Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Linear Classification Models

Paul Prasse, Niels Landwehr, Tobias Scheffer

Overview

- Linear classification models
- Empirical risk minimization
  - Gradient descent method
  - Inexact line search
  - Stochastic gradient descent methods
- Loss functions and regularizers for classification
- Special cases
  - Perceptron
  - Support vector machines
- Multi-class classification

Classification

- Input: an instance $\mathbf{x} \in X$
  - E.g., X can be a vector space over *attributes*
  - The Instance is then an assignment of attributes.
  - $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$ is a feature vector

- Output: Class $y \in Y$; where Y is a finite set.
  - The class is also referred to as the *target* attribute
  - y is also referred to as the (class) *label*

$$\mathbf{x} \longrightarrow \textbf{classifier} \longrightarrow y$$

Classification: Example

- Input: Instance $\mathbf{x} \in X$
  - X : the set of all possible combinations of regiment of medication

Attribute	Instance $\mathbf{x}$
Medication #1 included?	0
	1
⋮	1
	0
	1
Medication #6 included?	0

Attribute values
Feature vector

Medication combination

- Output: $y \in Y = \{\text{toxic}, \text{ok}\}$  / 

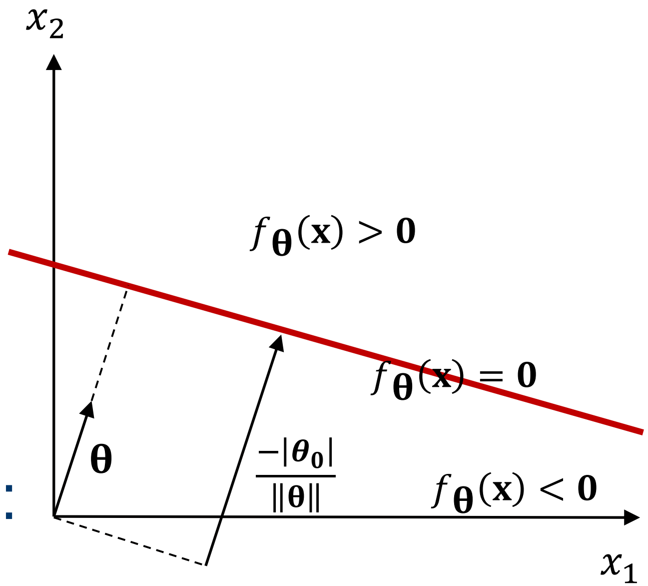
 $\longrightarrow$ **classifier** $\longrightarrow$ 

Linear Classification Models

- Hyperplane given by normal vector & displacement:

$$H_{\boldsymbol{\theta}} = \left\{\mathbf{x} \middle| f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + \theta_0 = 0\right\}$$

- Example: $X = \mathbb{R}^2$



- Decision function:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + \theta_0$$

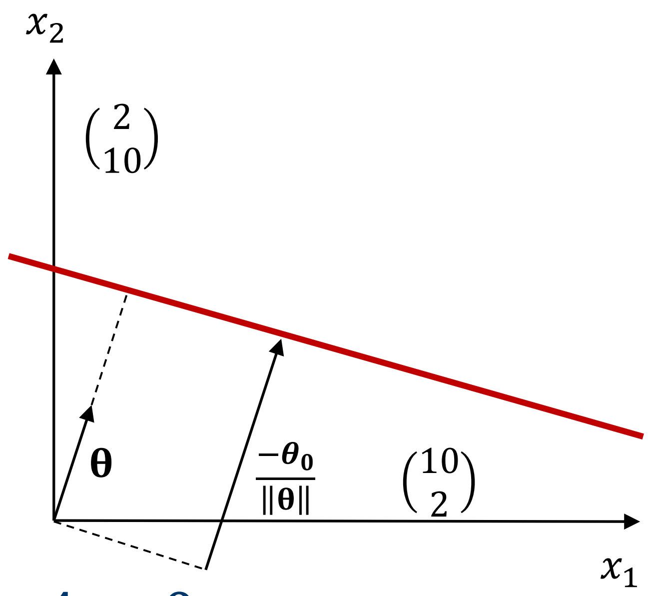
- Binary classifier, $y \in \{+1, -1\}$:

$$y_{\boldsymbol{\theta}}(\mathbf{x}) = \operatorname{sign}\left(f_{\boldsymbol{\theta}}(\mathbf{x})\right)$$

Linear Classification Models

- Hyperplane given by normal vector & displacement:

$$H_{\boldsymbol{\theta}} = \left\{\mathbf{x} \middle| f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} + \theta_0 = 0\right\}$$



- Example: $X = \mathbb{R}^2$
- Decision function:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\begin{pmatrix}1\\3\end{pmatrix} - 20$$

- Example points:

$$f_{\boldsymbol{\theta}}\begin{pmatrix}10\\2\end{pmatrix} = (10 \quad 2)\begin{pmatrix}1\\3\end{pmatrix} - 20 = -4 < 0$$

$$f_{\boldsymbol{\theta}}\begin{pmatrix}2\\10\end{pmatrix} = (2 \quad 10)\begin{pmatrix}1\\3\end{pmatrix} - 20 = 12 > 0$$

Linear Classification Model

- Offset can "disappear" into parameter vector.
- Example
  - Before: $f_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 1 \\ 3 \end{pmatrix} - 20$
  - After: $f_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -20 \\ 1 \\ 3 \end{pmatrix}$
- New constant attribute $x_0 = 1$ added to all instances
- Offset θ_0 integrated into $\boldsymbol{\theta}$.

Learning Linear Classifiers

- Input to the Learner: Training data T_n.
  - $\mathbf{X} = \begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{pmatrix}$  
  - $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$  
- Training Data:
  $T_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

- Output: a Model
  -  $\mapsto$ 
    $y_{\boldsymbol{\theta}} : X \rightarrow Y$
  - Linear classifier:
    $y_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{cases} \text{[sad face]} & \text{if } \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta} \geq 0 \\ \text{[happy face]} & otherwise \end{cases}$  

Linear classifier with parameter vector $\boldsymbol{\theta}$.

