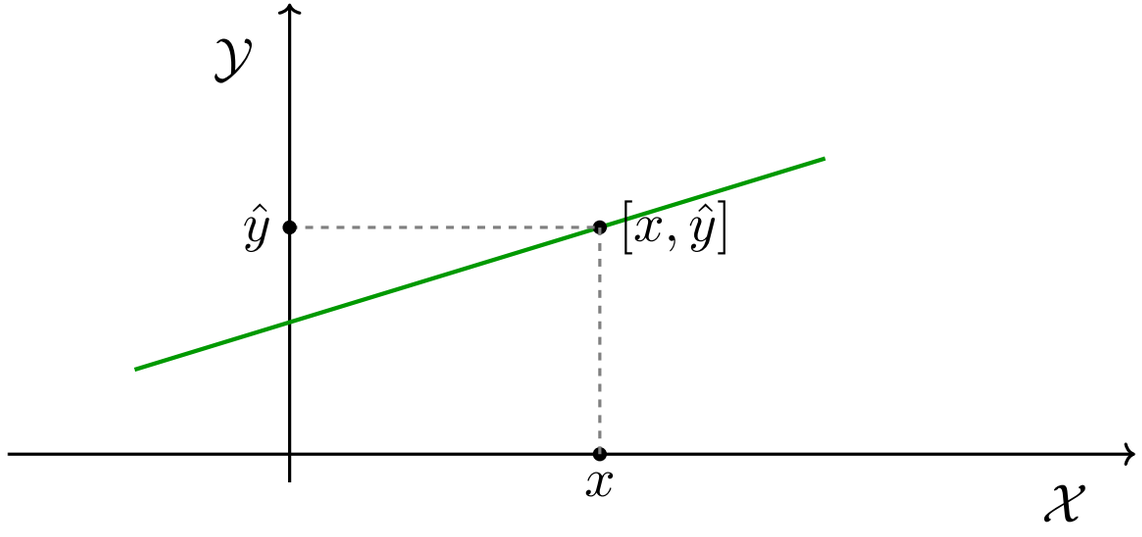
Machine Learning Fundamentals - LS2026
Linear models

Czech Technical University in Prague
V. Franc

Linear Regression



- Input: feature vector $\mathbf{x} = [x_1, \ldots, x_d] \in \mathcal{X} = \mathbb{R}^d$
- Target: $y \in \mathcal{Y} = \mathbb{R}$
- A linear predictor (regressor) has the form

$$\hat{y} = h(\mathbf{x}; \mathbf{w}, b) = \sum_{i=1}^{d} w_i \, x_i + b = \mathbf{w}^\top \mathbf{x} + b$$

- It is parametrized by a weight vector $\mathbf{w} \in \mathbb{R}^d$ and a biad (offset) $b \in \mathbb{R}$.

Linear Regression as Empirical Risk Minimization

Task: Given i.i.d. training samples $T_m = \big((x_1, y_1), \ldots, (x_m, y_m)\big) \in (\mathbb{R}^d \times \mathbb{R})^m$, learn a predictor minimizing the expected risk under the quadratic loss $\ell(y, \hat{y}) = (y - \hat{y})^2$. Assume that a good predictor lies in the hyposhesis space

$$\mathcal{H} = \left\{ h(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^\top \mathbf{x} + b \;\middle|\; \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \right\}$$

Empirical risk (training error) evaluated on T_m:

$$\hat{R}(T_m, \boldsymbol{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \big(h(\boldsymbol{x}_i; \boldsymbol{w}, b) - y_i \big)^2 = \frac{1}{m} \sum_{i=1}^{m} \big(\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i \big)^2$$

ERM learning:

$$(\boldsymbol{w}_m, b_m) = \arg \min_{\boldsymbol{w} \in \mathbb{R}^n, b \in \mathbb{R}} \hat{R}(T_m, \boldsymbol{w}, b)$$

Solution:

$$\begin{bmatrix} \boldsymbol{w}_m \\ b_m \end{bmatrix} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \qquad \text{where} \qquad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\top & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^\top & 1 \end{bmatrix} \in \mathbb{R}^{m \times (d+1)}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^{m \times 1}$$

Linear Regression via Maximum Likelihood

Generative model of the data:

- Input: $\mathbf{x} = (x_1, \ldots, x_d) \in \mathcal{X} = \mathbb{R}^d$, target: $y \in \mathcal{Y} = \mathbb{R}$.
- The inputs $\mathbf{x}$ assumed to be drawn from a distribution $p(\mathbf{x})$, which is left unspecified.
- Given an input $\mathbf{x}$, the target is generated as

$$y = \mathbf{w}^\top \mathbf{x} + b + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

  where $\mathcal{N}(0, \sigma^2)$ denotes a Gaussian distribution with mean 0 and variance σ^2.
- Therefore, the conditional distribution of the target is

$$p(y \mid \mathbf{x}; \mathbf{w}, b) = \mathcal{N}\big(\mathbf{w}^\top \mathbf{x} + b, \sigma^2\big) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{\big(y - (\mathbf{w}^\top \mathbf{x} + b)\big)^2}{2\sigma^2}\right).$$

- The pairs $(\mathbf{x}, y)$ are assumed to be drawn from a joint distribution

$$p(\mathbf{x}, y; \mathbf{w}, b) = p(\mathbf{x})\, p(y \mid \mathbf{x}; \mathbf{w}, b)$$

Linear Regression via Maximum Likelihood learning

Task: Given i.i.d. training samples $T_m = \big((x_1, y_1), \ldots, (x_m, y_m)\big) \in (\mathbb{R}^d \times \mathbb{R})^m$, learn a predictor minimizing the expected risk under the quadratic loss $\ell(y, \hat{y}) = (y - \hat{y})^2$. Assume the pairs $(\mathbf{x}, y)$ are drawn i.i.d. from a joint distribution $p(\mathbf{x}, y)$ belonging to the parametric family:

$$\mathcal{P} = \big\{ p(\mathbf{x})\, p(y \mid \mathbf{x}; \mathbf{w}, b) \;\big|\; \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \big\}$$

Log-likelihood function:

$$\begin{aligned} L(T_m, \mathbf{w}, b) &= \log \left(\prod_{i=1}^{m} p(\mathbf{x}_i)\, p(y_i \mid \mathbf{x}_i; \mathbf{w}, b) \right) \\ &= \sum_{i=1}^{m} \log p(\mathbf{x}_i) + \sum_{i=1}^{m} \log p(y_i \mid \mathbf{x}_i; \mathbf{w}, b) \\ &= \sum_{i=1}^{m} \log p(\mathbf{x}_i) - \frac{m}{2} \log(2\pi) - m \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{m} \big(\mathbf{w}^\top \mathbf{x}_i + b - y_i \big)^2 \end{aligned}$$

Remark: If we model only the conditional distribution $p(y \mid \mathbf{x}; \mathbf{w}, b)$ and leave $p(\mathbf{x})$ unspecified, then the resulting objective is called the *conditional likelihood*.

Linear Regression via Maximum Likelihood

Maximum-likelihood learning:

Given training data T_m, estimate the unknown parameters $(\mathbf{w}, b)$ by maximizing the average log-likelihood:

$$(\mathbf{w}_m, b_m) \in \operatorname*{arg\,max}_{\mathbf{w}\in\mathbb{R}^d,\, b\in\mathbb{R}} L(T_m, \mathbf{w}, b)$$

$$= \operatorname*{arg\,max}_{\mathbf{w},\, b} \left(\sum_{i=1}^{m} \log p(\mathbf{x}_i) - \frac{m}{2}\log(2\pi) - m\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{m}\left(\mathbf{w}^\top\mathbf{x}_i + b - y_i\right)^2 \right)$$

$$= \operatorname*{arg\,min}_{\mathbf{w},\, b} \frac{1}{2\sigma^2}\sum_{i=1}^{m}\left(\mathbf{w}^\top\mathbf{x}_i + b - y_i\right)^2$$

Closed-form solution:

$$\begin{bmatrix}\mathbf{w}_m \\ b_m\end{bmatrix} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \qquad \text{where} \qquad \mathbf{X} = \begin{bmatrix}\mathbf{x}_1^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_m^\top & 1\end{bmatrix} \in \mathbb{R}^{m\times(d+1)}, \quad \mathbf{y} = \begin{bmatrix}y_1 \\ \vdots \\ y_m\end{bmatrix} \in \mathbb{R}^{m\times 1}$$

Remark: This solution coincides with the one obtained earlier from empirical risk minimization.

