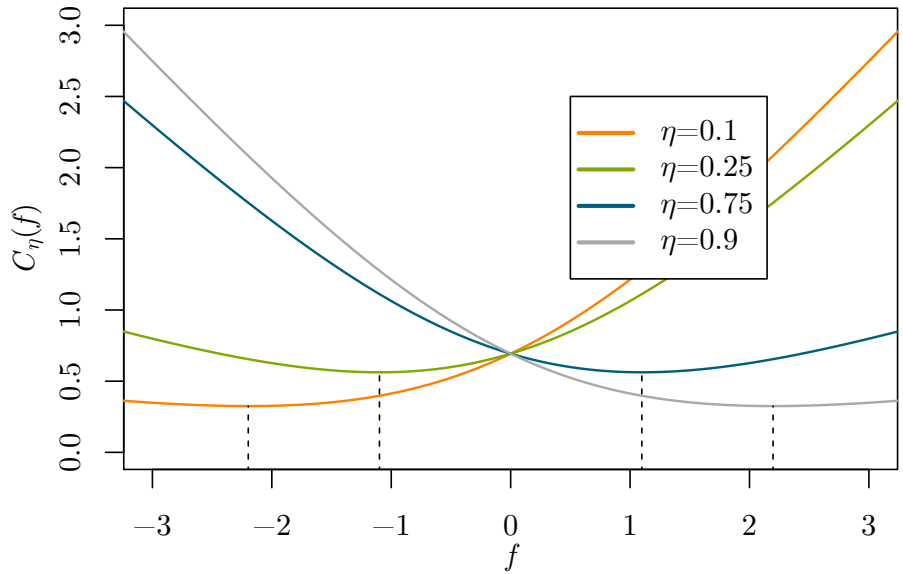
$$\begin{aligned} H(\eta) &= C_\eta(f^*) = 4\eta(1-\eta), \\ H^-(\eta) &= C_\eta(0) = 1 > H(\eta). \end{aligned}$$

# Example — logistic loss

$$\ell(f) = \log\left(1 + e^{-f}\right)$$

**Conditional risk**:

$$C_\eta(f) = \eta \log(1 + e^{-f}) + (1 - \eta) \log(1 + e^{f}).$$




**Minimized by**:

$$f^* = \log \frac{\eta}{1 - \eta}.$$

**Calibration**:

$$\ell'(f) = -\frac{1}{1 + e^{f}} \quad \Longrightarrow \quad \ell'(0) = -1/2 < 0$$

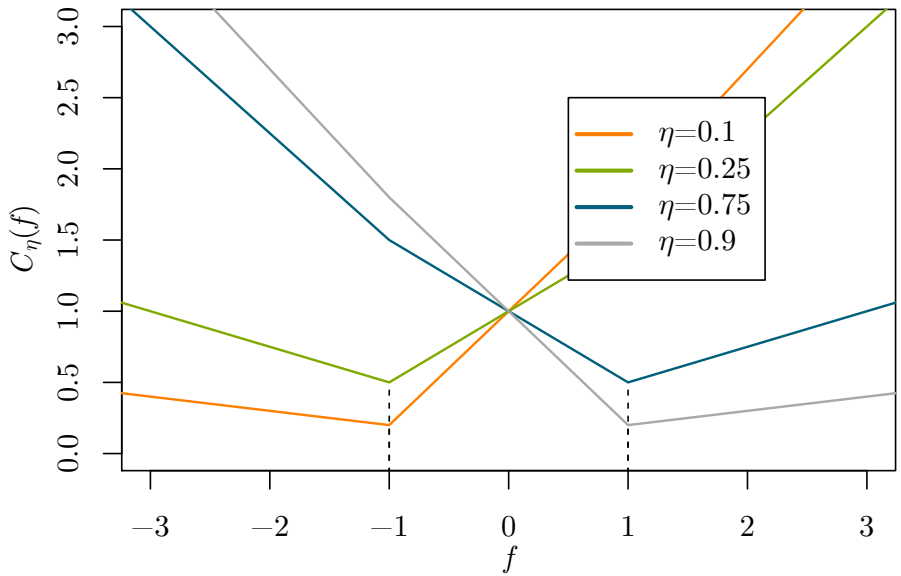
$$H(\eta) = C_\eta(f^*) = -\eta \log \eta - (1 - \eta) \log(1 - \eta)$$
$$H^-(\eta) = C_\eta(0) = \log 2 > H(\eta).$$

# Example — hinge loss

$$\ell(f) = \max\{0, 1 - f\}$$

**Conditional risk**:

$$C_\eta(f) = \eta \max\{0, 1 - f\} + (1 - \eta) \max\{0, 1 + f\}.$$




**Minimized by**:

$$f^* = \operatorname{sgn}(\eta - 1/2)$$

**Calibration**:

$$\ell'(f) = -1[\![f < 1]\!] \quad \Longrightarrow \quad \ell'(0) = -1 < 0$$

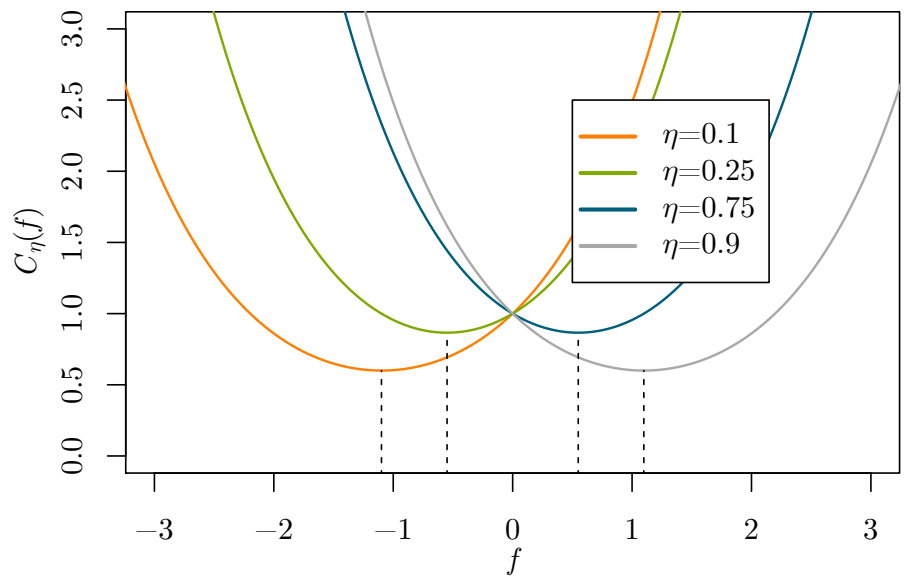
$$H(\eta) = C_\eta(f^*) = 2 \min\{\eta, 1 - \eta\}$$
$$H^-(\eta) = C_\eta(0) = 1 > H(\eta).$$

# Example — exponential loss

$$\ell(f) = e^{-f}$$

**Conditional risk**:

$$C_\eta(f) = \eta e^{-f} + (1-\eta)e^{f}$$




**Minimized by**:

$$f^* = \frac{1}{2}\log\frac{\eta}{1-\eta}$$

**Calibration**:

$$\ell'(f) = -e^{-f} \quad \Longrightarrow \quad \ell'(0) = -1 < 0$$

$$H(\eta) = C_\eta(f^*) = 2\sqrt{\eta(1-\eta)}$$
$$H^-(\eta) = C_\eta(0) = 1 > H(\eta).$$

## Calibration of surrogate losses

- All considered surrogate losses are **classification calibrated**.

## Calibration of surrogate losses

- All considered surrogate losses are **classification calibrated**.
- In each case $f^*$ is an increasing function of $\eta$.

## Calibration of surrogate losses

- All considered surrogate losses are **classification calibrated**.
- In each case $f^*$ is an increasing function of $\eta$.
- In all cases except hinge loss, $f^*(\eta)$ is **invertible**
  $\Longrightarrow$ we can estimate $\eta$ by inverting the relation.

