Classification and Regression III

Krzysztof Dembczyński

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

Software Development Technologies
Master studies, second semester
Academic year 2018/19 (winter course)

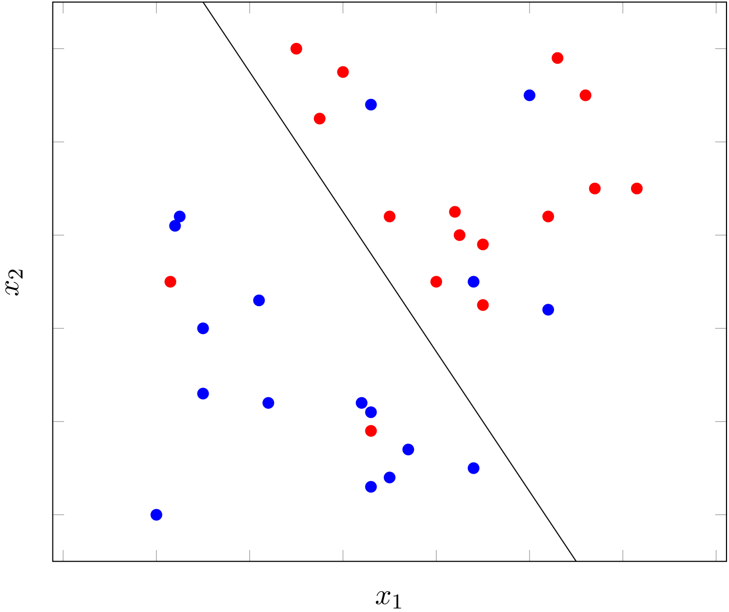
Review of the previous lectures

- Mining of massive datasets.
- Classification and regression
  - What is machine learning?
  - Supervised learning: statistical decision/learning theory, loss functions, risk.
  - Learning paradigms and principles.
  - Learning algorithms: lazy learning, decision trees, generative models, linear models.

Outline

Outline

Linear models for classification



- Let the output variable be $y \in \{-1, 1\}$ and prediction function $h(\boldsymbol{x}) \in \{-1, 1\}$.

Linear models for classification

- Let the output variable be $y \in \{-1, 1\}$ and prediction function $h(\boldsymbol{x}) \in \{-1, 1\}$.

- Alternatively, one can assume $y, h(\boldsymbol{x}) \in \{0, 1\}$.
- Mapping $m : \{-1, 1\} \longrightarrow \{0, 1\}$ and $m^{-1} : \{0, 1\} \longrightarrow \{-1, 1\}$:

$$m(y) = \frac{y+1}{2}\,, \qquad m^{-1}(y) = 2y - 1\,.$$

- Loss is measured usually in terms of 0/1 loss which can be expressed by:

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

- Loss is measured usually in terms of 0/1 loss which can be expressed by:
$$\ell_{0/1}(y, h(\boldsymbol{x})) = [\![yh(\boldsymbol{x}) \le 0]\!]$$
- Solve:
$$\widehat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell_{0/1}(y_i, h(\boldsymbol{x}_i)) = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} [\![y_i h(\boldsymbol{x}_i) \le 0]\!]$$

Linear models for classification

- Loss is measured usually in terms of 0/1 loss which can be expressed by:

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

- Solve:

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

- **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** $f \in \mathcal{F}$:

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

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

- 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

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

  - The quantity $y f(\boldsymbol{x})$ is usually referred to as **margin**.

- Solve:

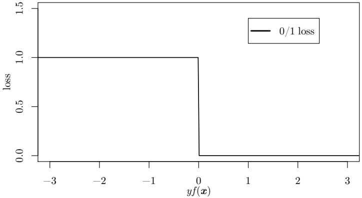
$$\widehat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [\![y_i f(\boldsymbol{x}_i) \leq 0]\!]$$

Linear models for classification

- Solve:

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

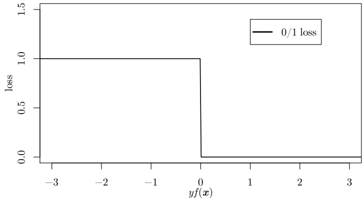
- Still **hard** to optimize: $0/1$ loss is **discontinuous** and **non-convex**.
  - e.g., when $\mathcal{H}$ is a class of linear function, the problem known to be **NP-hard**.



- Solve:

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

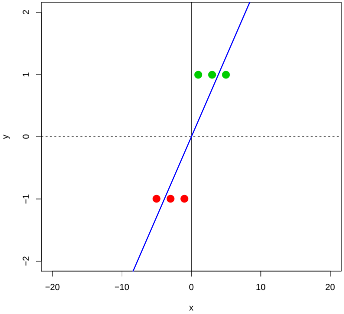
- Still **hard** to optimize: $0/1$ loss is **discontinuous** and **non-convex**.
  - e.g., when $\mathcal{H}$ is a class of linear function, the problem known to be **NP-hard**.



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

Linear regression for classification

- We can try to use linear regression to solve binary classification problem:



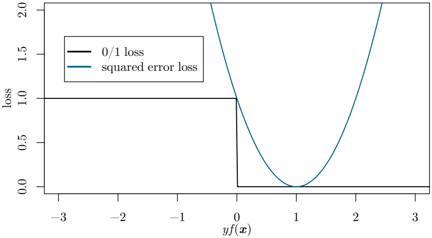
- Use $\operatorname{sgn}\left(\widehat{f}\right)$ to obtain prediction.
- Minimization of squared loss leads to estimation of the conditional probability, since:

$$P(y = 1|\boldsymbol{x}) = \frac{\mathbb{E}(y|\boldsymbol{x}) + 1}{2}\,.$$

Linear regression for classification

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

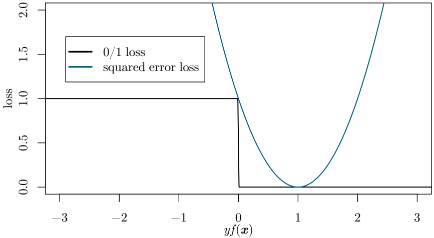
$$\ell_{\text{sq}}(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2 = (1 - yf(\boldsymbol{x}))^2.$$



Linear regression for classification

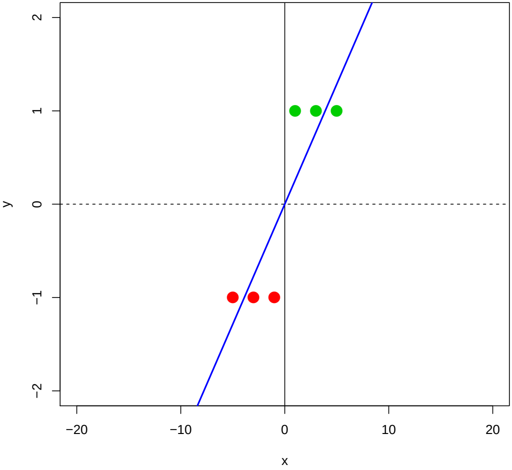
- Effectively, we replace 0/1 loss by **squared error loss**:

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

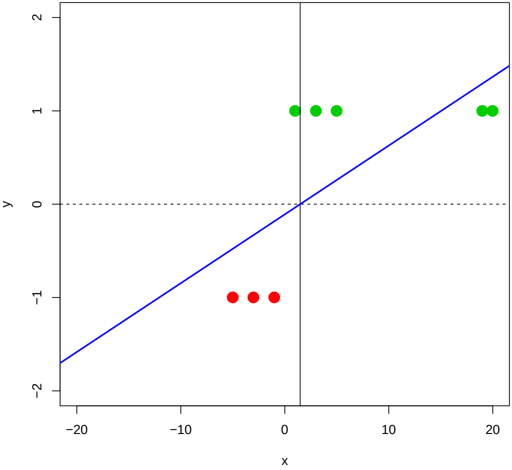


- Works nicely in practice, but has several drawbacks ...

