



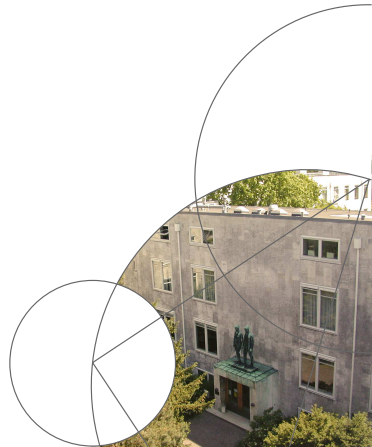
Faculty of Science



Linear Classification

Machine Learning

Christian Igel
Department of Computer Science



Outline

- ① Logistic Regression
- ② Linear Classification and Margins
- ③ Perceptron Learning
- ④ Convergence of Perceptron Learning
- ⑤ Summary



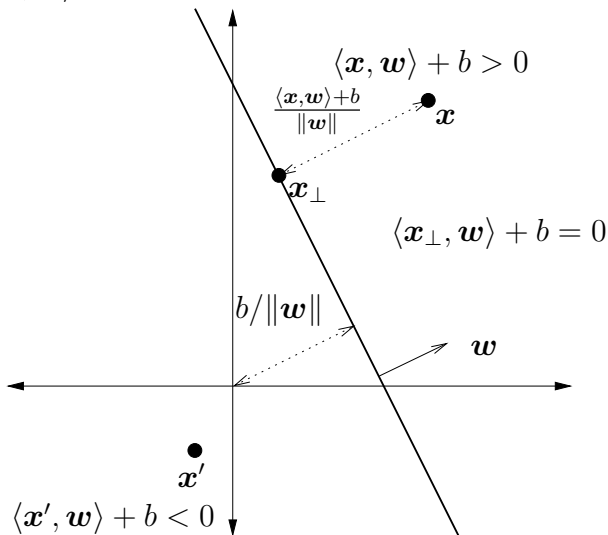
Outline

- ① Logistic Regression
- ② Linear Classification and Margins
- ③ Perceptron Learning
- ④ Convergence of Perceptron Learning
- ⑤ Summary



Linear functions

$$f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b$$



Decision functions

- Classification assigns an input $x \in \mathcal{X}$ to one of a finite set of classes $\mathcal{Y} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\}$, $2 \leq m$.
- One approach is to learn *discriminant functions* $\delta_k : \mathcal{X} \rightarrow \mathbb{R}$, $1 \leq k \leq m$, and assign a pattern x to class \hat{y} using

$$\hat{y} = h(x) = \operatorname{argmax}_k \delta_k(x) \ .$$



Linear classification

- We build affine linear decision functions

$$\delta(\mathbf{x}) = \sum_{i=1}^d w_i x_i + b = \mathbf{w}^\top \mathbf{x} + b$$

with $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.

- For convenience, we define $\tilde{\mathbf{x}}_i^\top = (x_1, \dots, x_d, 1)$ for $i = 1, \dots, N$ and $\tilde{\mathbf{w}}^\top = (w_1, \dots, w_d, b)$ and consider the equivalent formulation

$$\delta(\tilde{\mathbf{x}}) = \sum_{i=1}^{d+1} \tilde{w}_i \tilde{x}_i = \tilde{\mathbf{w}}^\top \tilde{\mathbf{x}} .$$

We omit the tilde in the following.



Binary decision functions

- If we have only two classes, we can consider a single function

$$\delta(x) = \delta_1(x) - \delta_2(x)$$

and the hypothesis

$$h(x) = \begin{cases} \mathcal{C}_1 & \text{if } \delta(x) > 0 \\ \mathcal{C}_2 & \text{otherwise} \end{cases}.$$

- For $\mathcal{Y} = \{-1, 1\}$ this is equal to

$$h(x) = \text{sgn}(\delta(x)) = \begin{cases} 1 & \text{if } \delta(x) > 0 \\ -1 & \text{otherwise} \end{cases}.$$