# Calibration of surrogate losses

- All considered surrogate losses are **classification calibrated**.
- In each case $f^*$ is an increasing function of $\eta$.
- In all cases except hinge loss, $f^*(\eta)$ is **invertible**
  $\Longrightarrow$ we can estimate $\eta$ by inverting the relation.

| loss | $f^*(\eta)$ | $\eta(f^*)$ |
|---|---|---|
| squared error | $2\eta - 1$ | $\frac{1+f^*}{2}$ |
| logistic | $\log \frac{\eta}{1-\eta}$ | $\frac{1}{1+e^{-f^*}}$ |
| exponential | $\frac{1}{2} \log \frac{\eta}{1-\eta}$ | $\frac{1}{1+e^{-2f^*}}$ |
| hinge | $\operatorname{sgn}(\eta - 1/2)$ | doesn't exist |

# Calibration of surrogate losses

- All considered surrogate losses are **classification calibrated**.
- In each case $f^*$ is an increasing function of $\eta$.
- In all cases except hinge loss, $f^*(\eta)$ is **invertible**
  $\Longrightarrow$ we can estimate $\eta$ by inverting the relation.

| loss | $f^*(\eta)$ | $\eta(f^*)$ |
|---|---|---|
| squared error | $2\eta - 1$ | $\frac{1+f^*}{2}$ |
| logistic | $\log \frac{\eta}{1-\eta}$ | $\frac{1}{1+e^{-f^*}}$ |
| exponential | $\frac{1}{2} \log \frac{\eta}{1-\eta}$ | $\frac{1}{1+e^{-2f^*}}$ |
| hinge | $\operatorname{sgn}(\eta - 1/2)$ | doesn't exist |

- Minimizing hinge loss **does not** lead to probability estimation.

# Outline

# Training vs. test error

**Training error of function $h$:**

$$\widehat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\Longrightarrow$ Average error on a sample of size $n$.

# Training vs. test error

**Training error of function $h$:**

$$\widehat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\Longrightarrow$ Average error on a sample of size $n$.

**Test error of function $h$:**

$$L(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P} \left[\ell(y, h(\boldsymbol{x}))\right]$$

$\Longrightarrow$ Expected error $h$ makes on a randomly drawn instance.

**Generalization error of function $h$:** $L(h) - \widehat{L}(h)$
(test error $-$ training error)

# Training vs. test error

**Training error of function $h$:**

$$\widehat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\Longrightarrow$ Average error on a sample of size $n$.

**Test error of function $h$:**

$$L(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P} \left[\ell(y, h(\boldsymbol{x}))\right]$$

$\Longrightarrow$ Expected error $h$ makes on a randomly drawn instance.

**Generalization error of function $h$:** $L(h) - \widehat{L}(h)$
(test error $-$ training error)

- For fixed $h$, $\widehat{L}(h) \simeq L(h)$ (law of large numbers).

# Training vs. test error

**Training error of function $h$:**

$$\widehat{L}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i))$$

$\Longrightarrow$ Average error on a sample of size $n$.

**Test error of function $h$:**

$$L(h) = \mathbb{E}_{(\boldsymbol{x},y)\sim P} \left[\ell(y, h(\boldsymbol{x}))\right]$$

$\Longrightarrow$ Expected error $h$ makes on a randomly drawn instance.

**Generalization error of function $h$:** $L(h) - \widehat{L}(h)$
(test error $-$ training error)

- For fixed $h$, $\widehat{L}(h) \simeq L(h)$ (law of large numbers).
- But $h$ is not fixed, it is **chosen** based on the training sample!

# Empirical Risk Minimization (ERM)

- Choose a prediction function $\widehat{h}$ which minimizes the loss on the training data within some **restricted** class of functions $\mathcal{H}$.

$$\widehat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i)).$$

- The average loss on the training data is called **empirical risk** $\widehat{L}(h)$.

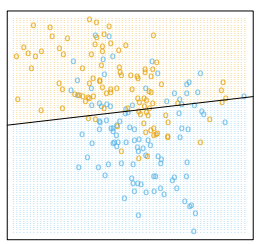
4

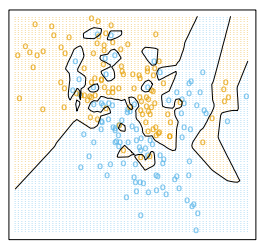
# Empirical Risk Minimization (ERM)

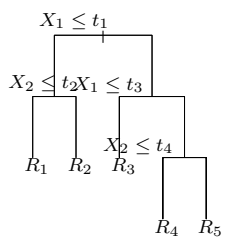
- Choose a prediction function $\widehat{h}$ which minimizes the loss on the training data within some **restricted** class of functions $\mathcal{H}$.

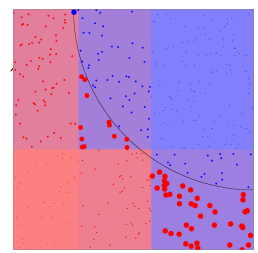
$$\widehat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(\boldsymbol{x}_i)).$$

- The average loss on the training data is called **empirical risk** $\widehat{L}(h)$.
- $\mathcal{H}$ can be: linear functions, polynomials, trees of a given depth, rules, linear combinations of trees, etc.[4]







[4] T. Hastie, R. Tibshirani, and J. H. Friedman. *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2003

# Overfitting

**Can you think of a "learning" algorithm which always have zero training error and poor test error?**

# Overfitting

**Can you think of a "learning" algorithm which always have zero training error and poor test error?**

**"Memorization"**: given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, choose:

$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \text{ for some } i = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

# Overfitting

**Can you think of a "learning" algorithm which always have zero training error and poor test error?**

**"Memorization"**: given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, choose:

$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \text{ for some } i = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

It is **very easy** to drop training error down to $0$ if our function class is flexible enough.

# Overfitting

**Can you think of a "learning" algorithm which always have zero training error and poor test error?**

**"Memorization"**: given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, choose:

$$h(\boldsymbol{x}) = \begin{cases} y_i & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \text{ for some } i = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

It is **very easy** to drop training error down to $0$ if our function class is flexible enough.

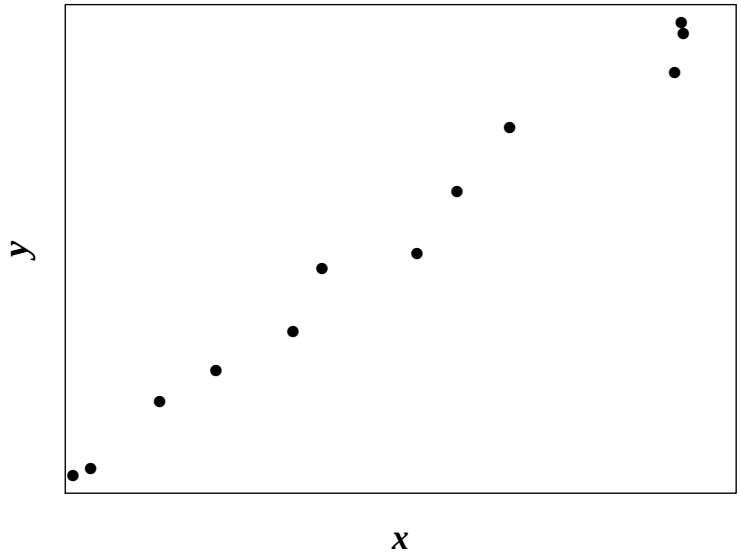
This phenomenon is called **overfitting** to the data.