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

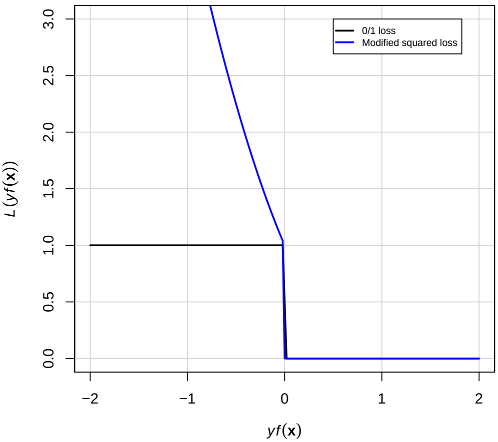
- We could consider non-zero loss only for examples with margin less or equal zero:

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

- Such a loss function could be defined as:

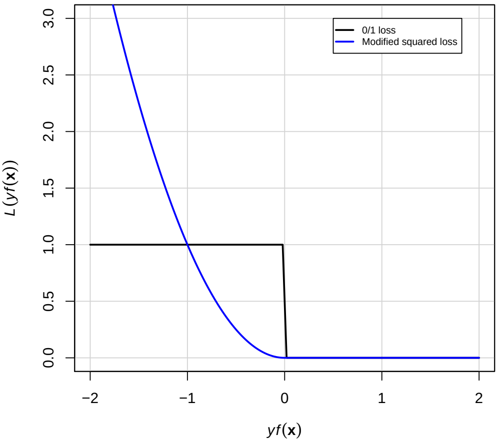
$$\ell(y, f(\boldsymbol{x})) = \begin{cases} 0, & \text{if } yf(\boldsymbol{x}) > 0 \\ (1 - yf(\boldsymbol{x}))^2, & \text{otherwise}. \end{cases}$$

Linear models for classification



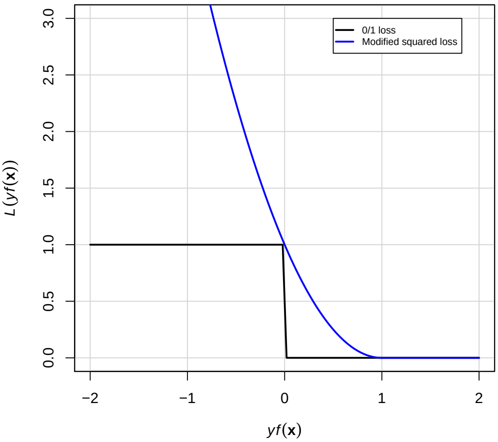
- This definition does not lead to a “nice” shape of the loss.

Linear models for classification



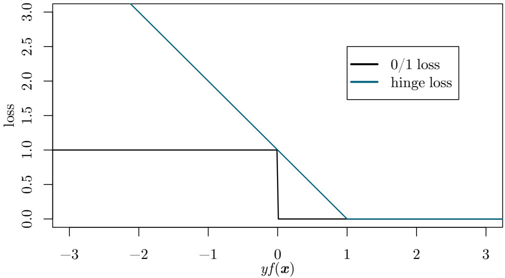
- A better solution: $\ell(y, f(\boldsymbol{x})) = (\max\{0, \epsilon - yf(\boldsymbol{x})\})^2$.

Linear models for classification



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

- 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 an incremental optimization technique ($\Rightarrow$ the stochastic gradient descent algorithm).

Perceptron

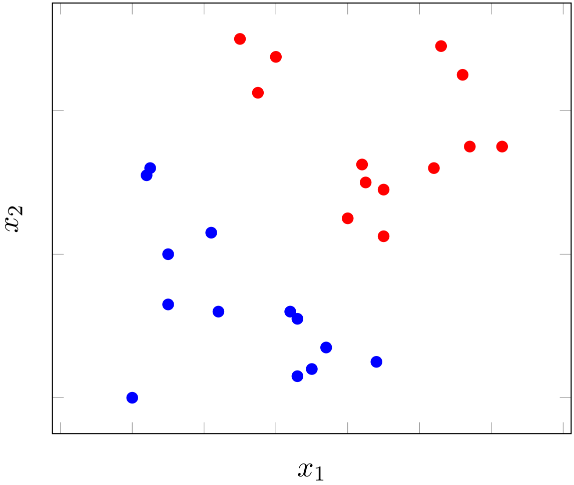
- Learning algorithm for perceptron:
  - The update in iteration t:

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

    where α is the learning rate.
  - Only misclassified examples are updated.