Decision functions and class posteriors

- If we know the class posteriors $P(Y | X)$ we can perform optimal classification: a pattern x is assigned to class \mathcal{C}_k with maximum $P(Y = \mathcal{C}_k | X = x)$, i.e.,

$$\hat{y} = h(x) = \operatorname{argmax}_k P(Y = \mathcal{C}_k | X = x)$$

or in the binary case with $\mathcal{Y} = \{-1, 1\}$

$$\delta(x) = P(Y = \mathcal{C}_1 | X = x) - P(Y = \mathcal{C}_2 | X = x)$$

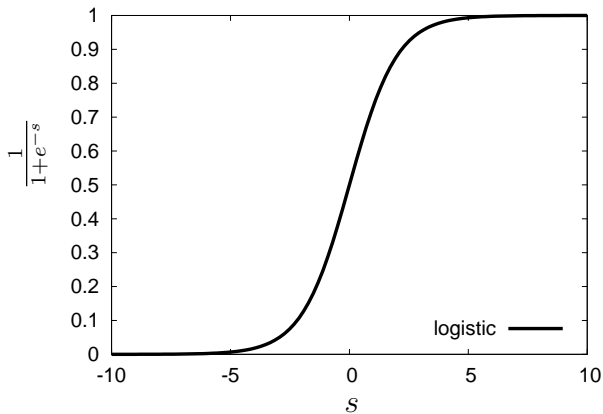
and $\hat{y} = h(x) = \operatorname{sgn}(\delta(x))$.

- $P(Y = \mathcal{C}_k | X = x)$ is proportional to the class-conditional density $p(X = x | Y = \mathcal{C}_k)$ times the class prior $P(Y = \mathcal{C}_k)$:

$$P(Y = \mathcal{C}_k | X = x) = \frac{p(X = x | Y = \mathcal{C}_k)P(Y = \mathcal{C}_k)}{p(X = x)}$$



Logistic function



$$\theta(s) = \frac{1}{1 + e^{-s}}$$



Predicting probabilities

- Instead of predicting the class label, we want to learn

$$f(\mathbf{x}) = P(Y = 1 \mid X = \mathbf{x})$$

assuming that the data is generated by

$$P(Y = y \mid X = \mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = 1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}.$$

- In the binary case, our model takes the form $h : \mathcal{X} \rightarrow [0, 1]$:

$$h(\mathbf{x}) = \theta(\mathbf{w}^\top \mathbf{x})$$



Likelihood function

- Our hypothesis h describes the probability distribution:

$$P(Y = y \mid X = \mathbf{x}; \mathbf{w}) = \theta(y\mathbf{w}^\top \mathbf{x}) = \begin{cases} \theta(\mathbf{w}^\top \mathbf{x}) & \text{for } y = 1 \\ 1 - \theta(\mathbf{w}^\top \mathbf{x}) & \text{for } y = -1 \end{cases}$$

- $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$
- **Likelihood (function)** of the parameters \mathbf{w} given training data S is the probability of observing S when the data is generated by h with parameters \mathbf{w} .
- Likelihood for i.i.d. S :

$$\prod_{i=1}^N P(Y = y_i \mid X = \mathbf{x}_i; h) \text{ or short } \prod_{i=1}^N P(y_i \mid \mathbf{x}_i)$$



Maximum likelihood

- **Learning principle:** Maximize the likelihood function!
- Equivalently, we can minimize the negative logarithmic likelihood.
- Negative log-likelihood (divided by N):

$$-\frac{1}{N} \ln \left(\prod_{i=1}^N P(y_i | \mathbf{x}_i) \right) = -\frac{1}{N} \sum_{i=1}^N \ln (P(y_i | \mathbf{x}_i))$$

- Plugging in our linear hypothesis gives the error function:

$$-\frac{1}{N} \sum_{n=1}^N \ln (\theta(y_n \mathbf{w}^\top \mathbf{x}_n)) = \frac{1}{N} \sum_{n=1}^N \ln (1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n})$$



Recall: Gradient

- The *gradient*

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_d} \right)^\top$$

points in the direction $\nabla f(\mathbf{x}) / \|\nabla f(\mathbf{x})\|$ giving maximum rate of change $\|\nabla f(\mathbf{x})\|$.