**Overfitting** – producing $h$ with large **generalization error**

# Overfitting with ERM

Simple artificial data set generator:

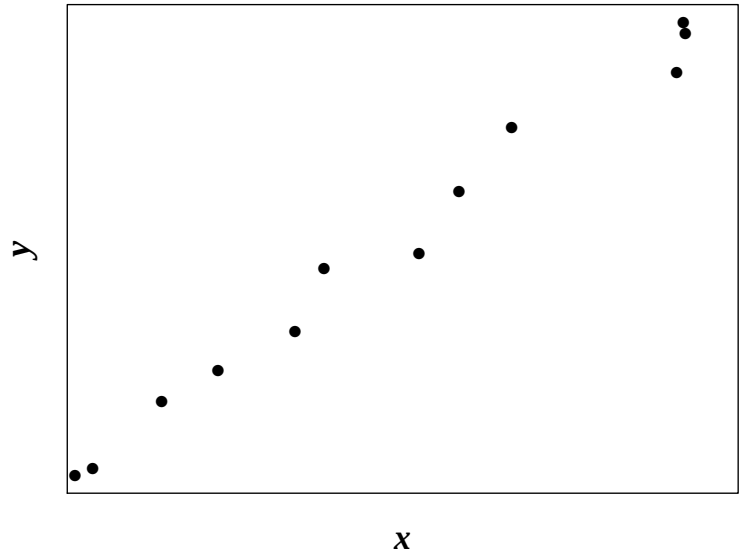
$$x \sim \text{uniform}(0, 1),$$
$$y = x + \epsilon, \qquad \epsilon \sim N(0,\ 0.05).$$



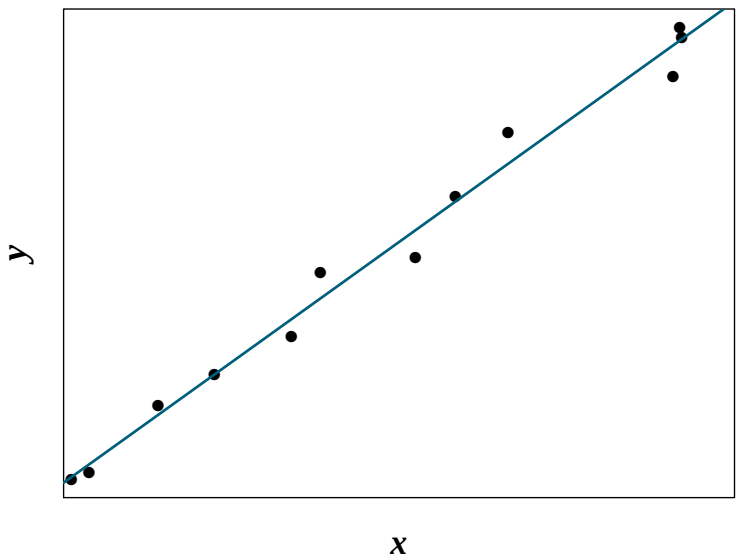
# Overfitting with ERM

Simple artificial data set generator:

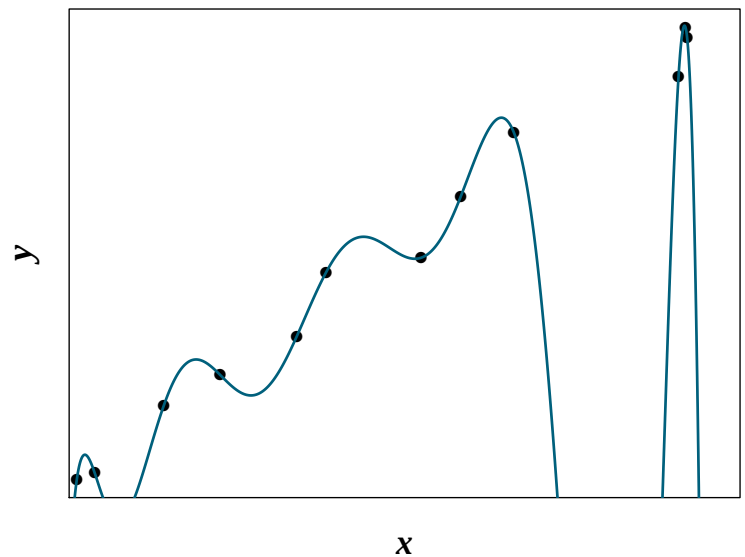
$$x \sim \text{uniform}(0, 1),$$
$$y = x + \epsilon, \qquad \epsilon \sim N(0,\ 0.05).$$




Fitting linear function:




Fitting polynomial of degree $n$:



# Overfitting

- A more complex/flexible function class is not always a good choice.
- Too complex class $\Longrightarrow$ overfitting.
- Too simple class $\Longrightarrow$ learning too constrained model.
- Can we control how much do we overfit?

# Overfitting

- A more complex/flexible function class is not always a good choice.
- Too complex class $\Longrightarrow$ overfitting.
- Too simple class $\Longrightarrow$ learning too constrained model.
- Can we control how much do we overfit?

$\Longrightarrow$ **Vapnik-Chervonenkis theory** (only shown for $0/1$ classification loss)

## Theorem for finite classes ($0/1$ loss) — Occam's Razor[5]

Let the class of functions $\mathcal{H}$ be finite. If the prediction function $\widehat{h}$ was trained on a sample of size $n$ by minimizing the empirical risk within $\mathcal{H}$:

$$\widehat{h} = \arg\min_{h \in \mathcal{H}} \widehat{L}(h)$$

then with probability $1 - \delta$:

$$L(\widehat{h}) - \widehat{L}(\widehat{h}) = O\left(\sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{n}}\right).$$

[5] Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, 1987

## Theorem for finite classes ($0/1$ loss) — Occam's Razor[5]

Let the class of functions $\mathcal{H}$ be finite. If the prediction function $\widehat{h}$ was trained on a sample of size $n$ by minimizing the empirical risk within $\mathcal{H}$:

$$\widehat{h} = \operatorname*{arg\,min}_{h \in \mathcal{H}} \widehat{L}(h)$$

then with probability $1 - \delta$:

$$\underbrace{L(\widehat{h}) - \widehat{L}(\widehat{h})}_{\text{generalization error}} = O\left(\sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{n}}\right).$$

[5] Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, 1987

# Proof [6]

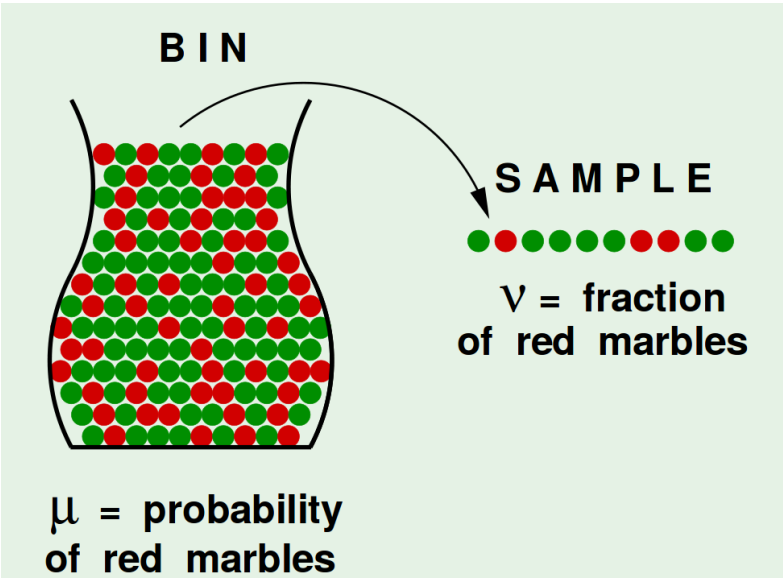
A bin with red and green marbles.

$$P(\text{picking a red marble}) = \mu,$$
$$P(\text{picking a green marble}) = 1 - \mu.$$

The value $\mu$ is unknown.

We pick $n$ marbles **independently**.

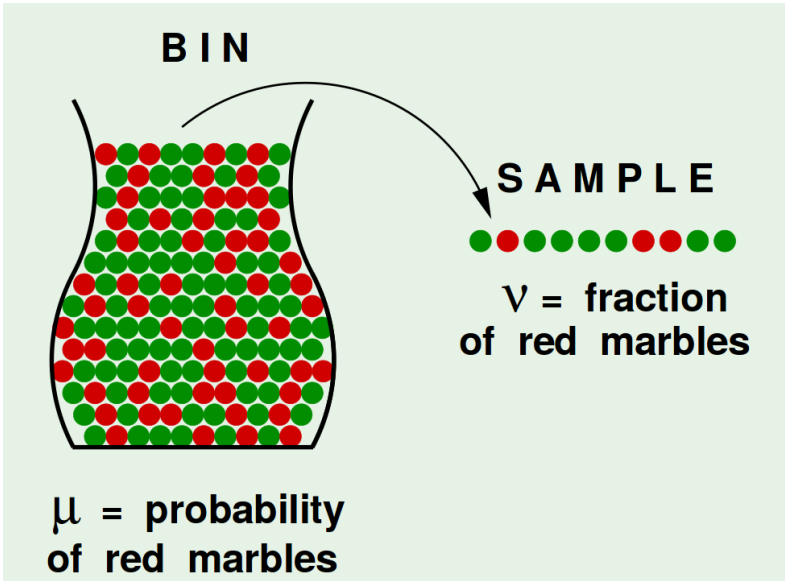
Fraction of red marbles in sample $= \nu$.