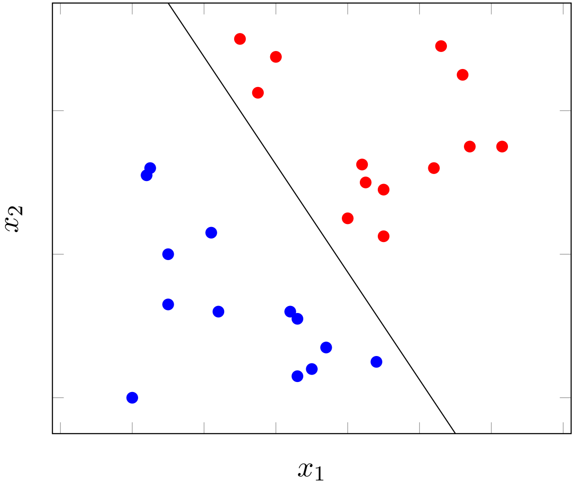
Perceptron – Graphical interpretation

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



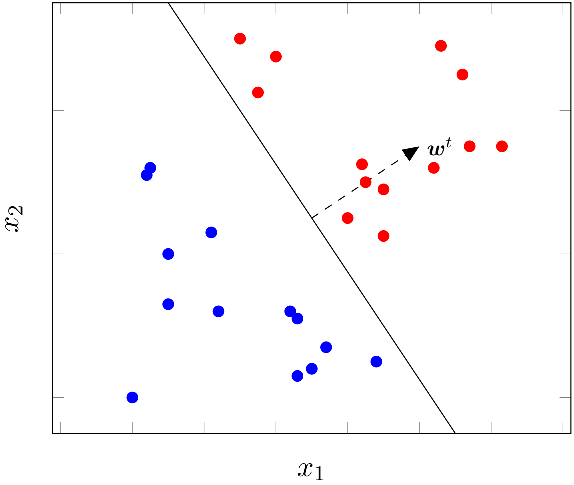
Perceptron – Graphical interpretation

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



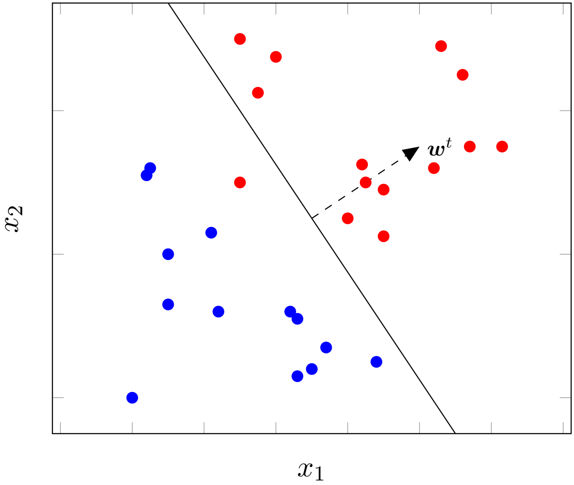
Perceptron – Graphical interpretation

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



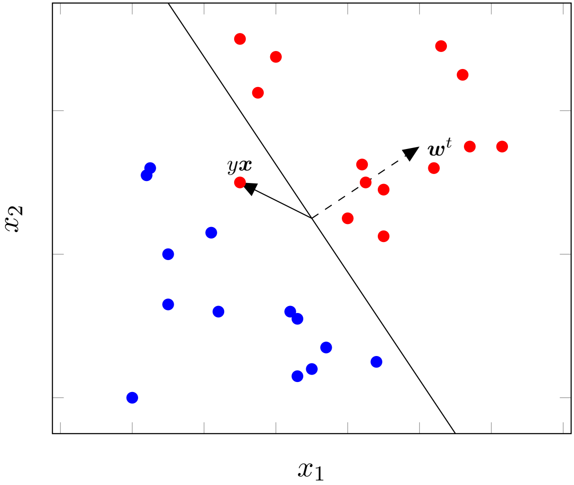
Perceptron – Graphical interpretation

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



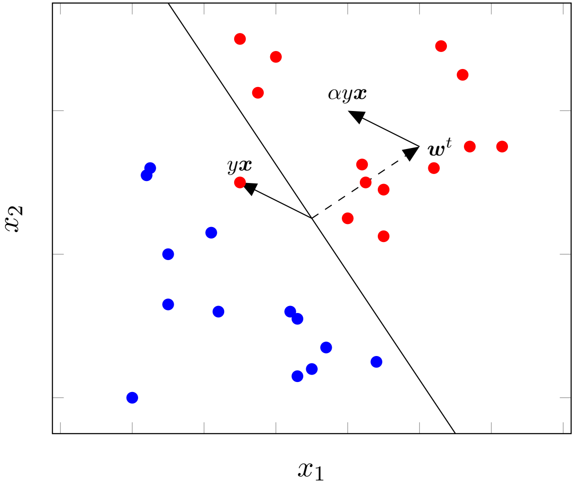
Perceptron – Graphical interpretation

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



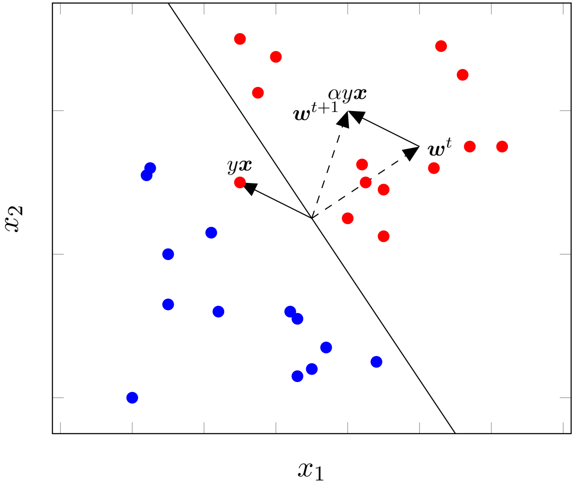
Perceptron – Graphical interpretation

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



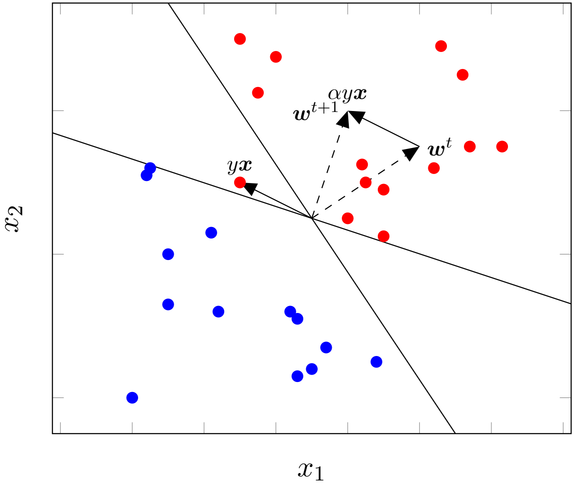
Perceptron – Graphical interpretation

- 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:



Perceptron

- The update can be interpreted in terms of gradient descent:
  - 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

- The update can be interpreted in terms of gradient descent:
  - 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:

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

- Such an algorithm is usually referred to as **stochastic gradient descent**.

Stochastic gradient descent

```
Input: learning rate α
w = 0; //(or use random values)
while (approximate minimum is obtained) {
    Randomly shuffle examples in the training set
    for i=1 to n {
        w := w − α ∂ℓ(y_i, f(x_i))/∂w
    }
}
```

$$\boldsymbol{w} := \boldsymbol{w} - \alpha \frac{\partial \ell(y_i, f(\boldsymbol{x}_i))}{\partial \boldsymbol{w}}$$

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:

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

- 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:

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

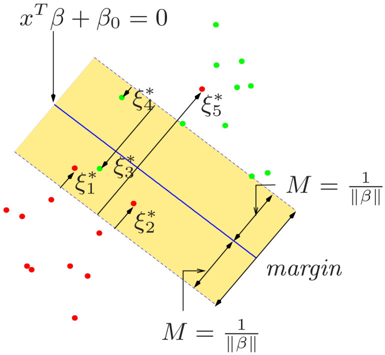
Binary classification by Support Vector Machines (SVM)

Find **maximal margin classifier**

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

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

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



Binary classification by Support Vector Machines (SVM)

Find **maximal margin classifier**

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

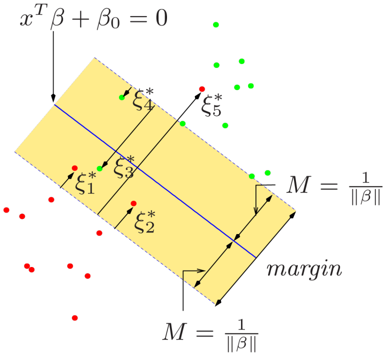
$$\text{s.t.} \quad y_i \boldsymbol{w}\cdot\boldsymbol{x} > 1-\xi_i \qquad \forall i = 1..n$$

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

$$\Updownarrow$$

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

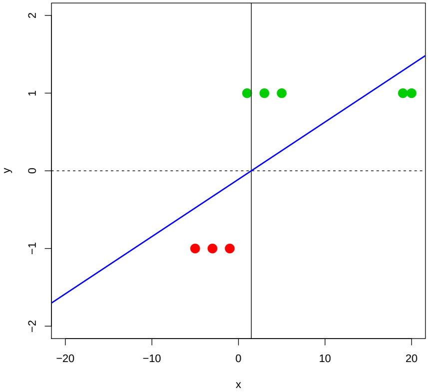
$\text{s.t.} \quad \|\boldsymbol{w}\|^2 \leq B$ for some B.