Gradient descent

- Consider learning by iteratively changing the parameters:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \Delta \boldsymbol{w}$$

- Simplest choice is (steepest) gradient descent

$$\Delta \boldsymbol{w} = -\eta \nabla f|_{\boldsymbol{w}}$$

with learning rate $\eta > 0$.



Gradient for training logistic regression

- For data $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$, we have the following gradient of the negative log-likelihood:

$$-\frac{1}{N} \sum_{n=1}^N \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^\top \mathbf{x}_n}}$$

- This equals:

$$-\frac{1}{N} \sum_{n=1}^N \left[\frac{y_n + 1}{2} - \theta(\mathbf{w}^\top \mathbf{x}) \right] \mathbf{x}_n$$

- Thus, for $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\} \subseteq (\mathbb{R}^n \times \{0, 1\})^N$, we have:

$$-\frac{1}{N} \sum_{n=1}^N [y_n - \theta(\mathbf{w}^\top \mathbf{x})] \mathbf{x}_n$$



Logistic regression algorithm (steepest descent)

Algorithm 1: Logistic regression

Input: data $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$,
learning rate η

Output: weights of linear hypothesis $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$

- 1 initialize \mathbf{w}
- 2 **repeat**
 - // gradient of negative log-likelihood over N
 - 3 $\mathbf{g} \leftarrow -\frac{1}{N} \sum_{n=1}^N \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^\top \mathbf{x}_n}}$
 - // model parameter update
 - 4 $\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$
- 5 **until** *stopping criterion is met*



Logistic regression algorithm (stochastic gradient descent, SGD)

Algorithm 2: Logistic regression

Input: data $\{(\mathbf{x}_1, y_1), \dots\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$, learning rate η

Output: weights of linear hypothesis $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$

- 1 initialize \mathbf{w}
 - 2 **repeat**
 - 3 pick $(\mathbf{x}, y) \in S$
 - 4 $\mathbf{w} \leftarrow \mathbf{w} + \eta \frac{y\mathbf{x}}{1+e^{y\mathbf{w}^\top \mathbf{x}}}$
 - 5 **until** *stopping criterion is met*
-



Logistic regression algorithm (mini-batch gradient descent)

Algorithm 3: Logistic regression

Input: data $\{(\mathbf{x}_1, y_1), \dots\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$, learning rate η

Output: weights of linear hypothesis $h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$

- 1 initialize \mathbf{w}
 - 2 **repeat**
 - 3 pick $S' \subset S$
 - 4 $\mathbf{g} \leftarrow -\frac{1}{|S'|} \sum_{(\mathbf{x}, y) \in S'} \frac{y\mathbf{x}}{1 + e^{y\mathbf{w}^\top \mathbf{x}}}$
 - 5 $\mathbf{w} \leftarrow \mathbf{w} - \eta \mathbf{g}$
 - 6 **until** *stopping criterion is met*
-



Multiple classes

- Binary, single decision function, $y \in \{0, 1\}$:

$$P(Y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-\delta(\mathbf{x})}} = \frac{e^{\delta(\mathbf{x})}}{1 + e^{\delta(\mathbf{x})}}$$

- Binary, two decision functions, $y \in \{1, 2\}$:

$$P(Y = y \mid \mathbf{x}) = \frac{e^{\delta_y(\mathbf{x})}}{e^{\delta_1(\mathbf{x})} + e^{\delta_2(\mathbf{x})}} = \frac{e^{\delta_y(\mathbf{x})+C}}{e^{\delta_1(\mathbf{x})+C} + e^{\delta_2(\mathbf{x})+C}}$$

for every constant C , thus logistic function is special case for $C = -\delta_1(\mathbf{x})$.