Overview

- Linear classification models
- Empirical risk minimization
  - Gradient descent method
  - Inexact line search
  - Stochastic gradient descent methods
- Loss functions and regularizers for classification
- Special cases
  - Perceptron
  - Support vector machines
- Multi-class classification

Regularized Empirical Risk Minimization

- Solve
$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \lambda\, \Omega(\boldsymbol{\theta})$$
- Loss function $\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$: cost of the model's output $f_{\boldsymbol{\theta}}(\mathbf{x})$ when the true value is y.
  - The empirical risk is $\hat{R}_n(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i)$
  - Empirical estimate of risk $R(\boldsymbol{\theta}) = \int \ell(f_{\boldsymbol{\theta}}(\mathbf{x}), y) d\mathrm{P}_{\mathbf{x},y}$
- Regularizer $\Omega(\boldsymbol{\theta})$ & trade-off parameter $\lambda \geq 0$:
  - Background information about preferred solutions
  - Provides numerical stability (Tikhonov-Regularizer)
  - allows for tighter error bounds (PAC-Theory)

Regularized Empirical Risk Minimization

- Solve

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Linear model:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(\mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

Regularized Empirical Risk Minimization

- Linear model: solve

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- How to find solution:
  - Classification: No analytic solution but numeric solutions (gradient descent, cutting plane, interior point method)
  - Regression: analytic solution.

Regularized Empirical Risk Minimization

- Linear classification model: minimize

$$\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Gradient:
  - Vector of the derivatives with respect to each individual parameter
  - Direction of the steepest increase of the function $L(\boldsymbol{\theta})$.

$$\nabla L(\boldsymbol{\theta}) = \begin{pmatrix} \dfrac{\partial L(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \dfrac{\partial L(\boldsymbol{\theta})}{\partial \theta_m} \end{pmatrix}$$

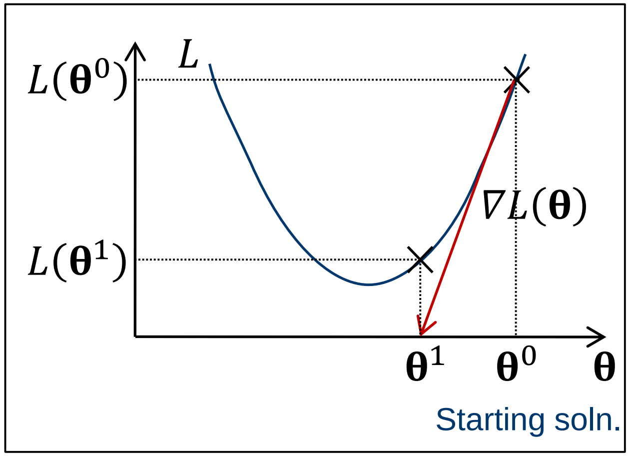
Regularized Empirical Risk Minimization

- Linear classification model: minimize

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Gradient descent method:

RegERM(*Data:* $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$)
  Set $\boldsymbol{\theta}^0 = \mathbf{0}$ and $t = 0$
  DO
    Compute gradient $\nabla L(\boldsymbol{\theta}^t)$
    Compute step size α^t
    Set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t \nabla L(\boldsymbol{\theta}^t)$
    Set $t = t + 1$
  WHILE $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t+1}\| > \varepsilon$
RETURN $\boldsymbol{\theta}^t$



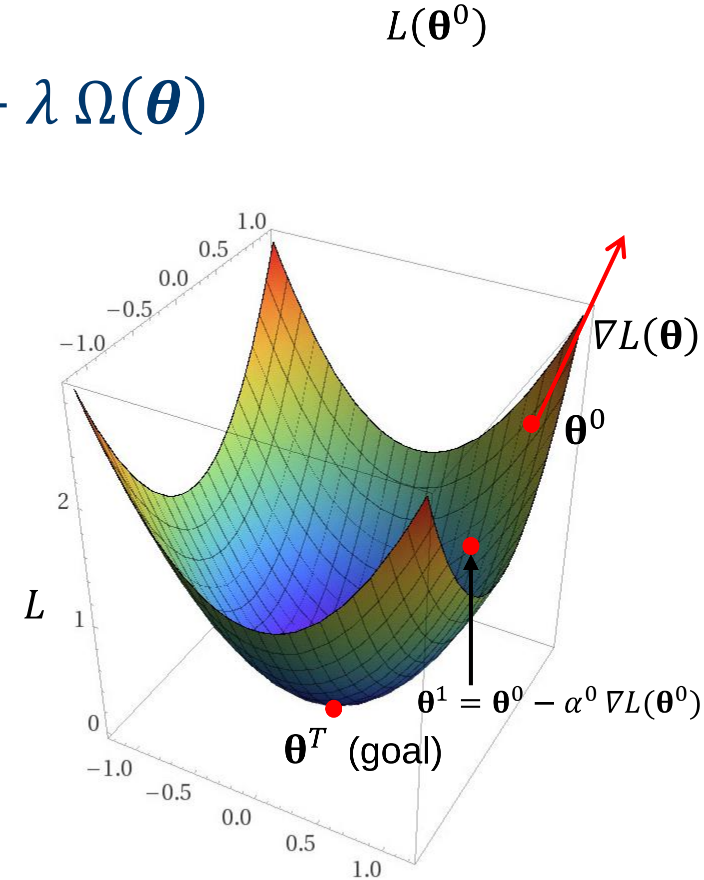
Regularized Empirical Risk Minimization

- Linear classification model: minimize

$$\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Gradient descent method:

RegERM(*Data:* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$)
    Set $\boldsymbol{\theta}^0 = \mathbf{0}$ and $t = 0$
    DO
        Compute gradient $\nabla L(\boldsymbol{\theta}^t)$
        Compute step size α^t
        Set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t \nabla L(\boldsymbol{\theta}^t)$
        Set $t = t + 1$
    WHILE $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t+1}\| > \varepsilon$
RETURN $\boldsymbol{\theta}^t$