Linear Regression via Maximum Likelihood Learning

Plug-in Bayes predictor

- Assume the squared loss

$$\ell(y, y') = (y - y')^2.$$

- The Bayes-optimal predictor minimizes the true risk

$$R(p, h) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}\big[\ell\big(y, h(\mathbf{x})\big)\big].$$

Under squared loss, it is given by the conditional mean:

$$\hat{y} = \arg\min_{y'} \mathbb{E}_{y\sim p(y|\mathbf{x})}\big[(y - y')^2\big] = \mathbb{E}_{y\sim p(y|\mathbf{x})}[y].$$

- Assume the learned posterior is a good approximation of the true posterior:

$$p(y \mid \mathbf{x}) \approx p(y \mid \mathbf{x}; \boldsymbol{w}_m, b_m) = \mathcal{N}\big(\boldsymbol{w}_m^\top \mathbf{x} + b_m, \sigma^2\big).$$

- Then the plug-in Bayes predictor becomes

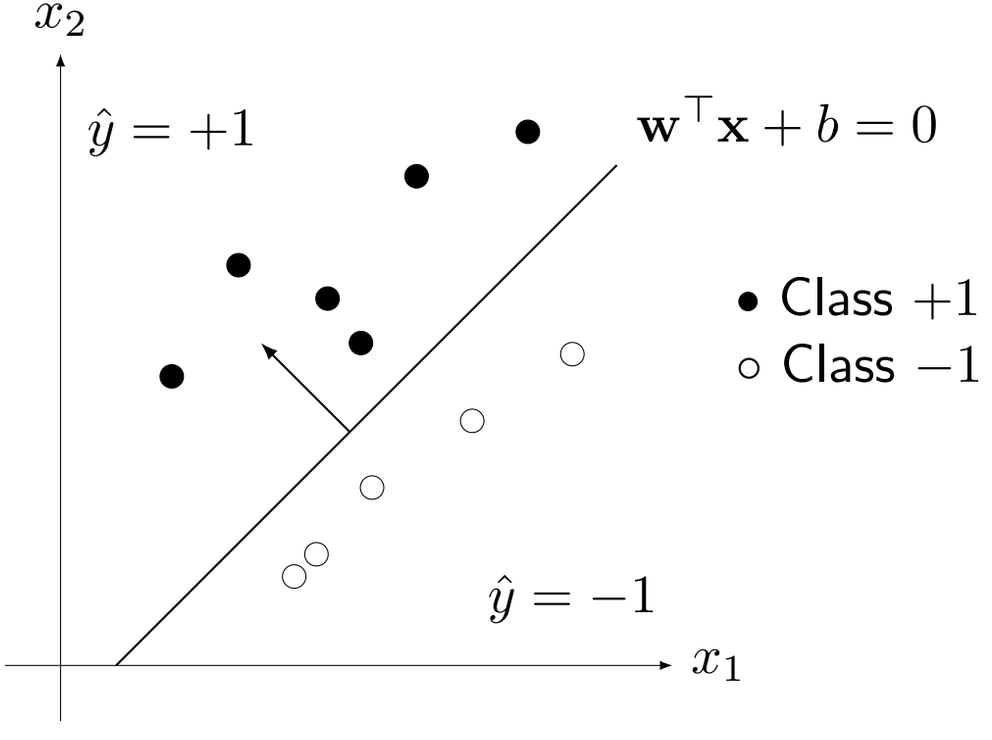
$$\hat{y} = h(\mathbf{x}; \boldsymbol{w}_m, b_m) = \boldsymbol{w}_m^\top \mathbf{x} + b_m.$$

Binary Linear Classifier

- Input: feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_d] \in \mathcal{X} \subseteq \mathbb{R}^d$
- Target: binary label $y \in \mathcal{Y} = \{-1, +1\}$
- A linear binary classifier has the form

$$\hat{y} = h(\mathbf{x}; \mathbf{w}, b) = \text{sign}\left(\sum_{i=1}^{d} w_i\, x_i + b\right) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b) = \begin{cases} +1 & \text{if} \quad \mathbf{w}^\top \mathbf{x} + b \geq 0, \\ -1 & \text{if} \quad \mathbf{w}^\top \mathbf{x} + b < 0. \end{cases}$$

- It is parameterized by a weight vector $\mathbf{w} \in \mathbb{R}^d$ and an bias (offset) $b \in \mathbb{R}$.