[6] Based on: Y. Abu-Mustafa, *Learning from data*, lecture notes at: `https://work.caltech.edu/lectures.html`

# Proof

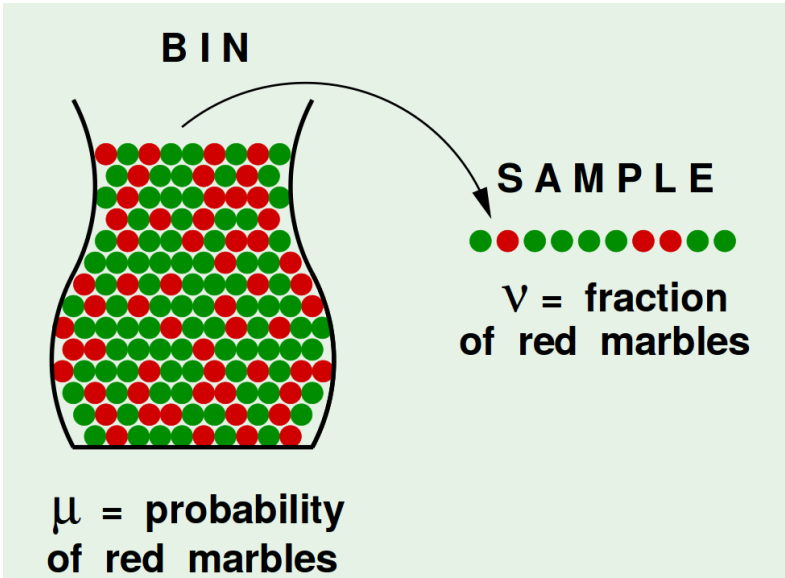
Does $\nu$ say anything about $\mu$?



# Proof

Does $\nu$ say anything about $\mu$?

$\Longrightarrow$ **No:** $\nu$ Sample can be mostly green while bin is mostly red.

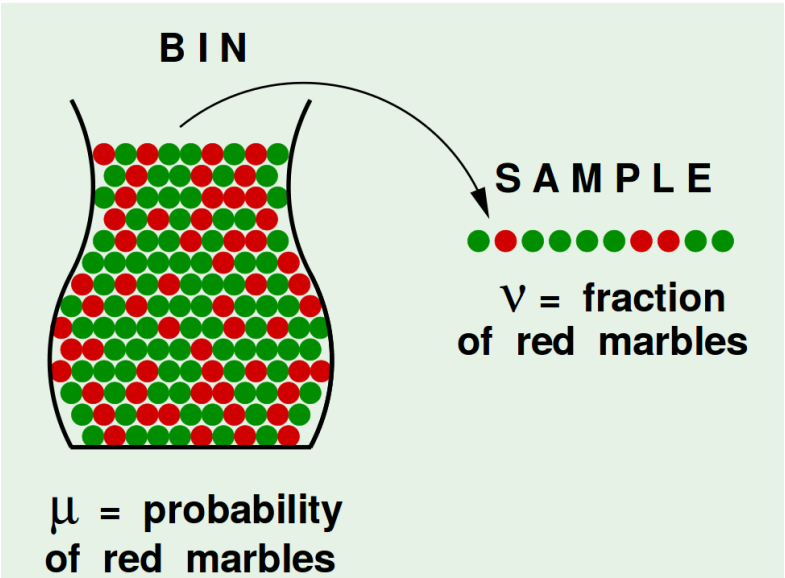


# Proof

Does $\nu$ say anything about $\mu$?

$\Longrightarrow$ **No:** $\nu$ Sample can be mostly green while bin is mostly red.

$\Longrightarrow$ **Yes:** Sample frequency $\nu$ is **likely** to be close to bin frequency $\mu$.



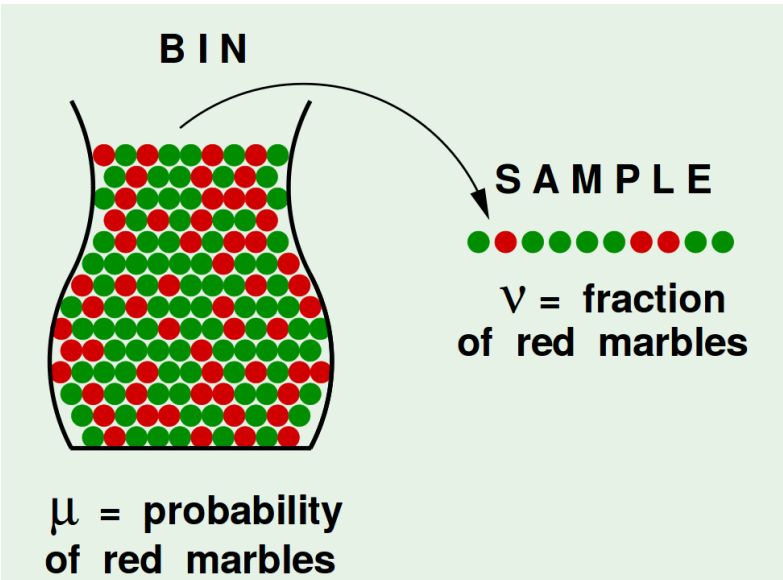
# Proof

Does $\nu$ say anything about $\mu$?

**Hoeffding's inequality:**

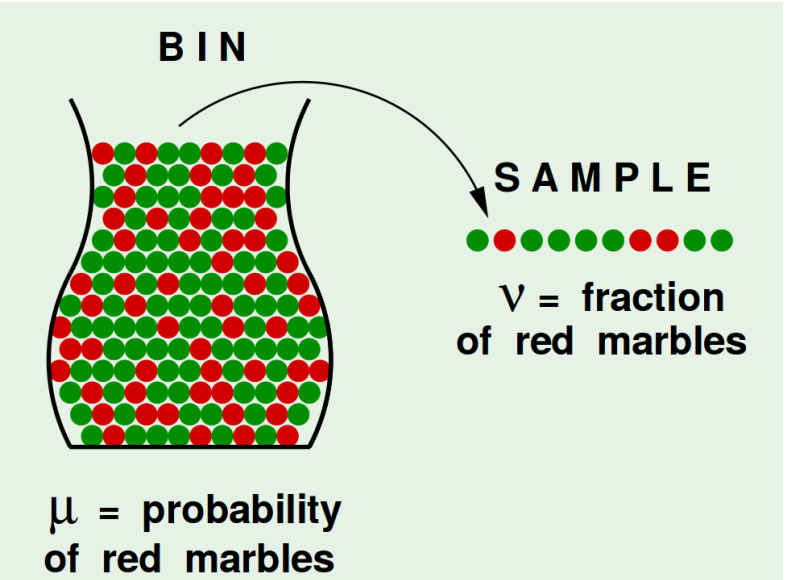
$$P(|\mu - \nu| > \epsilon) \leq 2e^{-2\epsilon^2 n}$$

Probability of large deviation of sample frequence from true probability decreases **exponentially** fast with $n$.



# Proof

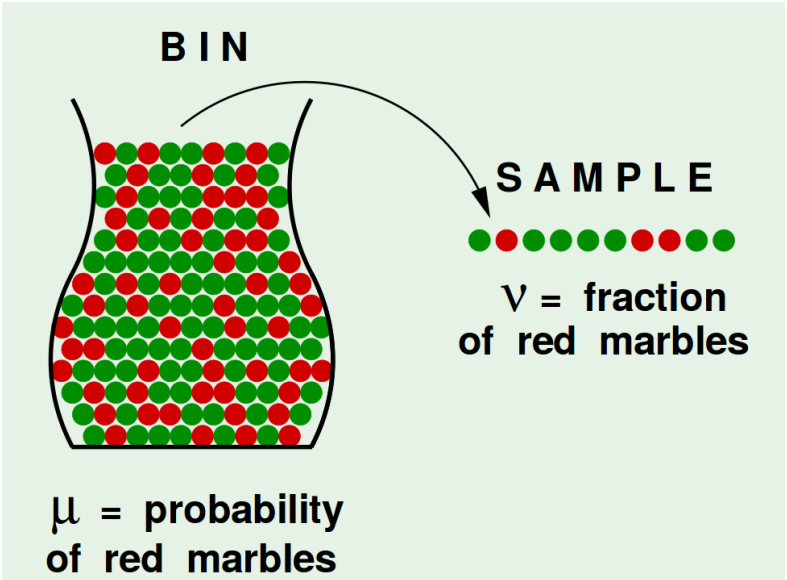
Given a **fixed** classifier $h$:



# Proof

Given a **fixed** classifier $h$:

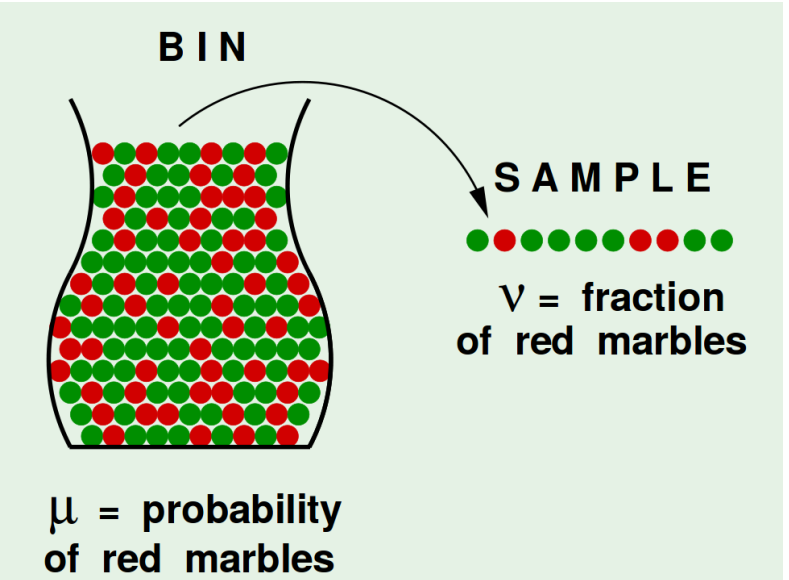
- Each example $(\boldsymbol{x}, y)$ is a marble



# Proof

Given a **fixed** classifier $h$:

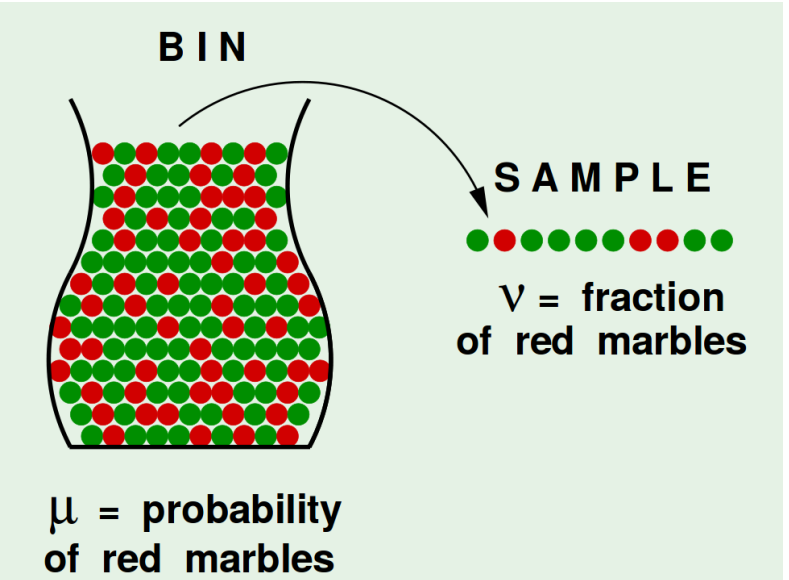
- Each example $(\boldsymbol{x}, y)$ is a marble
- red marble $=$ misclassification $(h(\boldsymbol{x}) \neq y)$
- green marble $=$ correct classification $(h(\boldsymbol{x}) = y)$



# Proof

Given a **fixed** classifier $h$:

- Each example $(\boldsymbol{x}, y)$ is a marble
- red marble $=$ misclassification $(h(\boldsymbol{x}) \neq y)$
- green marble $=$ correct classification $(h(\boldsymbol{x}) = y)$

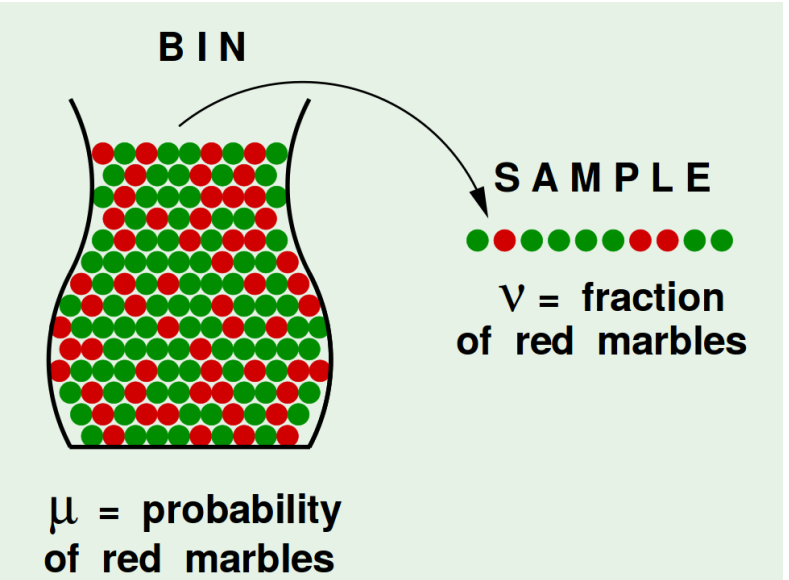



$$\mu = P(\text{picking a red marble}) = P(h \text{ misclassifies } (\boldsymbol{x}, y)) = L(h).$$

# Proof

Given a **fixed** classifier $h$:

- Each example $(\boldsymbol{x}, y)$ is a marble
- red marble $=$ misclassification $(h(\boldsymbol{x}) \neq y)$
- green marble $=$ correct classification $(h(\boldsymbol{x}) = y)$



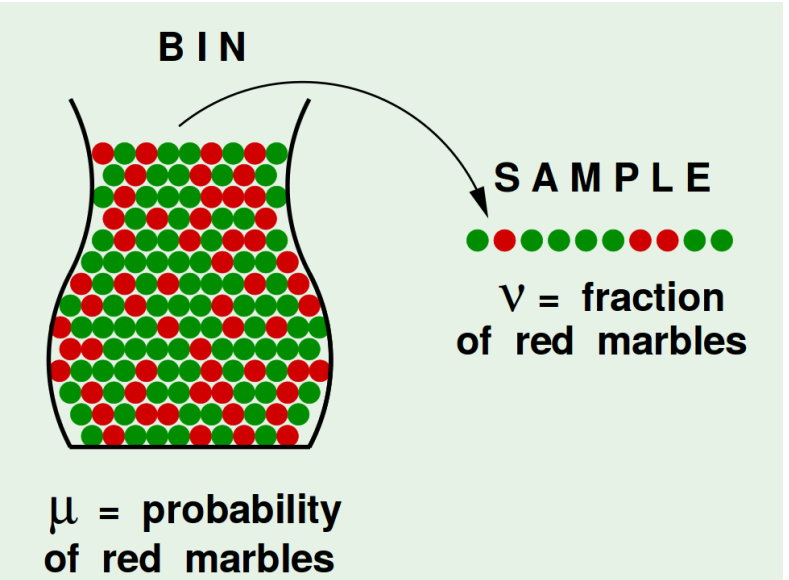

$$\mu = P(\text{picking a red marble}) = P(h \text{ misclassifies } (\boldsymbol{x}, y)) = L(h).$$

$$\nu = \text{fraction of red marbles in a sample} = \widehat{L}(h).$$

# Proof

Given a **fixed** classifier $h$:

- Each example $(\boldsymbol{x}, y)$ is a marble
- red marble $=$ misclassification $(h(\boldsymbol{x}) \neq y)$
- green marble $=$ correct classification $(h(\boldsymbol{x}) = y)$




$$\mu = P(\text{picking a red marble}) = P(h \text{ misclassifies } (\boldsymbol{x}, y)) = L(h).$$

$$\nu = \text{fraction of red marbles in a sample} = \widehat{L}(h).$$

**Hoeffding's inequality**: for any **fixed** classifier $h$,

$$P(|\widehat{L}(h) - L(h)| > \epsilon) \leq 2e^{-2\epsilon^2 n}$$

# Proof

How to handle the fact that $h$ is **not fixed**, but **learns on a sample**?