Regularized Empirical Risk Minimization

- Linear classification model: minimize

$$\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(\mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}, y_i\right) + \lambda\, \Omega(\boldsymbol{\theta})$$

- Gradient descent method:

RegERM(*Data:* $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$)

  Set $\boldsymbol{\theta}^0 = \mathbf{0}$ and $t = 0$

  DO

    Compute gradient $\nabla L(\boldsymbol{\theta}^t)$

    Compute step size α^t

    Set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t \nabla L(\boldsymbol{\theta}^t)$

    Set $t = t + 1$

  WHILE $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t+1}\| > \varepsilon$

RETURN $\boldsymbol{\theta}^t$

- The step size α^t can be determined through
  - Line search
  - Barzilai-Borwein method
  - …

Intelligent Data Analysis

ERM: Gradient Method with Line Search

- Determine step size through line search:

```
RegERM-LineSearch(Data: (x_1, y_1), ..., (x_n, y_n))
    Set θ^0 = 0 and t = 0
    DO
        Compute gradient ∇L(θ^t)
        Choose step size α^t:
            α^t = argmin_{α>0} L(θ^t − α∇L(θ^t))
        Set θ^{t+1} = θ^t − α^t ∇L(θ^t)
        Set t = t + 1
    WHILE ||θ^t − θ^{t+1}|| > ε
RETURN θ^t
```

- In practice it is often too expensive to compute the optimal step size.
  - Necessary Criterion: $L\left(\boldsymbol{\theta}^t - \alpha \nabla L(\boldsymbol{\theta}^t)\right) < L(\boldsymbol{\theta}^t)$.

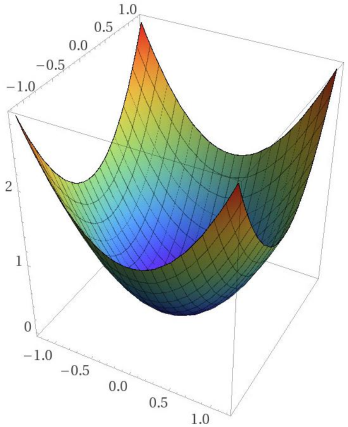
ERM: Gradient with Inexact Line Search

- Determine step size through inexact line search:

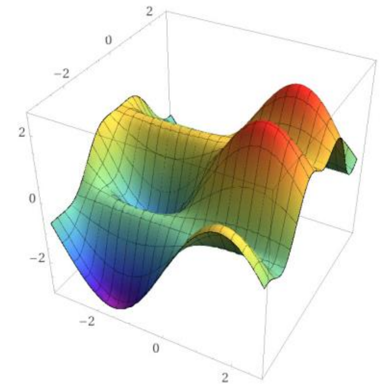
```
RegERM-LineSearch(Data: (x_1, y_1), ..., (x_n, y_n))
    Set θ^0 = 0 and t = 0
    DO
        Compute gradient ∇L(θ^t)
        Set α^t = 1
        WHILE L(θ^t − α^t ∇L(θ^t)) ≥ L(θ^t)
            Set α^t = α^t/2
        Set θ^{t+1} = θ^t − α^t ∇L(θ^t)
        Set t = t + 1
    WHILE ‖θ^t − θ^{t+1}‖ > ε
RETURN θ^t
```

Regularized Empirical Risk Minimization

- Properties of the gradient descent method:
  - Optimization criterion improved with every step.
  - Converges to the global minimum of the optimization criterion when this criterion is convex.
- The sum of convex functions is convex.
- Therefore, optimization criterion is convex if
  - Loss function is convex and
  - Regularizer is convex



convex



not convex

ERM: Stochastic Gradient Method

- Idea: Determine the gradient for a random subset of the samples (e.g., a single instance).
- Less computation per optimization step, but only approximate descent direction.
- Optimization criterion with regularizer in sum:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \frac{\lambda}{n} \Omega(\boldsymbol{\theta}) \right]$$

- Stochastic gradient for a single instance:

$$\nabla_{\mathbf{x}_i} L(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \frac{\lambda}{n} \frac{\partial}{\partial \boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$$

ERM: Stochastic Gradient Method

- Approximate gradient using single examples.

RegERM-Stoch(*Data:* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$)

  Set $\boldsymbol{\theta}^0 = \mathbf{0}$ and $t = 0$

  DO

    Shuffle data randomly

    FOR $i = 1, \ldots, n$

      Compute subset gradient $\nabla_{\mathbf{x}_i} L(\boldsymbol{\theta}^t)$

      Compute step size α^t

      Set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t \nabla_{\mathbf{x}_i} L(\boldsymbol{\theta}^t)$

      Set $t = t + 1$

    END

  WHILE $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t+1}\| > \varepsilon$

RETURN $\boldsymbol{\theta}^t$

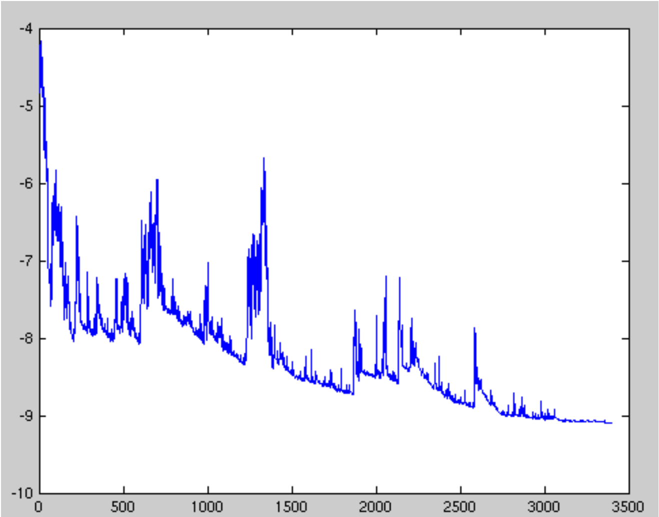
ERM: Stochastic Gradient Method

- In every step only one summand of the optimization criterion is improved.
- The total optimization criterion can be worsened by these individual steps.