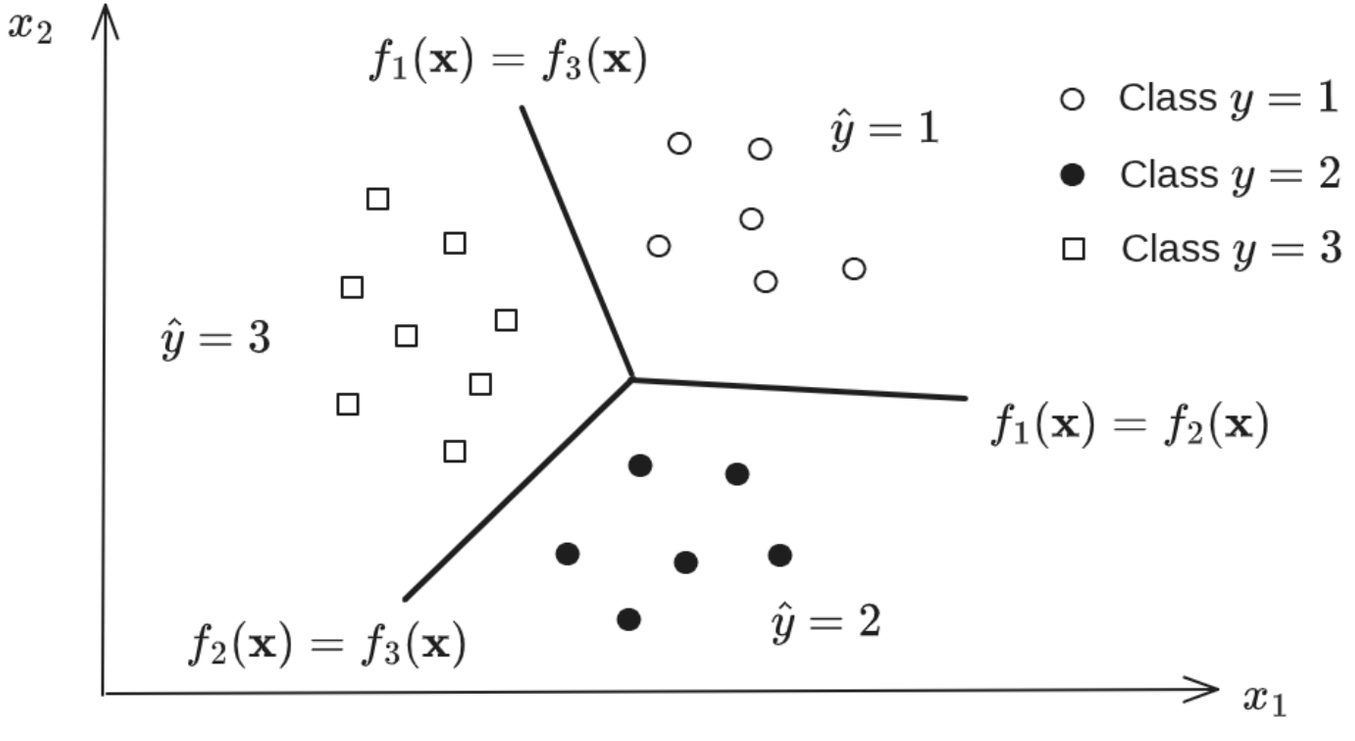


Multiclass Linear Classifier

- Input: a feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_d] \in \mathcal{X} \subseteq \mathbb{R}^d$
- Output: categorical label $y \in \mathcal{Y} = \{1, 2, \ldots, Y\}$
- A multiclass linear classifier predicts the class with the largest score:

$$\hat{y} = h(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \arg\max_{y \in \mathcal{Y}} \left(\mathbf{w}_y^\top \mathbf{x} + b_y \right) = \arg\max_{y \in \mathcal{Y}} f_y(\mathbf{x})$$

- It is parameterized by the weight matrix $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_Y] \in \mathbb{R}^{d \times Y}$ and the bias vector $\mathbf{b} = [b_1, b_2, \ldots, b_Y] \in \mathbb{R}^Y$.