## Proof

How to handle the fact that $h$ is **not fixed**, but **learns on a sample**?

### Union bound

$$P(A_1 \text{ or } A_2 \text{ or } \ldots \text{ or } A_K) \leq P(A_1) + P(A_2) + \ldots + P(A_K).$$

## Proof

How to handle the fact that $h$ is **not fixed**, but **learns on a sample**?

### Union bound

$$P(A_1 \text{ or } A_2 \text{ or } \ldots \text{ or } A_K) \leq P(A_1) + P(A_2) + \ldots + P(A_K).$$

Given $\mathcal{H} = \{h_1, \ldots, h_K\}$ is finite,

$$P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon)$$

## Proof

How to handle the fact that $h$ is **not fixed**, but **learns on a sample**?

### Union bound

$$P(A_1 \text{ or } A_2 \text{ or } \ldots \text{ or } A_K) \leq P(A_1) + P(A_2) + \ldots + P(A_K).$$

Given $\mathcal{H} = \{h_1, \ldots, h_K\}$ is finite,

$$P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon)$$
$$= P\Big(|\widehat{L}(h_1) - L(h_1)| > \epsilon \text{ or } \ldots \text{ or } |\widehat{L}(h_K) - L(h_K)| > \epsilon\Big)$$

## Proof

How to handle the fact that $h$ is **not fixed**, but **learns on a sample**?

### Union bound

$$P(A_1 \text{ or } A_2 \text{ or } \ldots \text{ or } A_K) \leq P(A_1) + P(A_2) + \ldots + P(A_K).$$

Given $\mathcal{H} = \{h_1, \ldots, h_K\}$ is finite,

$$\begin{aligned} & P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon) \\ & \qquad = P\Big(|\widehat{L}(h_1) - L(h_1)| > \epsilon \text{ or } \ldots \text{ or } |\widehat{L}(h_K) - L(h_K)| > \epsilon\Big) \\ \text{(union bnd)} \quad & \qquad = P(|\widehat{L}(h_1) - L(h_1)| > \epsilon) + \ldots + P(|\widehat{L}(h_K) - L(h_K)| > \epsilon) \end{aligned}$$

## Proof

How to handle the fact that $h$ is **not fixed**, but **learns on a sample**?

### Union bound

$$P(A_1 \text{ or } A_2 \text{ or } \ldots \text{ or } A_K) \leq P(A_1) + P(A_2) + \ldots + P(A_K).$$

Given $\mathcal{H} = \{h_1, \ldots, h_K\}$ is finite,

$$\begin{aligned}
& P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon) \\
& \qquad = P\Big(|\widehat{L}(h_1) - L(h_1)| > \epsilon \text{ or } \ldots \text{ or } |\widehat{L}(h_K) - L(h_K)| > \epsilon\Big) \\
\text{(union bnd)} \quad & \qquad = P(|\widehat{L}(h_1) - L(h_1)| > \epsilon) + \ldots + P(|\widehat{L}(h_K) - L(h_K)| > \epsilon) \\
\text{(Hoeffding)} \quad & \qquad \leq K \times 2e^{-2\epsilon^2 n}
\end{aligned}$$

## Proof

How to handle the fact that $h$ is **not fixed**, but **learns on a sample**?

### Union bound

$$P(A_1 \text{ or } A_2 \text{ or } \ldots \text{ or } A_K) \leq P(A_1) + P(A_2) + \ldots + P(A_K).$$

Given $\mathcal{H} = \{h_1, \ldots, h_K\}$ is finite,

$$\begin{aligned}
& P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon) \\
& \qquad = P\Big(|\widehat{L}(h_1) - L(h_1)| > \epsilon \text{ or } \ldots \text{ or } |\widehat{L}(h_K) - L(h_K)| > \epsilon\Big) \\
\text{(union bnd)} \quad & \qquad = P(|\widehat{L}(h_1) - L(h_1)| > \epsilon) + \ldots + P(|\widehat{L}(h_K) - L(h_K)| > \epsilon) \\
\text{(Hoeffding)} \quad & \qquad \leq K \times 2e^{-2\epsilon^2 n}
\end{aligned}$$

Probability that the training error of **any** classifier from $\mathcal{H}$ deviates by more than $\epsilon$ from its test error is at most $2Ke^{-2\epsilon^2 n}$, where $K = |\mathcal{H}|$.

But:

$$P(\text{train. err. of } \widehat{h} \text{ deviates from test err.})$$
$$\leq \quad P(\text{train. err. of any } h \text{ deviates from test. err.})$$

But:

$$
\begin{aligned}
&P(\text{train. err. of } \widehat{h} \text{ deviates from test err.}) \\
&\qquad\qquad \leq \quad P(\text{train. err. of any } h \text{ deviates from test. err.})
\end{aligned}
$$

$$
\begin{aligned}
P(|\widehat{L}(\widehat{h}) - L(\widehat{h})| > \epsilon) \quad &\leq \quad P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon) \\
&\leq \quad \underbrace{2Ke^{-2\epsilon^2 n}}_{=\delta}.
\end{aligned}
$$

But:

$$P(\text{train. err. of } \widehat{h} \text{ deviates from test err.}) \leq P(\text{train. err. of any } h \text{ deviates from test. err.})$$

$$\begin{aligned} P(|\widehat{L}(\widehat{h}) - L(\widehat{h})| > \epsilon) &\leq P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon) \\ &\leq \underbrace{2Ke^{-2\epsilon^2 n}}_{=\delta}. \end{aligned}$$

$$\delta = 2Ke^{-2\epsilon^2 n} \qquad \Longleftrightarrow \qquad \epsilon = \sqrt{\frac{\log(K) + \log(2/\delta)}{2n}}$$

But:

$$P(\text{train. err. of } \widehat{h} \text{ deviates from test err.}) \leq P(\text{train. err. of any } h \text{ deviates from test. err.})$$

$$\begin{aligned} P(|\widehat{L}(\widehat{h}) - L(\widehat{h})| > \epsilon) &\leq P(\exists h \in \mathcal{H} \colon |\widehat{L}(h) - L(h)| > \epsilon) \\ &\leq \underbrace{2Ke^{-2\epsilon^2 n}}_{=\delta}. \end{aligned}$$

$$\delta = 2Ke^{-2\epsilon^2 n} \qquad \Longleftrightarrow \qquad \epsilon = \sqrt{\frac{\log(K) + \log(2/\delta)}{2n}}$$

**Conclusion:** With probability at most $1 - \delta$:

$$L(\widehat{h}) - \widehat{L}(\widehat{h}) \leq \epsilon = \sqrt{\frac{\log(|\mathcal{H}|) + \log(2/\delta)}{2n}}.$$

# Generalization bounds

$$L(\widehat{h}) = \widehat{L}(\widehat{h}) + O\left(\sqrt{\frac{\log|\mathcal{H}| + \log(1/\delta)}{n}}\right).$$

- Overhead grows very **slowly** (logarithmically) with size of $|\mathcal{H}|$.
- Overhead decreases as $O(1/\sqrt{n})$ with the sample size.
- Impractical: does not work for infinite classes of functions.

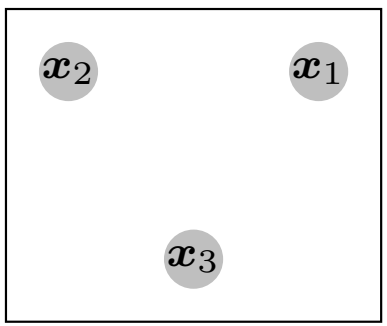
# Vapnik-Chervonenkis (VC) dimension

A function class $\mathcal{H}$ can **shatter** a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ if for any set of $0/1$ labels $\{y_1, \ldots, y_n\}$ there exists $h \in \mathcal{H}$ which gets training error $0$.

# Vapnik-Chervonenkis (VC) dimension

A function class $\mathcal{H}$ can **shatter** a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ if for any set of $0/1$ labels $\{y_1, \ldots, y_n\}$ there exists $h \in \mathcal{H}$ which gets training error $0$.