- Converges to the optimum if the step sizes satisfy:

$\sum_{t=1}^{\infty} \alpha^t = \infty$ and $\sum_{t=1}^{\infty} (\alpha^t)^2 < \infty$

(Robbins & Monro, 1951)

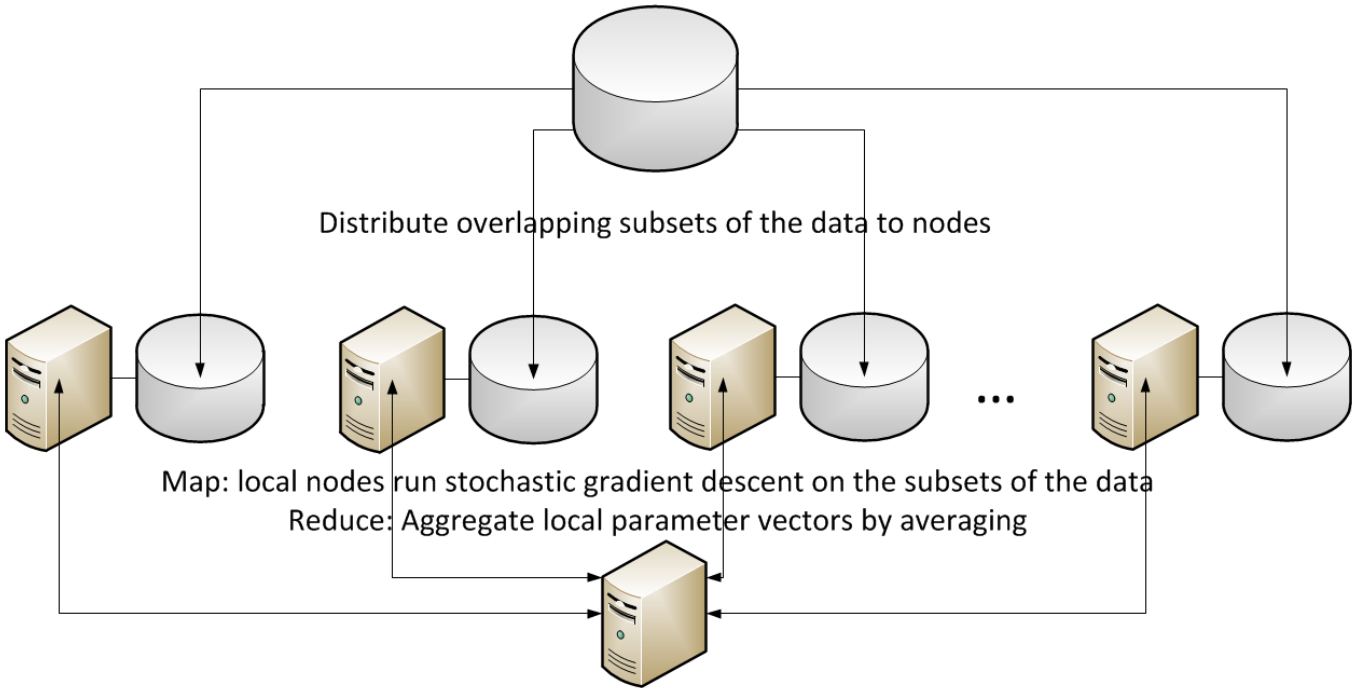


ERM: Parallel Stochastic Gradient

- Stochastic Gradient slows down when training data does not fit into main memory.
- Examples then have to be paged into and out of memory.
- For even larger training samples, data may not fit onto local persistent memory, have to be moved over the network during iterations.
- Remedy: distribute data over multiple nodes, perform computation in parallel on these nodes.

ERM: Parallel Stochastic Gradient

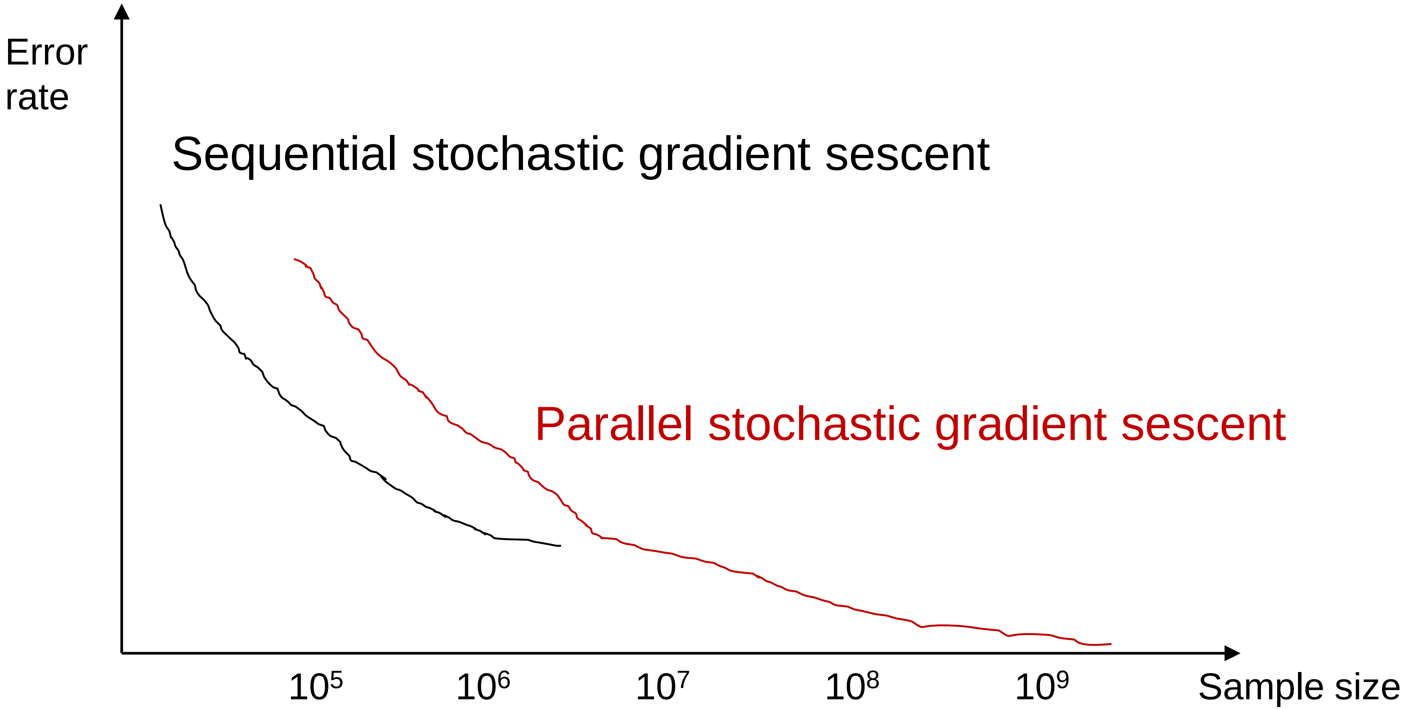
- Resulting parameters: average of parameters found by parallel stochastic gradient on data subsets.