Logistic regression – motivation

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

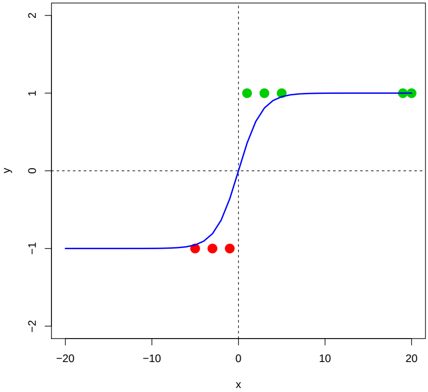
$$g(\boldsymbol{x}) = \frac{1}{1+\exp(-f(\boldsymbol{x}))} \in (0,1) \quad g(\boldsymbol{x}) = \frac{1-\exp(-f(\boldsymbol{x}))}{1+\exp(-f(\boldsymbol{x}))} \in (-1,1)$$



Logistic regression – motivation

- 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 $f(\boldsymbol{x}) \in \mathbb{R}$.

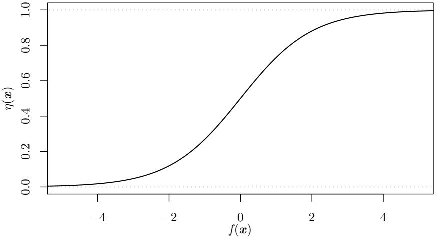
Logistic regression – motivation

- 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**

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

$$\Updownarrow$$

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



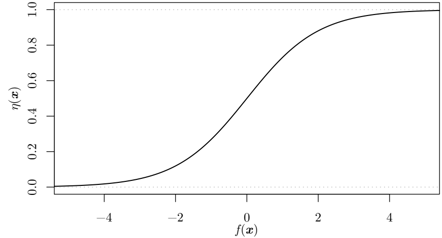
Logistic regression – motivation

- 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**

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

$$\Updownarrow$$

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



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

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

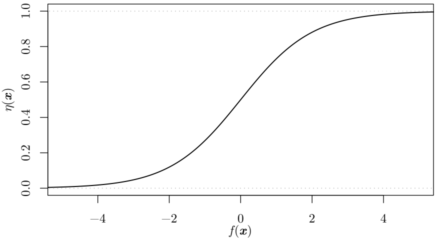
Logistic regression – motivation

- Get an estimate of $\eta(\boldsymbol{x}) = P(y = 1 \mid \boldsymbol{x})$ 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**.

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

Why this is correct?

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

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

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

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

Logistic regression

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

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

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

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

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

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

Logistic regression

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

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

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

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

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

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

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

Logistic regression

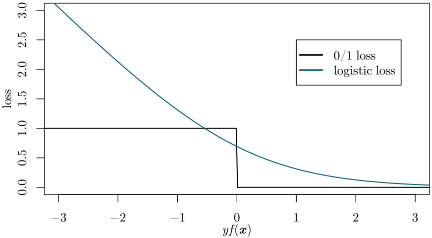
$$\begin{aligned}
\widehat{f} &= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right) \\
&= \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + e^{-y_i f(\boldsymbol{x}_i)}\right).
\end{aligned}$$

Logistic regression

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

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

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

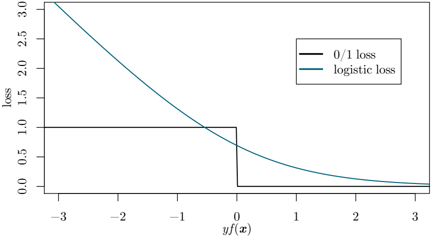


Logistic regression

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

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

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



- Commonly used, better than least squares in practice.

Learning algorithm for logistic regression

- Let $f(\boldsymbol{x})$ be a linear function of 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.

Learning algorithm for 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 α is the step size (learning rate) and

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

Learning algorithm for logistic regression

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

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

Learning algorithm for logistic regression

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

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

Learning algorithm for logistic regression

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

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

Learning algorithm for logistic regression

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

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

Learning algorithm for logistic regression

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

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

Learning algorithm for logistic regression

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

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

Learning algorithm for logistic regression

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

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

- Denote:

$$\widehat{\eta}_i = \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x}_i)}$$

Learning algorithm for logistic regression

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

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

- Denote:

$$\widehat{\eta}_i = \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x}_i)}$$

- Then:

Learning algorithm for logistic regression

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

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

- Denote:

$$\widehat{\eta}_i = \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x}_i)}$$

- Then:

$$\frac{\partial L}{\partial w_j} = -\sum_{y_i=1} (1 - \widehat{\eta}_i) x_{ij} + \sum_{y_i=-1} \widehat{\eta}_i x_{ij} =$$

Learning algorithm for logistic regression

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

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

- Denote:

$$\widehat{\eta}_i = \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x}_i)}$$

- Then:

$$\frac{\partial L}{\partial w_j} = -\sum_{y_i=1} (1 - \widehat{\eta}_i) x_{ij} + \sum_{y_i=-1} \widehat{\eta}_i x_{ij} = -\sum_{i=1}^{n} (y'_i - \widehat{\eta}_i) x_{ij} \,,$$

where $y' = \frac{y+1}{2} \in \{0, 1\}$.

Learning algorithm for logistic regression

- Connection with linear regression:
  - The partial derivative of the squared loss over training examples with respect to w_j is similar:

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

Learning algorithm for logistic regression

- Connection with linear regression:
  - The partial derivative of the squared loss over training examples with respect to w_j is similar:

$$\frac{\partial \widehat{L}_{se}}{\partial w_j} = -2\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}))x_{ij}$$

Learning algorithm for logistic regression

- Connection with linear regression:
  - The partial derivative of the squared loss over training examples with respect to w_j is similar:

$$\frac{\partial \widehat{L}_{se}}{\partial w_j} = -2\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}))x_{ij}$$

  - For the squared error loss, however, the solution can be found analytically since $f(\boldsymbol{x})$ is linear.