## Example

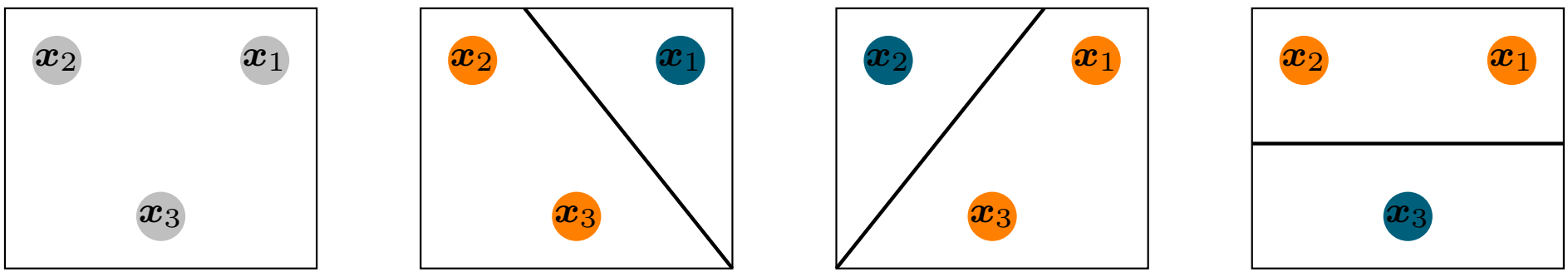



Class of linear functions in $d = 2$ can shatter any three points in general positions.

# Vapnik-Chervonenkis (VC) dimension

A function class $\mathcal{H}$ can **shatter** a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ if for any set of $0/1$ labels $\{y_1, \ldots, y_n\}$ there exists $h \in \mathcal{H}$ which gets training error $0$.

## Example




Class of linear functions in $d = 2$ can shatter any three points in general positions.

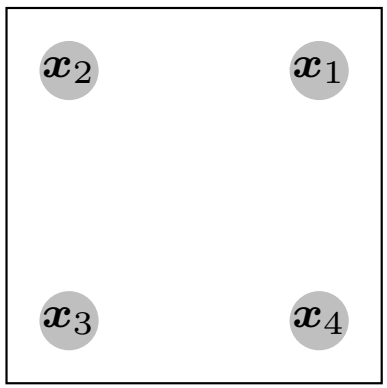
## Vapnik-Chervonenkis (VC) dimension

The largest $n$, such that there exists a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ which can be shattered by $\mathcal{H}$ is called **VC dimension** of $\mathcal{H}$.

# Vapnik-Chervonenkis (VC) dimension

The largest $n$, such that there exists a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ which can be shattered by $\mathcal{H}$ is called **VC dimension** of $\mathcal{H}$.

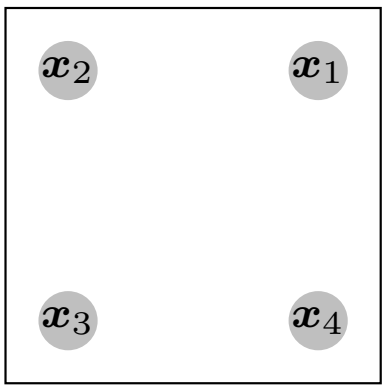
**Example** Class of linear functions in $d = 3$ has VC dimension $3$.

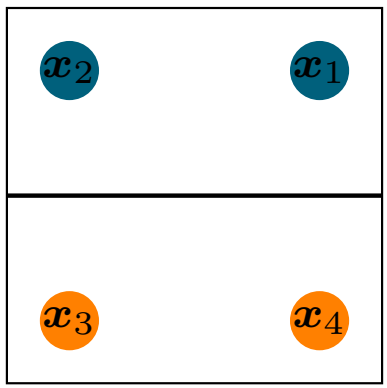


# Vapnik-Chervonenkis (VC) dimension

The largest $n$, such that there exists a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ which can be shattered by $\mathcal{H}$ is called **VC dimension** of $\mathcal{H}$.

**Example** Class of linear functions in $d = 3$ has VC dimension $3$.

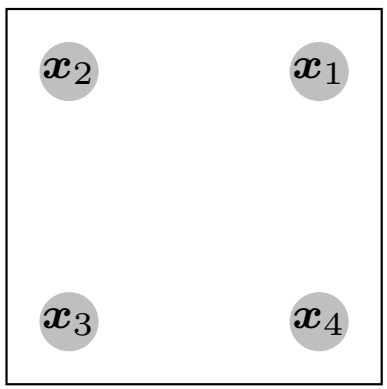


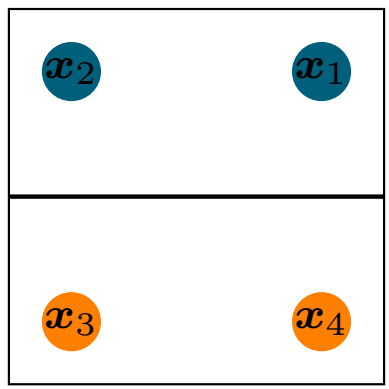



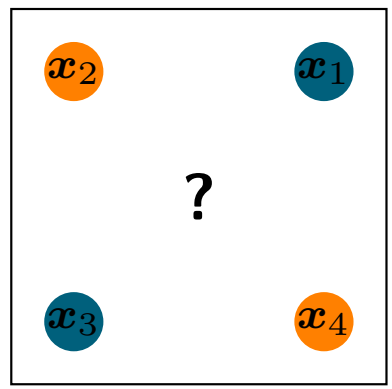
# Vapnik-Chervonenkis (VC) dimension

The largest $n$, such that there exists a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ which can be shattered by $\mathcal{H}$ is called **VC dimension** of $\mathcal{H}$.

**Example** Class of linear functions in $d = 3$ has VC dimension $3$.



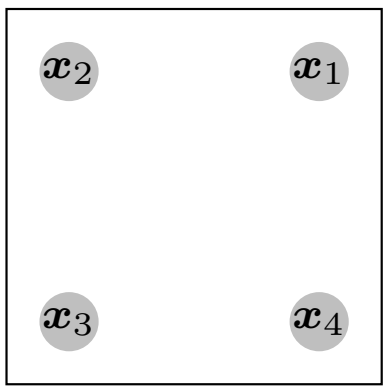



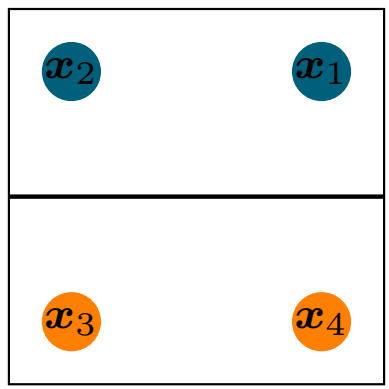



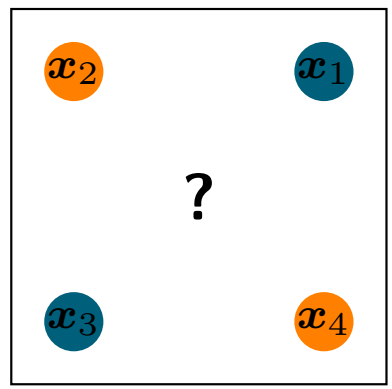
# Vapnik-Chervonenkis (VC) dimension

The largest $n$, such that there exists a set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ which can be shattered by $\mathcal{H}$ is called **VC dimension** of $\mathcal{H}$.

**Example** Class of linear functions in $d = 3$ has VC dimension $3$.




Generally, class of linear functions has VC dimension $d + 1$.

# Vapnik-Chervonenkis (VC) dimension

What is VC dimension of:

- Circles centered at origin?
- Rectangles on a plane?
- Sine functions in one dimension?

# Vapnik-Chervonenkis (VC) dimension

What is VC dimension of:

- Circles centered at origin? **VC-dim** $= 2$
- Rectangles on a plane?
- Sine functions in one dimension?

# Vapnik-Chervonenkis (VC) dimension

What is VC dimension of:

- Circles centered at origin? **VC-dim** $= 2$
- Rectangles on a plane? **VC-dim** $= 4$
- Sine functions in one dimension?

# Vapnik-Chervonenkis (VC) dimension

What is VC dimension of:

- Circles centered at origin? **VC-dim** $= 2$
- Rectangles on a plane? **VC-dim** $= 4$
- Sine functions in one dimension? **VC-dim** $= \infty$

# Vapnik-Chervonenkis (VC) dimension

What is VC dimension of:

- Circles centered at origin? **VC-dim** $= 2$
- Rectangles on a plane? **VC-dim** $= 4$
- Sine functions in one dimension? **VC-dim** $= \infty$

In general, **VC-dim** $\neq$ **num. of parameters**!