ERM: Parallel Stochastic Gradient

- Resulting parameters: average of parameters found by parallel stochastic gradient on data subsets.
- Averaging the local parameter vectors is an approximation: typically not as good as sequantial stochastic gradient descent would on all be.
- Caveat: for non-convex problems, local parameters can be different local minima; averaging different local minima can be bad.

ERM: Parallel Stochastic Gradient



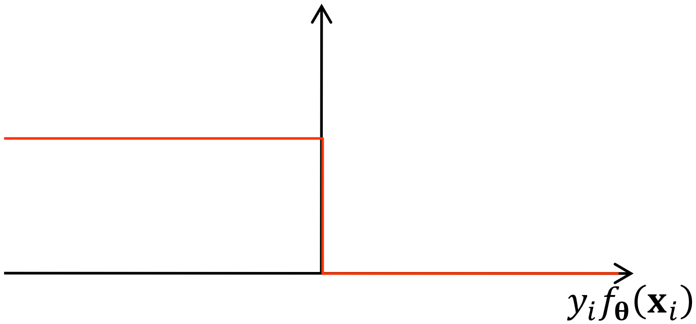
Overview

- Linear classification models
- Empirical risk minimization
  - Gradient descent method
  - Inexact line search
  - Stochastic gradient descent methods
- Loss functions and regularizers for classification
- Special cases
  - Perceptron
  - Support vector machines
- Multi-class classification

ERM: Loss Functions for Classification

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} \overbrace{1 \quad -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ \underbrace{0 \quad -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$

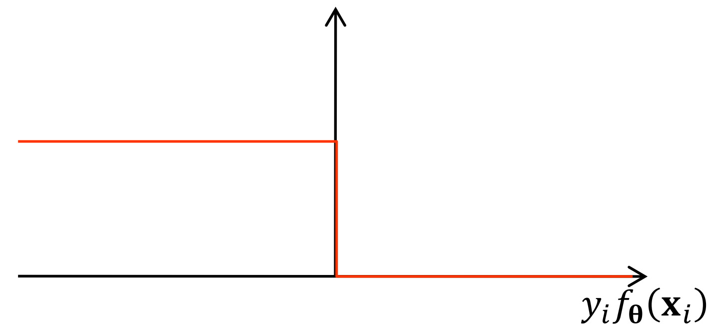


ERM: Loss Functions for Classification

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 & \overbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ 0 & \underbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$

Zero-one loss is not convex
$\Rightarrow$ difficult to minimize!



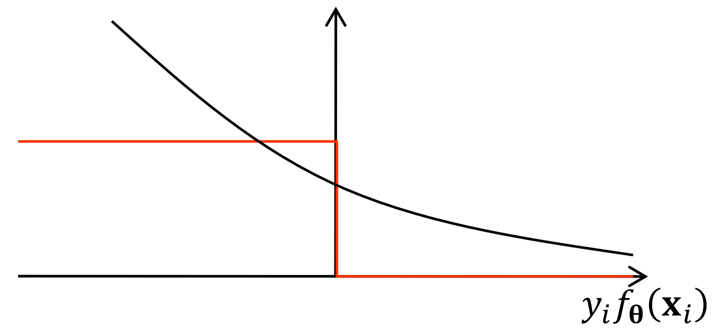
ERM: Loss Functions for Classification

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} \overbrace{1 \quad -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\operatorname{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ \underbrace{0 \quad -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\operatorname{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$

- Logistic loss:

$$\ell_{log}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \log\left(1 + e^{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)}\right)$$



ERM: Loss Functions for Classification

- Zero-one loss:

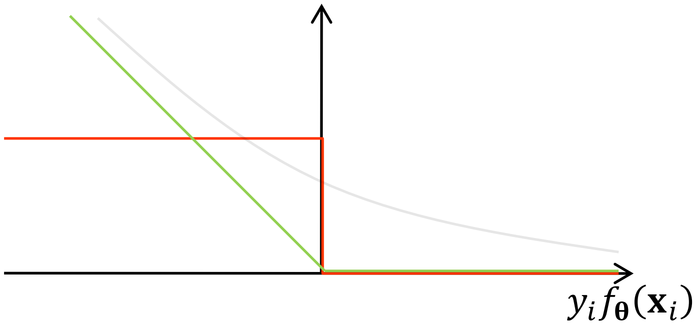
$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 & \overbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\operatorname{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ 0 & \underbrace{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\operatorname{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$

- Logistic loss:

$$\ell_{log}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \log\left(1 + e^{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)}\right)$$

- Perceptron loss:

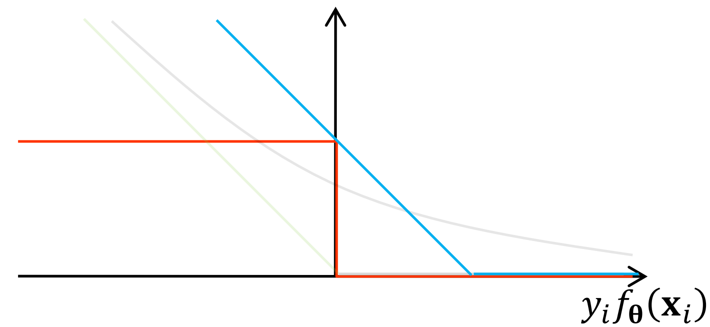
$$\ell_p(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\left(0, -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$