- Multiple classes, $y \in \{1, \dots, m\}$:

$$P(Y = y \mid \mathbf{x}) = \frac{e^{\delta_y(\mathbf{x})}}{\underbrace{\sum_{i=1}^m e^{\delta_i(\mathbf{x})}}_{\text{softmax function}}}$$



Outline

- ① Logistic Regression
- ② Linear Classification and Margins
- ③ Perceptron Learning
- ④ Convergence of Perceptron Learning
- ⑤ Summary



Margins I

The functional margin of an example (\mathbf{x}_i, y_i) with respect to a hyperplane (\mathbf{w}, b) is

$$\gamma_i := y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \quad .$$

The geometric margin of an example (\mathbf{x}_i, y_i) with respect to a hyperplane (\mathbf{w}, b) is

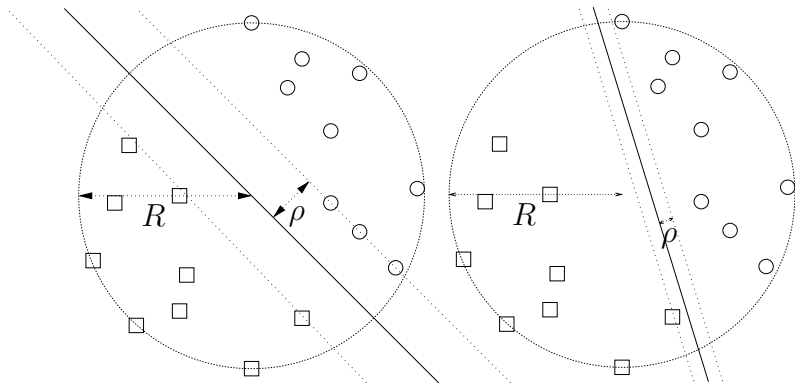
$$\rho_i := y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) / \|\mathbf{w}\| = \gamma_i / \|\mathbf{w}\| \quad .$$

A positive margin implies correct classification.

The margin of a hyperplane (\mathbf{w}, b) with respect to a training set S is $\min_i \rho_i$. The margin of a training set S is the maximum geometric margin over all hyperplanes. A hyperplane realizing this margin is called maximum margin hyperplane.



Margins II



Outline

- ① Logistic Regression
- ② Linear Classification and Margins
- ③ Perceptron Learning
- ④ Convergence of Perceptron Learning
- ⑤ Summary



Analyzing the Perceptron

Why should we look at the Perceptron?

- Linear classifiers such as perceptrons are the basis of technical neurocomputing
- Support Vector Machines are basically linear classifiers
- Basic concepts of learning theory can be explained easily:
 - Margins
 - Dual representation
 - Bounds involving margins and the radius of the ball containing the data



Perceptron learning algorithm (primal form)

For simplicity, consider hyperplanes with no bias ($b = 0$), i.e., $\mathcal{H} = \{h(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle) \mid \mathbf{w} \in \mathbb{R}^n\}$.

Algorithm 4: Perceptron

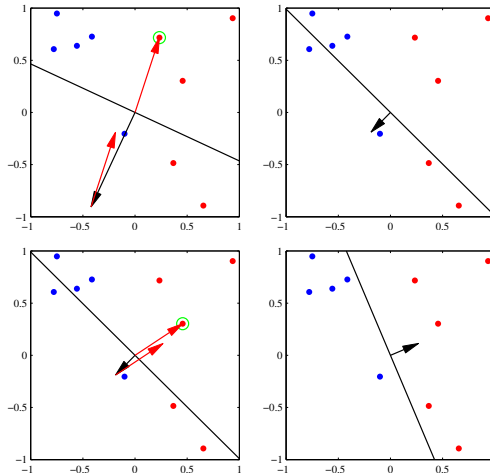
Input: separable data $\{(\mathbf{x}_1, y_1), \dots\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$

Output: hypothesis $h(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}_k, \mathbf{x} \rangle)$

```
1  $\mathbf{w}_0 \leftarrow \mathbf{0}; k \leftarrow 0$ 
2 repeat
3   for  $i = 1, \dots, N$  do
4     if  $y_i \langle \mathbf{w}_k, \mathbf{x}_i \rangle \leq 0$  then
5        $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + y_i \mathbf{x}_i$ 
6        $k \leftarrow k + 1$ 
7 until no mistake made within for loop
```