VC dimension measures the flexibility of a class of functions for classification purposes.

## Theorem for VC classes ($0/1$ loss)[7]

Let the class of functions $\mathcal{H}$ has VC dimension $d_{\mathrm{VC}}$. If the prediction function $\widehat{h}$ was trained on a sample of size $n$ by minimizing the empirical risk within $\mathcal{H}$:

$$\widehat{h} = \arg\min_{h \in \mathcal{H}} \widehat{L}(h)$$

then with probability $1 - \delta$:

$$\underbrace{L(\widehat{h}) - \widehat{L}(\widehat{h})}_{\text{generalization error}} = O\left(\sqrt{\frac{d_{\mathrm{VC}} + \log(1/\delta)}{n}}\right).$$

[7] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971

# Generalization bounds – summary

- A theoretical framework behind machine learning which helps us to understand why and how learning algorithms work.
- Quantification of overfitting.
- The bounds should only be used qualitatively, not quantitatively! (they account for the worst case and are loose in general).
- Much more than what was presented here:
  - tighter bounds in special cases,
  - data dependent bounds (margin bounds, PAC-Bayes bounds),
  - other loss functions (regression, convex surrogates),
  - etc.

# Estimation and approximation error

## Estimation and approximation error

- If we learn $h$ within class $\mathcal{H}$, we can only hope for getting $h$ close to the **best prediction function within $\mathcal{H}$**:

$$h^*_{\mathcal{H}} = \arg\min_{h \in \mathcal{H}} L(h)$$

## Estimation and approximation error

- If we learn $h$ within class $\mathcal{H}$, we can only hope for getting $h$ close to the **best prediction function within $\mathcal{H}$**:

$$h^*_{\mathcal{H}} = \arg\min_{h \in \mathcal{H}} L(h)$$

compare with $h^* = \arg\min_h L(h)$

# Estimation and approximation error

- If we learn $h$ within class $\mathcal{H}$, we can only hope for getting $h$ close to the **best prediction function within $\mathcal{H}$**:

$$h^*_{\mathcal{H}} = \arg\min_{h \in \mathcal{H}} L(h)$$

- Decomposition

compare with $h^* = \arg\min_h L(h)$

$$\begin{aligned} \mathrm{Reg}(h) &= L(h) - L(h^*) \\ &= \underbrace{L(h) - L(h^*_{\mathcal{H}})}_{\text{estimation error}} + \underbrace{L(h^*_{\mathcal{H}}) - L(h^*)}_{\text{approximation error}} \end{aligned}$$

# Estimation and approximation error

- If we learn $h$ within class $\mathcal{H}$, we can only hope for getting $h$ close to the **best prediction function within $\mathcal{H}$**:

$$h^*_{\mathcal{H}} = \arg\min_{h \in \mathcal{H}} L(h)$$

- Decomposition

compare with $h^* = \arg\min_h L(h)$

$$\begin{aligned} \mathrm{Reg}(h) &= L(h) - L(h^*) \\ &= \underbrace{L(h) - L(h^*_{\mathcal{H}})}_{\text{estimation error}} + \underbrace{L(h^*_{\mathcal{H}}) - L(h^*)}_{\text{approximation error}} \end{aligned}$$

- Only the estimation error depends on the training data.
- The approximation error depends on $\mathcal{H}$ and $P(\boldsymbol{x}, y)$.

# Estimation and approximation error

- If we learn $h$ within class $\mathcal{H}$, we can only hope for getting $h$ close to the **best prediction function within $\mathcal{H}$**:

$$h^*_{\mathcal{H}} = \arg\min_{h \in \mathcal{H}} L(h)$$

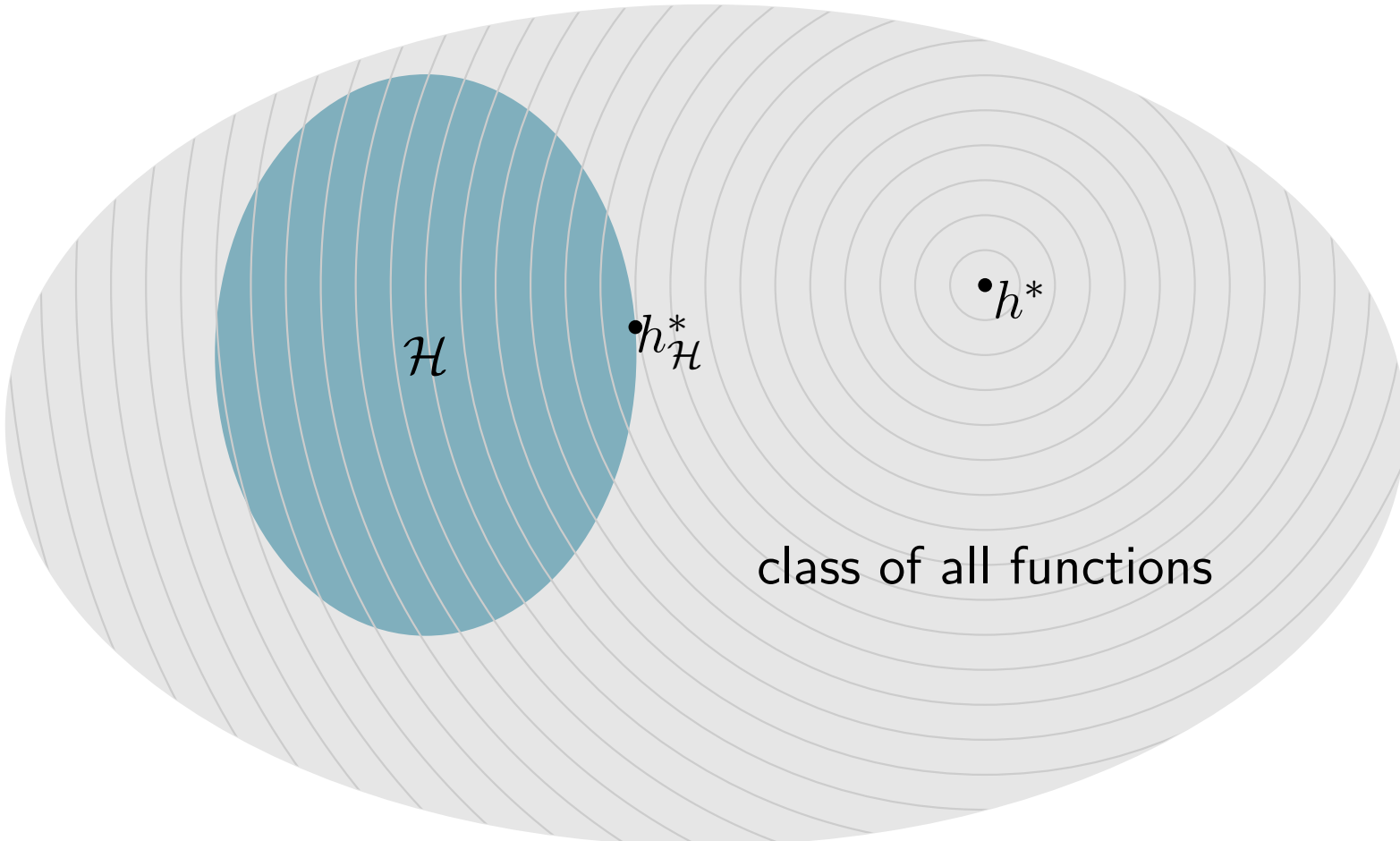
- Decomposition

compare with $h^* = \arg\min_h L(h)$

$$\begin{aligned} \mathrm{Reg}(h) &= L(h) - L(h^*) \\ &= \underbrace{L(h) - L(h^*_{\mathcal{H}})}_{\text{estimation error}} + \underbrace{L(h^*_{\mathcal{H}}) - L(h^*)}_{\text{approximation error}} \end{aligned}$$

- Only the estimation error depends on the training data.
- The approximation error depends on $\mathcal{H}$ and $P(\boldsymbol{x}, y)$.
- A version of this decomposition: **bias-variance decomposition**.

# Estimation and approximation error



# Estimation and approximation error