ERM: Loss Functions for Classification

- Zero-one loss:

$$\ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} \overbrace{1 \quad -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0}^{\operatorname{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) \neq y_i} \\ \underbrace{0 \quad -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0}_{\operatorname{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i)) = y_i} \end{cases}$$



- Logistic loss:

$$\ell_{log}(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \log\left(1 + e^{-y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)}\right)$$

- Perceptron loss:

$$\ell_p(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\left(0, -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$

- Hinge loss:

$$\ell_h(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$

ERM: Regularizers for Classification

- Idea: use as few attributes as possible:
  - $\Omega_0(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_0 =$ number of j with $\theta_j \neq 0$

Ω_0 is not convex $\Rightarrow$ difficult to minimize!

- Manhattan norm (encourages scarcity):

  $\Omega_1(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_1 = \sum_{j=1}^{m} |\theta_j|$

- Squared Euclidean norm (encourages small weights):
  - $\Omega_2(\boldsymbol{\theta}) \propto \|\boldsymbol{\theta}\|_2^2 = \sum_{j=1}^{m} \theta_j^{\ 2}$

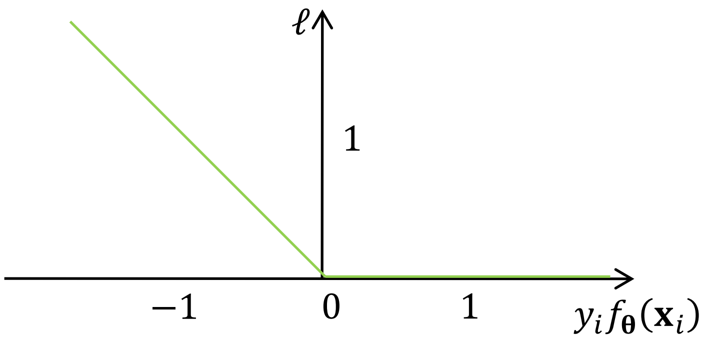
Overview

- Linear classification models
- Empirical risk minimization
  - Gradient descent method
  - Inexact line search
  - Stochastic gradient descent methods
- Loss functions and regularizers for classification
- Special cases
  - Perceptron
  - Support vector machines
- Multi-class classification

ERM: Perceptron

- Loss function:

$$\ell_p(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \le 0 \end{cases} = \max\big(0, -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\big)$$



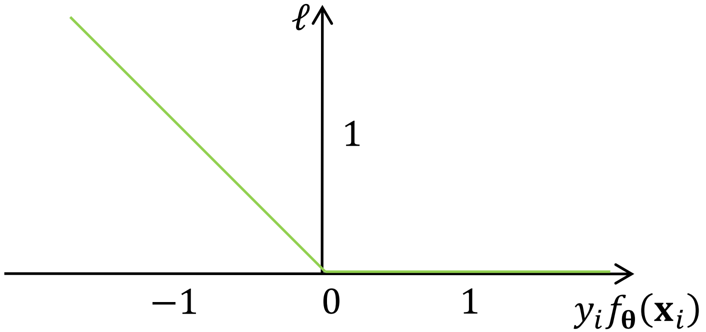
- No regularizer
- Classes $y \in \{-1, +1\}$
- Stochastic gradient method:
  - $\nabla L_{\mathbf{x}_i}(\boldsymbol{\theta}) = ?$

Rosenblatt, 1960

ERM: Perceptron

- Loss function:

$$\ell_p(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\left(0, -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$$



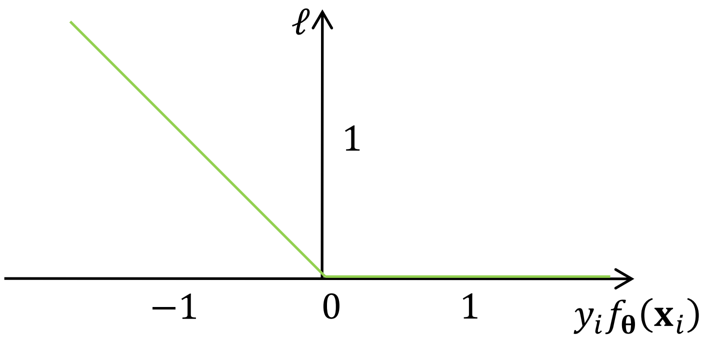
- No regularizer
- Classes $y \in \{-1, +1\}$
- Stochastic gradient method:
  - $\nabla L_{\mathbf{x}_i}(\boldsymbol{\theta}) = \Bigg\{$

Rosenblatt, 1960

ERM: Perceptron

- Loss function:

$$\ell_p(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases} = \max\big(0, -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\big)$$



- No regularizer
- Classes $y \in \{-1, +1\}$
- Stochastic gradient method:
  - $\nabla L_{\mathbf{x}_i}(\boldsymbol{\theta}) = \begin{cases} -y_i \mathbf{x}_i & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & -y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) < 0 \end{cases}$

Rosenblatt, 1960

ERM: Perceptron Algorithm

Perceptron(*Instances* $\{(\mathbf{x}_i, y_i)\}$)

  Set $\boldsymbol{\theta} = \mathbf{0}$

  DO

    FOR $i = 1, \ldots, n$

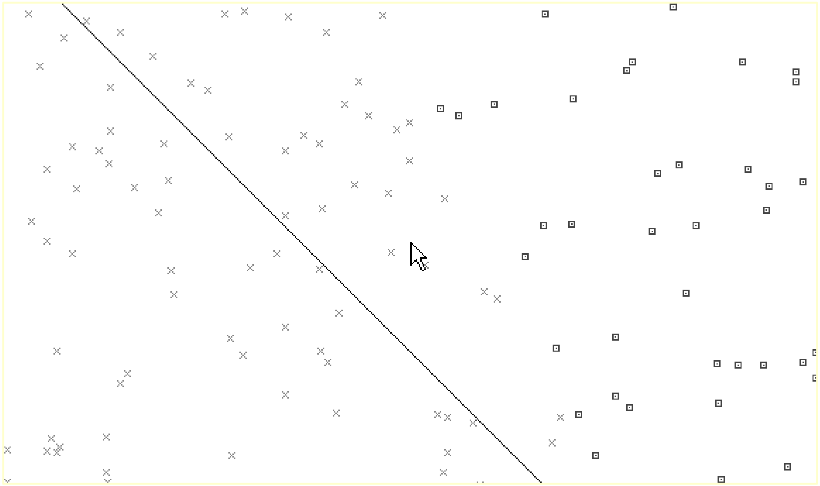
      IF $\quad y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$