Generic Linear Classifier

- Input: $x \in \mathcal{X}$ where $\mathcal{X}$ is an arbitrary set.
- Output: label $y \in \mathcal{Y}$ where $\mathcal{Y}$ is an arbitrary finite set.
- Let

$$\boldsymbol{\phi} \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$$

  be a joint feature map of an input-label pair.
- A generic linear classifier is defined as

$$h(x; \boldsymbol{\theta}) = \arg\max_{y \in \mathcal{Y}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(x, y) = \arg\max_{y \in \mathcal{Y}} s(x, y)$$

- It is parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^n$.
- *Interpretation:* for a given input $x \in \mathcal{X}$, the classifier predicts the label $y \in \mathcal{Y}$ that achieves the highest compatibility score

$$s(x, y) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(x, y)$$

- Binary and multiclass linear classifiers are special cases of this framework.

Generic Linear Classifier

- A binary linear classifier is a special case of the generic linear classifier:

$$\begin{aligned} h(\mathbf{x}; \mathbf{w}, b) &= \operatorname{sign}(\mathbf{w}^\top \mathbf{x} + b) \\ &= \operatorname*{arg\,max}_{y \in \{-1,+1\}} \; y \, (\mathbf{w}^\top \mathbf{x} + b) \\ &= \operatorname*{arg\,max}_{y \in \{-1,+1\}} \; \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}, y) \\ &= h(\mathbf{x}; \boldsymbol{\theta}). \end{aligned}$$

- Here, the joint feature map is

$$\boldsymbol{\phi} \colon \mathbb{R}^d \times \{-1, +1\} \to \mathbb{R}^{d+1}, \qquad \boldsymbol{\phi}(\mathbf{x}, y) = [y \, \mathbf{x}, \, y],$$

and the parameter vector is

$$\boldsymbol{\theta} = [\mathbf{w}, \, b].$$

Generic Linear Classifier

- A multiclass linear classifier is a special case of the generic linear classifier:

$$h(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \arg\max_{y \in \mathcal{Y}} \left(\mathbf{w}_y^\top \mathbf{x} + b_y \right)$$
$$= \arg\max_{y \in \mathcal{Y}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}, y).$$

- The joint feature map

$$\boldsymbol{\phi} \colon \mathbb{R}^d \times \mathcal{Y} \to \mathbb{R}^{(d+1)Y}$$

  is defined by placing the block $[\mathbf{x}, 1]$ into the y-th slot and zeros elsewhere:

$$\boldsymbol{\phi}(\mathbf{x}, y) = \big[[\mathbf{0}, 0], \ldots, \underbrace{[\mathbf{x}, 1]}_{y\text{-th slot}}, \ldots, [\mathbf{0}, 0] \big].$$

- The parameter vector is

$$\boldsymbol{\theta} = [\mathbf{w}_1, b_1, \mathbf{w}_2, b_2, \ldots, \mathbf{w}_Y, b_Y].$$

Learning a Linear Classifier from Separable Examples

- A training example $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ is classified correctly if

$$y_i = h(x_i; \boldsymbol{\theta}) = \arg\max_{y \in \mathcal{Y}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(x_i, y).$$

- This is equivalent to the set of inequalities

$$\boldsymbol{\theta}^\top \boldsymbol{\phi}(x_i, y_i) > \boldsymbol{\theta}^\top \boldsymbol{\phi}(x_i, y), \qquad \forall y \in \mathcal{Y} \setminus \{y_i\}.$$

Definition. The training set $T_m = \big((x_i, y_i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \ldots, m\big)$ is *linearly separable* with respect to the joint feature map $\boldsymbol{\phi}\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$ if there exists a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^n$ such that

$$\boldsymbol{\theta}^\top \boldsymbol{\phi}(x_i, y_i) > \boldsymbol{\theta}^\top \boldsymbol{\phi}(x_i, y), \qquad \forall i \in \{1, \ldots, m\},\ \forall y \in \mathcal{Y} \setminus \{y_i\}. \tag{1}$$

ERM-based learning. Learning a generic linear classifier

$$h(x; \boldsymbol{\theta}) = \arg\max_{y \in \mathcal{Y}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(x, y)$$

from a linearly separable training set T_m leads to solving the system of linear inequalities (1).