Learning algorithm for logistic regression

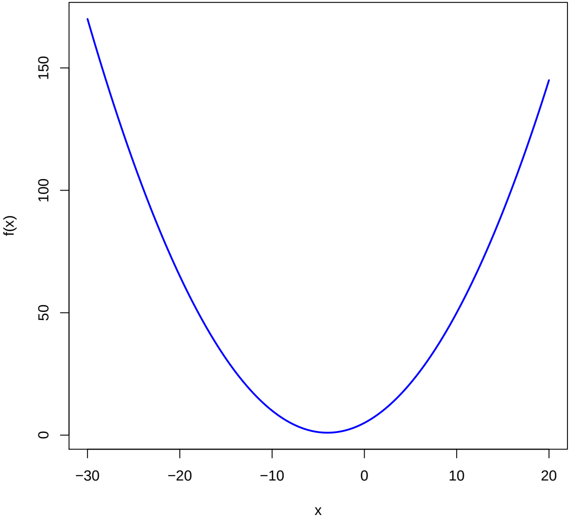
- Connection with linear regression:
  - The partial derivative of the squared loss over training examples with respect to w_j is similar:

$$\frac{\partial \widehat{L}_{se}}{\partial w_j} = -2 \sum_{i=1}^{n} (y_i - f(\boldsymbol{x})) x_{ij}$$

  - For the squared error loss, however, the solution can be found analytically since $f(\boldsymbol{x})$ is linear.
  - For logistic regression we need to use iterative methods, since $\widehat{\eta}_i$ is not linear, but sigmoid.
- The simplest method assumes α to be constant.
- It does not mean that the step is always the same:
  - As we approach (local) minimum, $\frac{\partial \widehat{L}(\boldsymbol{w})}{\partial \boldsymbol{w}}$ takes smaller values.

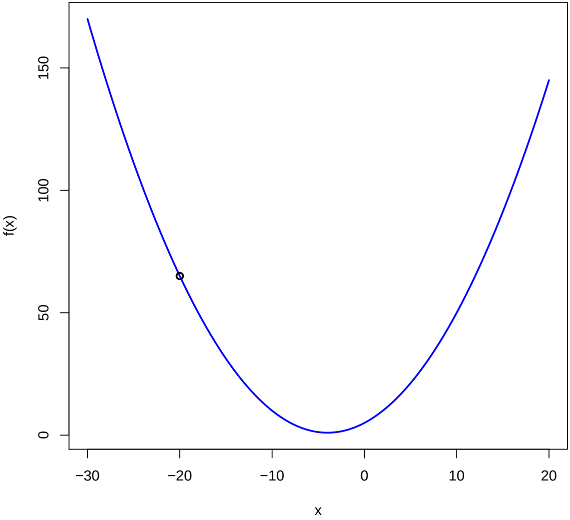
Logistic regression

- Gradient descent example:



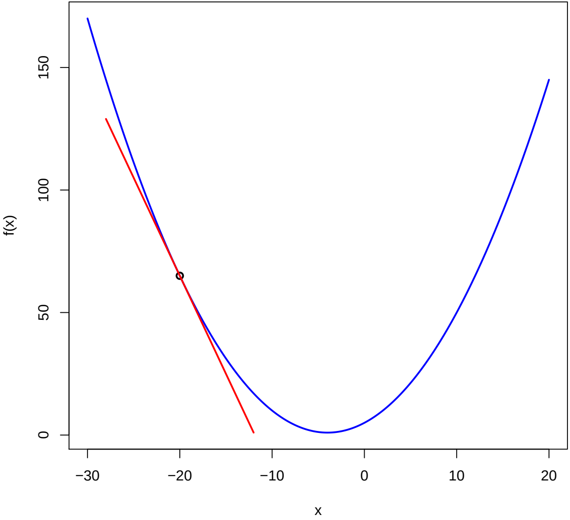
Logistic regression

- Gradient descent example:



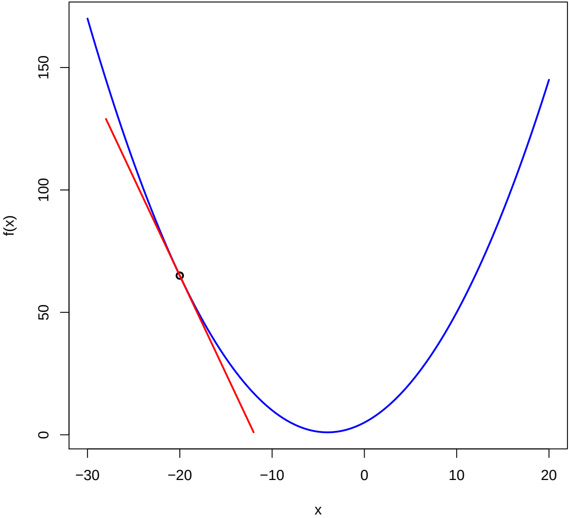
Logistic regression

- Gradient descent example:



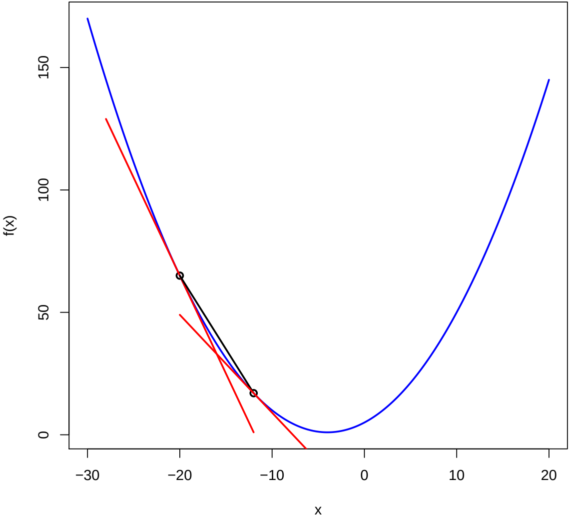
Logistic regression

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



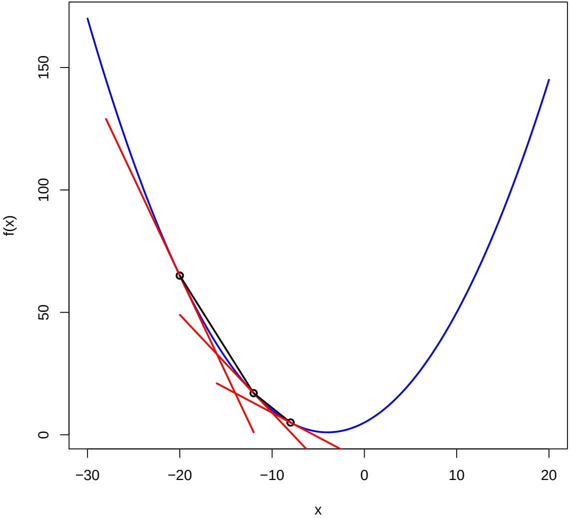
Logistic regression

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



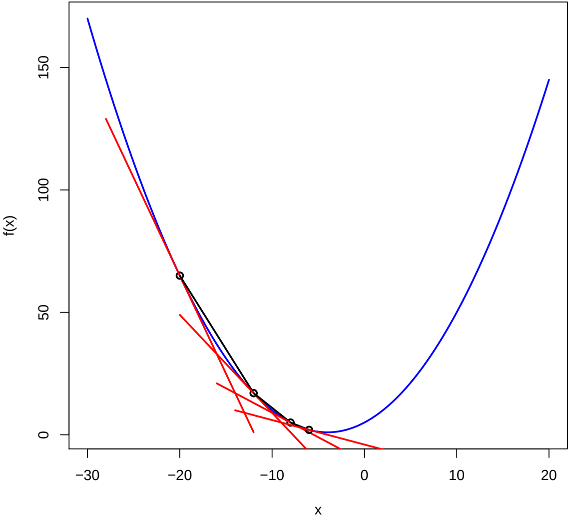
Logistic regression

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



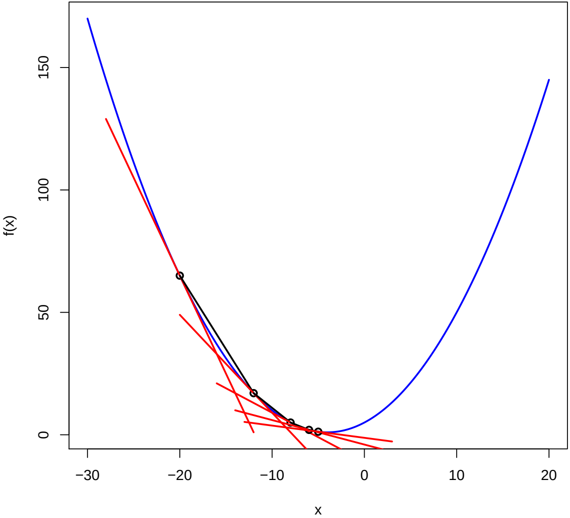
Logistic regression

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



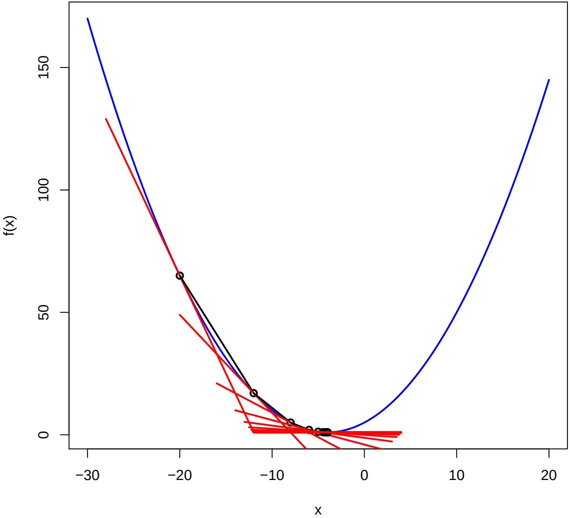
Logistic regression

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



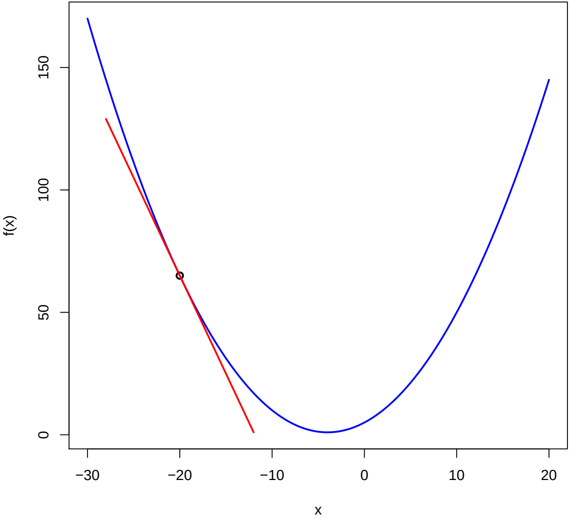
Logistic regression

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



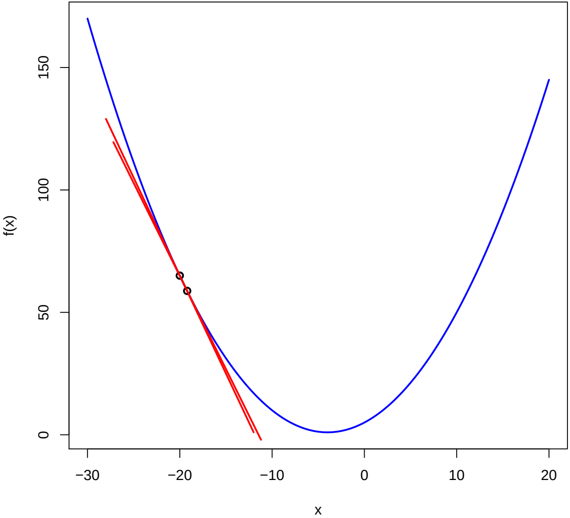
Logistic regression

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



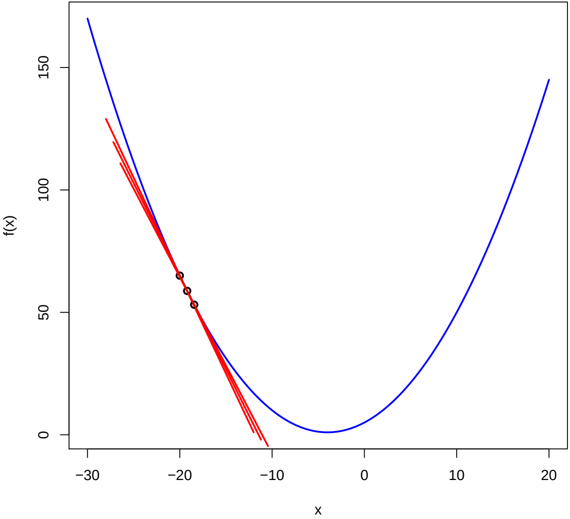
Logistic regression

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