      THEN $\quad \boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

    END

  WHILE $\boldsymbol{\theta}$ changes

  RETURN $\boldsymbol{\theta}$



- Stochastic gradient method with $\varepsilon = 0$ and step size $\alpha^t = 1$
  - Terminates, although $\sum_{t=1}^{\infty} (\alpha^t)^2 = \infty$ when data is linearly separable.

Rosenblatt, 1960

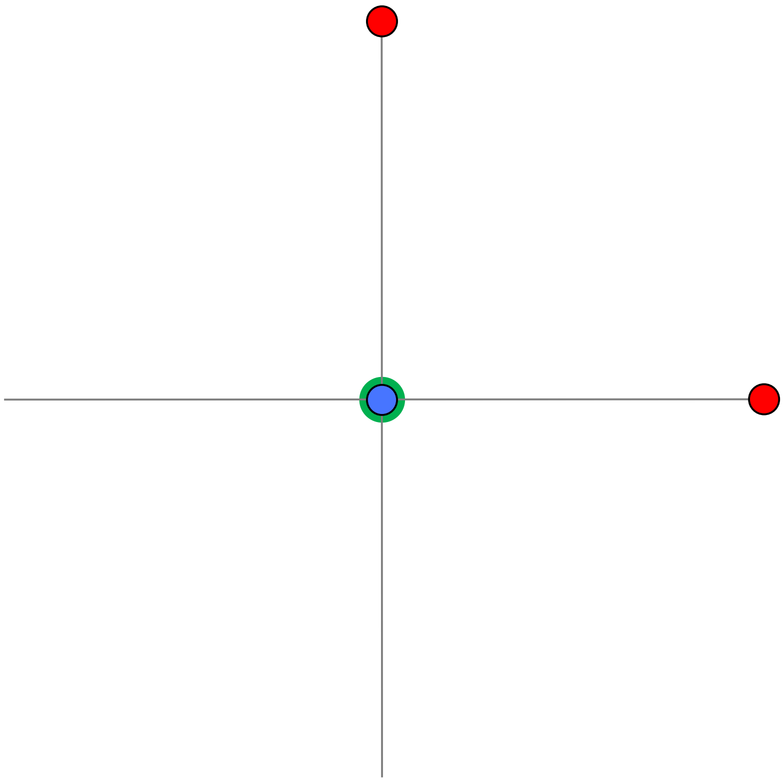
IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \le 0$
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$

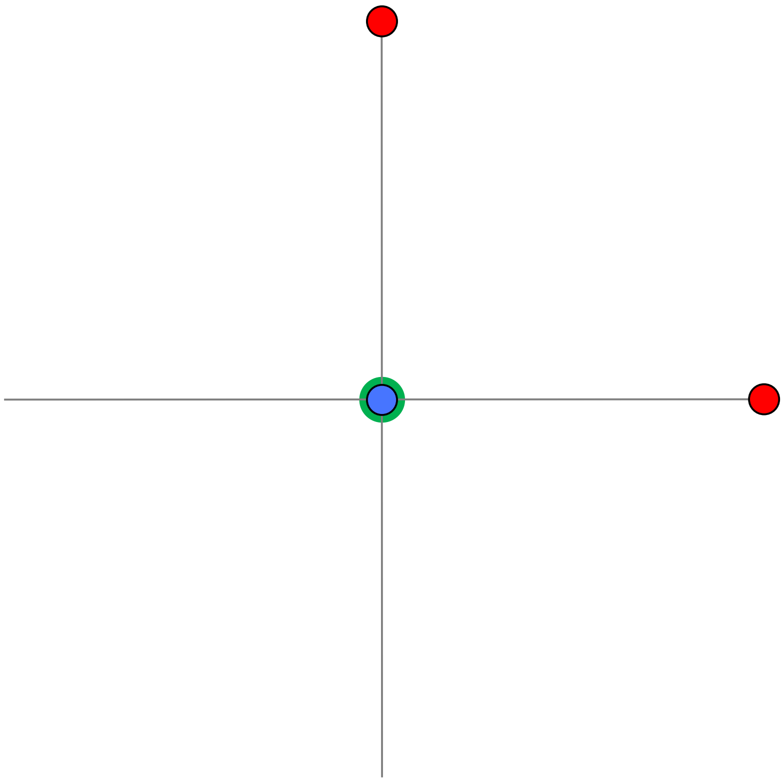
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