Perceptron learning in pictures



Outline

- ① Logistic Regression
- ② Linear Classification and Margins
- ③ Perceptron Learning
- ④ Convergence of Perceptron Learning
- ⑤ Summary



Novikoff

Theorem (Novikoff)

Let $S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ be a non-trivial training set (i.e., containing patterns of both classes), $b = \mathbf{0} = \sum_{i=1}^m 0\mathbf{x}_i$ and let

$$R \leftarrow \max_{1 \leq i \leq N} \|\mathbf{x}_i\| \ .$$

Suppose that there exists \mathbf{w}_{opt} and $\rho > 0$ such that $\|\mathbf{w}_{opt}\| = 1$ and

$$y_i \langle \mathbf{w}_{opt}, \mathbf{x}_i \rangle \geq \rho > 0$$

for $1 \leq i \leq N$. Then the number of updates k made by the online perceptron algorithm on S is at most

$$\left(\frac{R}{\rho} \right)^2 \ .$$



Novikoff, sketch of proof I

Let i be the index of the example in update k

$$\begin{aligned}\|\mathbf{w}_{k+1}\|^2 &= \langle \mathbf{w}_k + y_i \mathbf{x}_i, \mathbf{w}_k + y_i \mathbf{x}_i \rangle \\ &= \|\mathbf{w}_k\|^2 + 2y_i \langle \mathbf{w}_k, \mathbf{x}_i \rangle + \|\mathbf{x}_i\|^2 \\ &\leq \|\mathbf{w}_k\|^2 + R^2 \\ &\leq (k+1)R^2\end{aligned}$$



Novikoff, sketch of proof II

$$\begin{aligned}\langle \mathbf{w}_{\text{opt}}, \mathbf{w}_{k+1} \rangle &= \langle \mathbf{w}_{\text{opt}}, \mathbf{w}_k \rangle + y_i \langle \mathbf{w}_{\text{opt}}, \mathbf{x}_i \rangle \\ &\geq \langle \mathbf{w}_{\text{opt}}, \mathbf{w}_k \rangle + \rho \\ &\geq (k+1)\rho\end{aligned}$$

$$k^2 \rho^2 \leq \langle \mathbf{w}_{\text{opt}}, \mathbf{w}_k \rangle^2 \leq \|\mathbf{w}_{\text{opt}}\|^2 \|\mathbf{w}_k\|^2 \leq kR^2$$

$$k \leq \frac{R^2}{\rho^2}$$



Dual representation

- Weight vector of hyperplane computed by online perceptron algorithm can be written as

$$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$$

- Function $h(\mathbf{x}) = \text{sgn}(\delta(\mathbf{x}))$ can be written in dual coordinates

$$\begin{aligned}\delta(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x} \rangle \\ &= \left\langle \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i, \mathbf{x} \right\rangle \\ &= \sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle\end{aligned}$$



Perceptron learning algorithm (dual form)

Algorithm 5: Perceptron (dual form)

Input: separable data $\{(\mathbf{x}_1, y_1), \dots\} \subseteq (\mathbb{R}^n \times \{-1, 1\})^N$

Output: hypothesis $h(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^N \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle \right)$

```
1  $\alpha \leftarrow 0$ 
2 repeat
3   for  $i = 1, \dots, N$  do
4     if  $y_i \sum_{j=1}^N \alpha_j y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle \leq 0$  then
5        $\alpha_i \leftarrow \alpha_i + 1$ 
6 until no mistake made within for loop
```



Outline

- ① Logistic Regression
- ② Linear Classification and Margins
- ③ Perceptron Learning
- ④ Convergence of Perceptron Learning
- ⑤ Summary



Summary I

Logistic regression

- is easy to use, has in its simplest form no hyperparameters (not counting η),
- gives surprisingly good results, is highly recommended as baseline method,
- does typically not tend to overfit (assuming $d \ll N$), but does not capture non-linearities,
- can be used with non-linear transformations,
- can be parallelized and is applicable to “Big Data”.



Summary II

Hey, we also now know about

- perceptron learning,
- margins,
- dual representation,
- bounds involving margins and the radius of the ball containing the data.