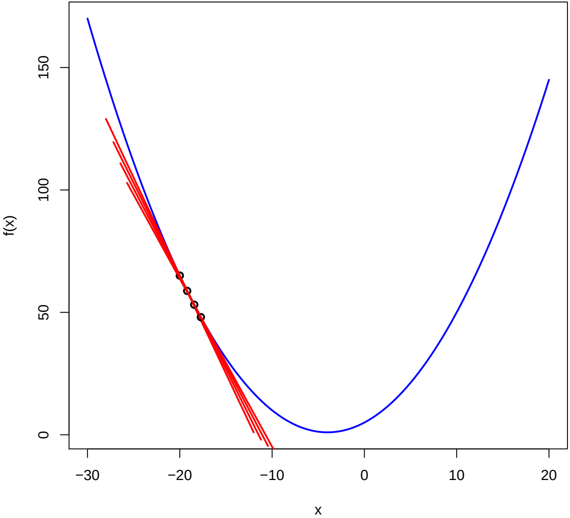
Logistic regression

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



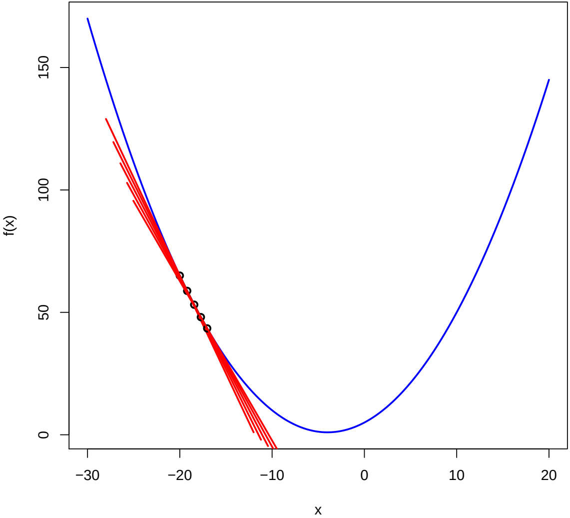
Logistic regression

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



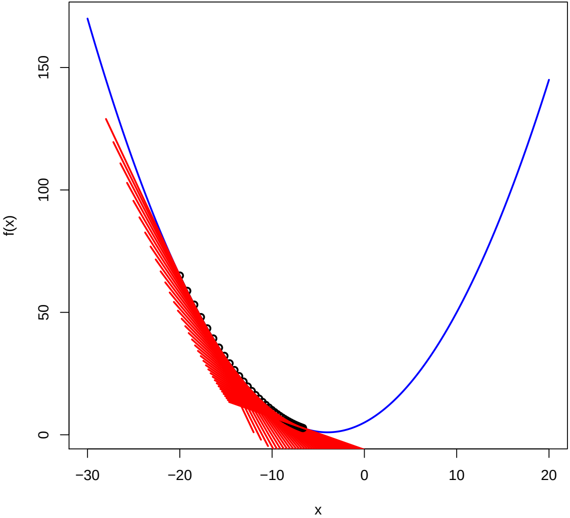
Logistic regression

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



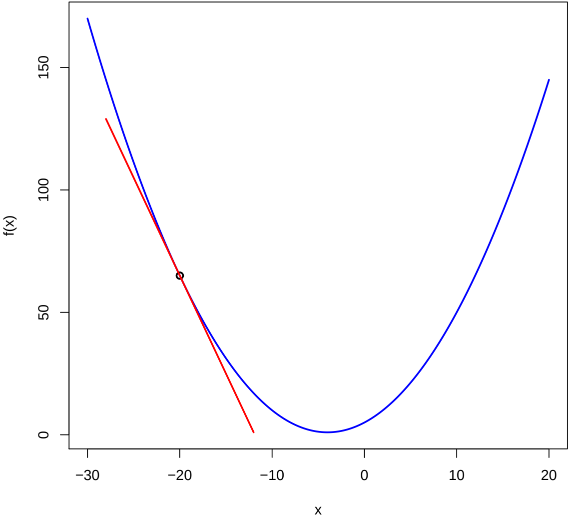
Logistic regression

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



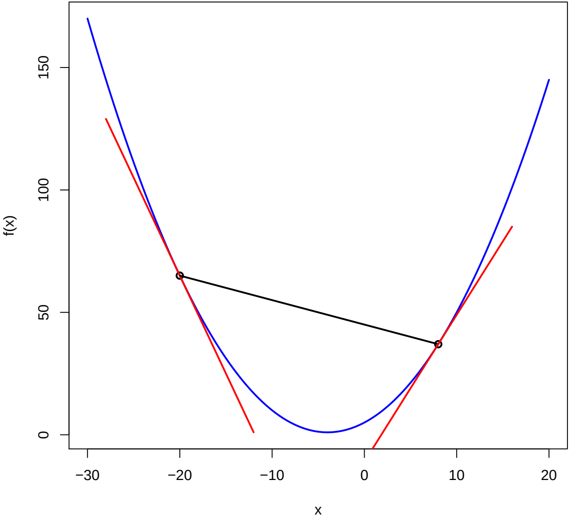
Logistic regression

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



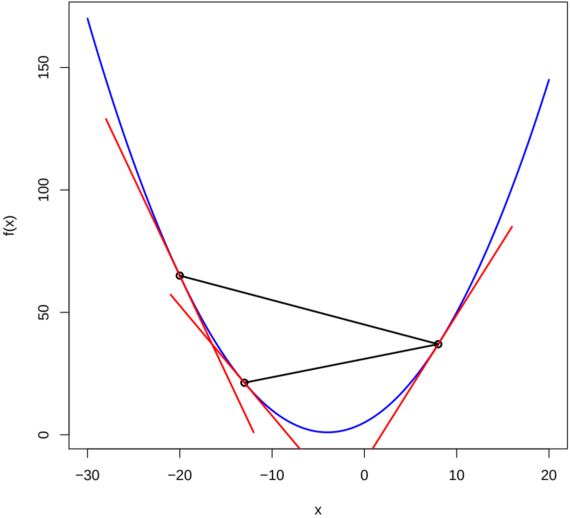
Logistic regression

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

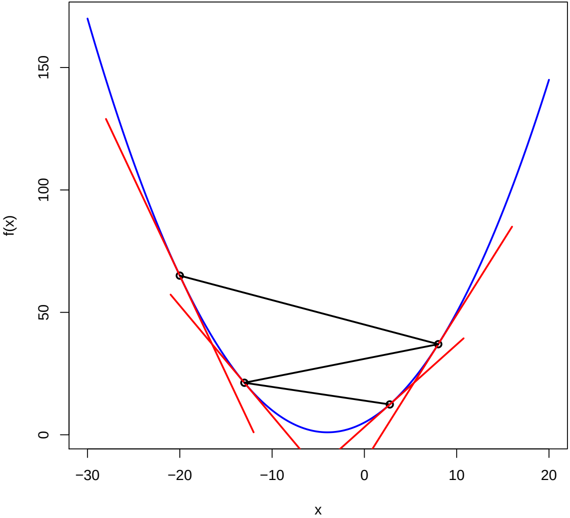


Logistic regression

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

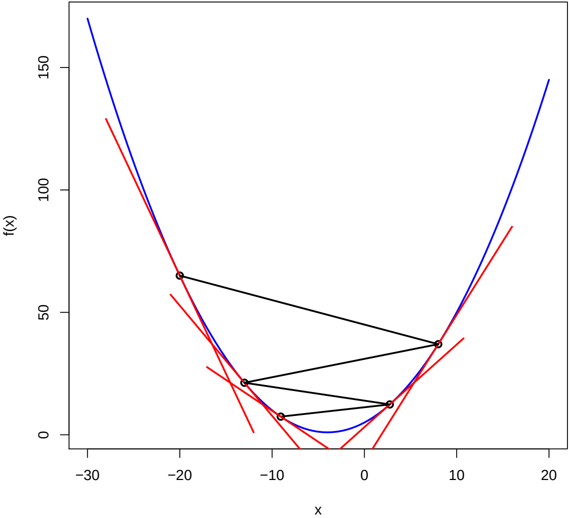


- Gradient descent example ($\alpha = 0.1$):