Perceptron Learning Algorithm

Task: Given a set of vectors $\{\mathbf{a}_i \in \mathbb{R}^n \mid i = 1, 2, \ldots, m\}$, find a vector $\boldsymbol{\theta} \in \mathbb{R}^n$ such that

$$\boldsymbol{\theta}^\top \mathbf{a}_i > 0, \qquad \forall i \in \{1, 2, \ldots, m\}.$$

Algorithm:

1. Initialize
$$\boldsymbol{\theta} \leftarrow \mathbf{0}.$$
2. Find an index $u \in \{1, 2, \ldots, m\}$ such that
$$\boldsymbol{\theta}^\top \mathbf{a}_u \leq 0.$$
3. If no such index exists, return $\boldsymbol{\theta}$. Otherwise, update
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mathbf{a}_u$$
and go back to step 2.

Novikoff Theorem

Task: Given a set of vectors $\{\mathbf{a}_i \in \mathbb{R}^n \mid i = 1, 2, \ldots, m\}$, find a vector $\boldsymbol{\theta} \in \mathbb{R}^n$ such that

$$\boldsymbol{\theta}^\top \mathbf{a}_i > 0, \qquad \forall i \in \{1, 2, \ldots, m\}. \tag{2}$$

Theorem (Novikoff): If the system of inequalities (2) is feasible, then the Perceptron learning algorithm terminates after at most

$$\frac{R^2}{\gamma^2}$$

updates, where

$$R = \max_{i=1,\ldots,m} \|\mathbf{a}_i\|, \qquad \gamma = \max_{\|\theta\|=1} \min_{i=1,\ldots,m} \boldsymbol{\theta}^\top \mathbf{a}_i.$$

Remark: The number of iterations does not depend on the number of inequalitites m neither on the dimension n.

Perceptron Learning Algorithm

Task: Given a linearly separable training set $T_m = \big((x_1, y_1), \ldots, (x_m, y_m)\big) \in (\mathcal{X} \times \mathcal{Y})^m$, find parameters $\boldsymbol{\theta} \in \mathbb{R}^n$ such that the linear classifier $h(x; \boldsymbol{\theta}) = \arg\max_{y \in \mathcal{Y}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(x, y)$ has zero training error on T_m, that is,

$$h(x_i; \boldsymbol{\theta}) = y_i, \qquad \forall i \in \{1, \ldots, m\}.$$

Algorithm:

1. Initialize
$$\boldsymbol{\theta} \leftarrow \mathbf{0}.$$
2. Find a misclassified example $(x_u, y_u) \in T_m$ such that
$$y_u \neq \hat{y}_u = \arg\max_{y \in \mathcal{Y}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(x_u, y) \qquad \text{prediction}$$
3. If no misclassified example exists, return $\boldsymbol{\theta}$. Otherwise, update
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\phi}(x_u, y_u) - \boldsymbol{\phi}(x_u, \hat{y}_u) \qquad \text{parameter update}$$
and go back to step 2.

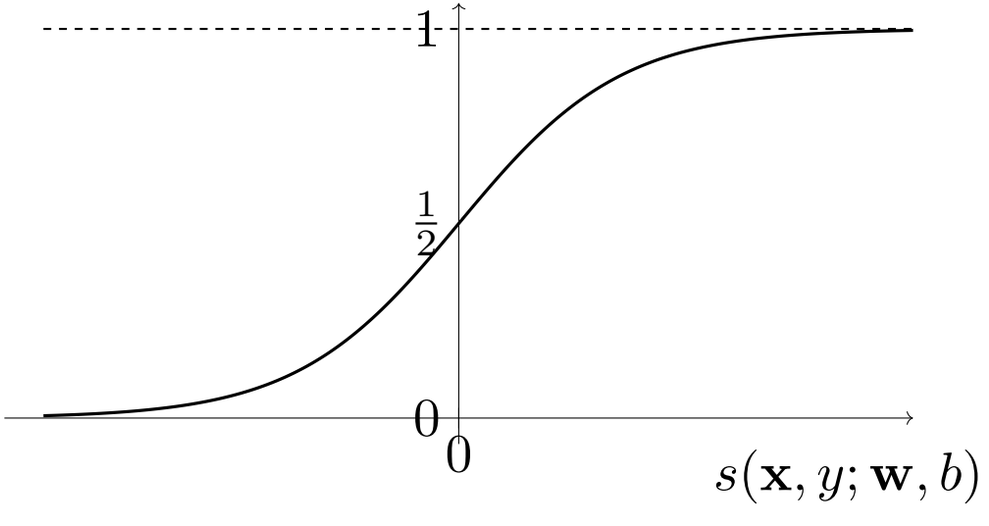
Logistic Regression

Generative model of the data:

- Input: a feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_d] \in \mathcal{X} = \mathbb{R}^d$
- Output: a binary label $y \in \mathcal{Y} = \{-1, +1\}$
- The input $\mathbf{x}$ is an i.i.d. sample drawn from $p(\mathbf{x})$, which is not modeled explicitly.
- Given $\mathbf{x}$, the label is distributed according to

$$p(y \mid \mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + \exp\big(-y(\mathbf{w}^\top \mathbf{x} + b)\big)} = \frac{1}{1 + \exp\big(-s(\mathbf{x}, y; \mathbf{w}, b)\big)}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the model parameters.



Logistic Regression

Task: Given i.i.d. training samples $T_m = \big((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\big) \in (\mathbb{R}^d \times \{+1, -1\})^m$, learn a binary classifier minimizing the expected risk under the 0/1-loss $\ell(y, \hat{y}) = [\![y \neq \hat{y}]\!]$. Assume the pairs $(\mathbf{x}, y)$ are generated from a joint distribution $p(\mathbf{x}, y)$ belonging to the parametric family:

$$\mathcal{P} = \big\{ p(\mathbf{x})\, p(y \mid \mathbf{x}; \mathbf{w}, b) \;\big|\; \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \big\}$$

Log-likelihood function:

$$\begin{aligned} L(T_m, \boldsymbol{w}, b) &= \log \left(\prod_{i=1}^{m} p(\mathbf{x}_i)\, p(y_i \mid \mathbf{x}_i; \boldsymbol{w}, b) \right) \\ &= \sum_{i=1}^{m} \log p(\mathbf{x}_i) + \sum_{i=1}^{m} \log p(y_i \mid \mathbf{x}_i; \boldsymbol{w}, b) \\ &= \sum_{i=1}^{m} \log p(\mathbf{x}_i) + m - \sum_{i=1}^{m} \log(1 + \exp(-y_i(\mathbf{w}^T \mathbf{x}_i + b)) \end{aligned}$$

MLE: leads to solving a smooth convex optimization problem

$$(\mathbf{w}_m, b_m) = \arg\max_{\mathbf{w}, b} L(T_m, \mathbf{w}, b) = \arg\min_{\mathbf{w}, b} \sum_{i=1}^{m} \log\big(1 + \exp(-y_i(\mathbf{w}^T \mathbf{x}_i + b))\big)$$

Logistic Regression

Plug-in Bayes predictor:

- Assume 0/1-loss

$$\ell(\hat{y}, y') = [\![\hat{y} \neq y']\!]$$

- The Bayes optimal predictor, minimizing the true risk

$$R(p, h) = \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y)}[\ell(y, h(\mathbf{x})]$$

leads to MAP predcitor

$$\hat{y} = \arg\min_{y'} \mathbb{E}_{y\sim p(y|\mathbf{x})}[\![y \neq y']\!] = \begin{cases} +1 & \text{if} \quad p(y = +1 \mid \mathbf{x}) \geq p(y = -1 \mid \mathbf{x}) \\ -1 & \text{if} \quad p(y = +1 \mid \mathbf{x}) < p(y = -1 \mid \mathbf{x}) \end{cases}$$

- Assume the learned posterior is a good proxy of the true posterior, i.e.

$$p(y \mid \mathbf{x}) \approx p(y \mid \mathbf{x}; \mathbf{w}_m, b_m) = \frac{1}{1 + \exp\big(-y(\boldsymbol{w}_m^\top \mathbf{x} + b_m)\big)}$$

- The plug-in Bayes predictor for the logist posterior:

$$\hat{y} = h(\mathbf{x}; \mathbf{w}_m, b_m) = \text{sign}(\mathbf{w}_m^\top \mathbf{x} + b_m)$$

Summary

- Linear regression via Emprical Risk Minimization principle.
- Linear regression via Maximum Likelihood Learning.
- Linear classifier (binary, multiclass, generic).
- Perceptrin Learning Algorithm.
- Novikoff theorem
- Logistic regression