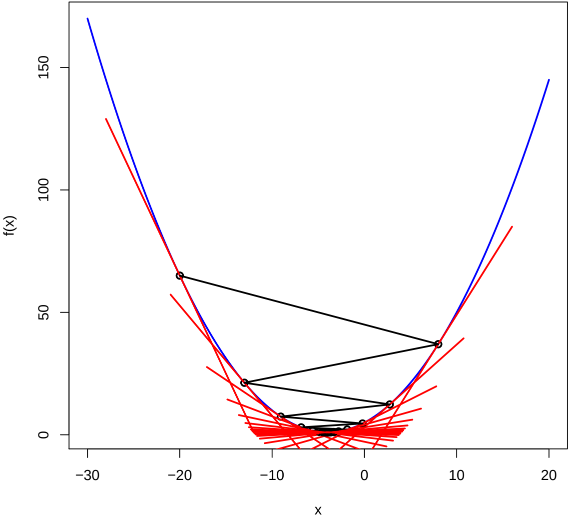
Logistic regression

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



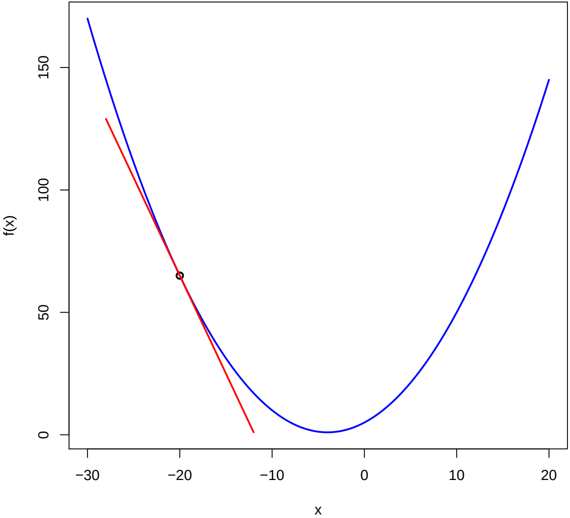
Logistic regression

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



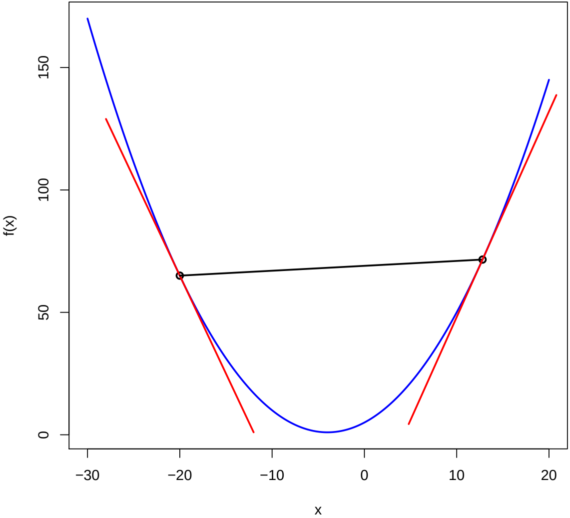
Logistic regression

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



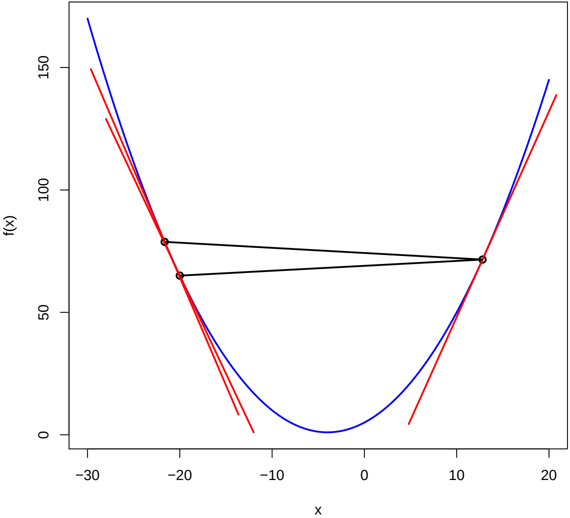
Logistic regression

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



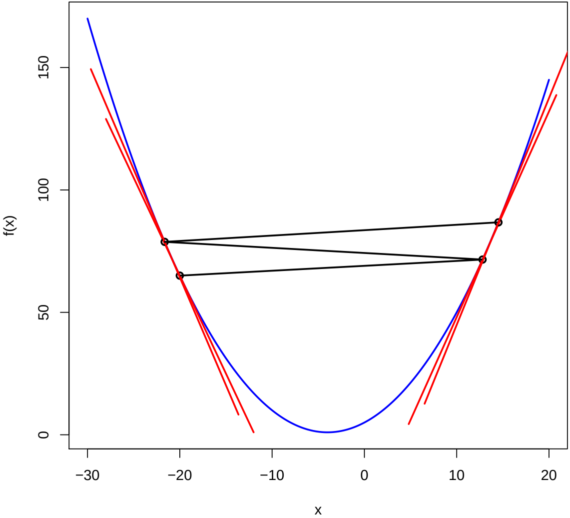
Logistic regression

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

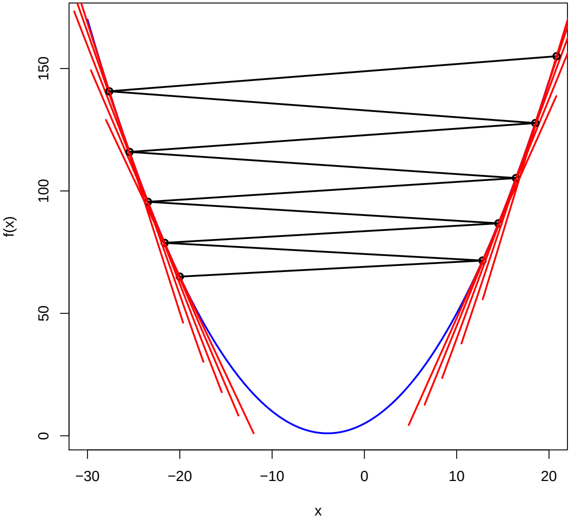


Logistic regression

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



- 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:

$$\frac{\partial \widehat{L}_i}{\partial w_j} = -(y_i' - \widehat{\eta}_i) x_{ij} \,,$$

    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 α
w = 0; //(or use random values)
while (approximate minimum is obtained) {
    Randomly shuffle examples in the training set
    for i=1 to n {
        w := w − α ∂L̂_i(w)/∂w
    }
}
```

$$\boldsymbol{w} := \boldsymbol{w} - \alpha \frac{\partial \widehat{L}_i(\boldsymbol{w})}{\partial \boldsymbol{w}}$$

Loss functions

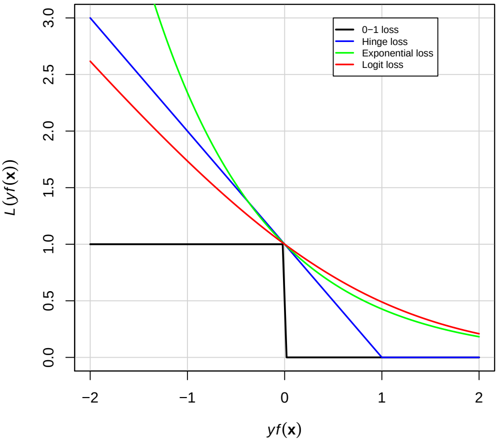


Figure: Loss functions for classification task

Outline

Summary

- Linear models for classification:
  - Linear regression.
  - Perceptron.
  - Support vector machines.
  - Logistic regression.

Bibliography

- T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning: Second Edition*.
  Springer, 2009
  `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*.
  Springer-Verlag, 2006
- David Barber. *Bayesian Reasoning and Machine Learning*.
  Cambridge University Press, 2012
  `http://www.cs.ucl.ac.uk/staff/d.barber/brml/`
- Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning From Data*.
  AMLBook, 2012