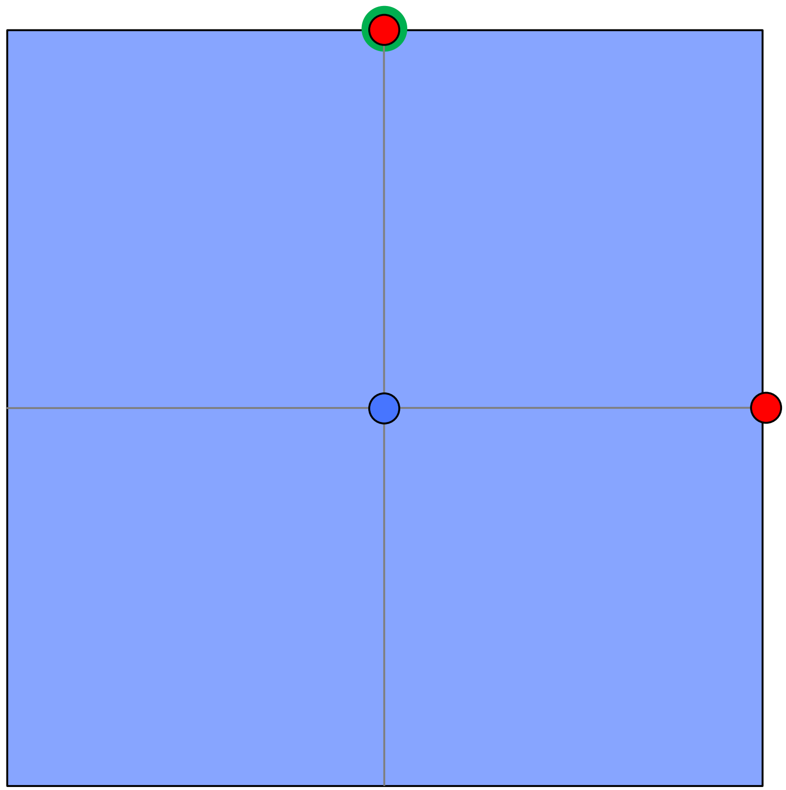
IF	$y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \le 0$
THEN	$\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



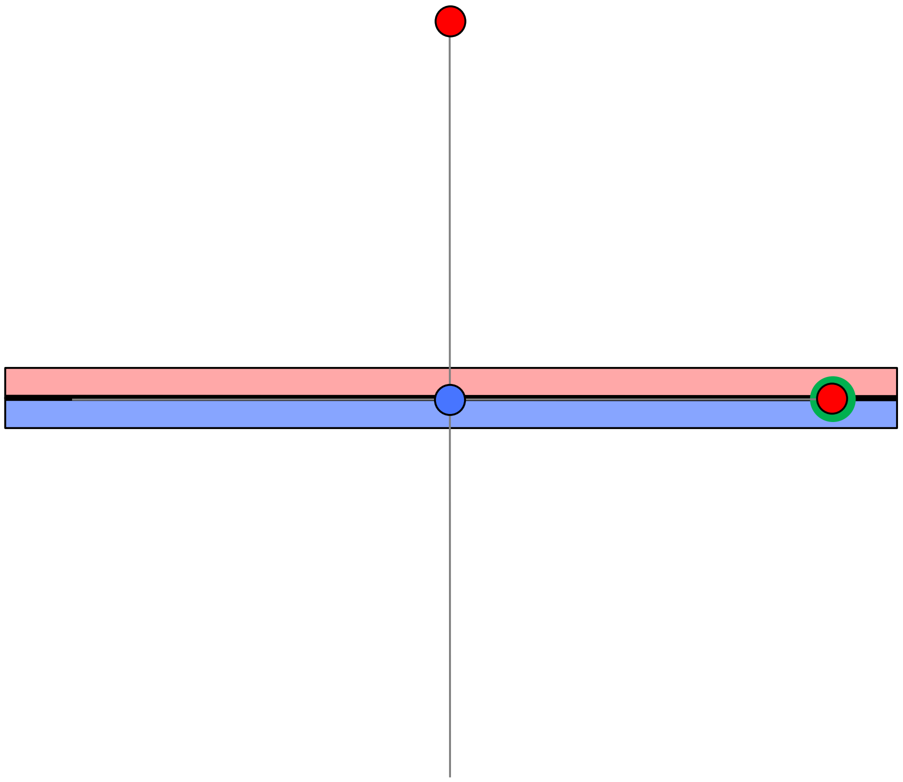
IF	$y_i f_{\boldsymbol\theta}(\mathbf{x}_i) \le 0$
THEN	$\boldsymbol\theta = \boldsymbol\theta + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol\theta}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



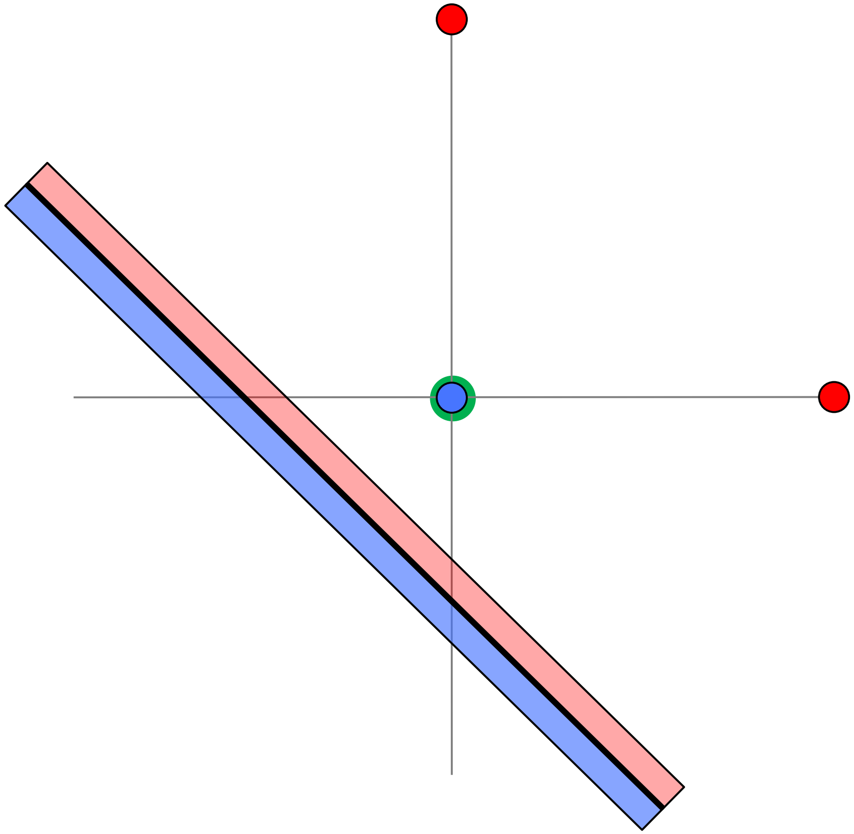
$$\texttt{IF} \quad y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$$
$$\texttt{THEN} \quad \boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



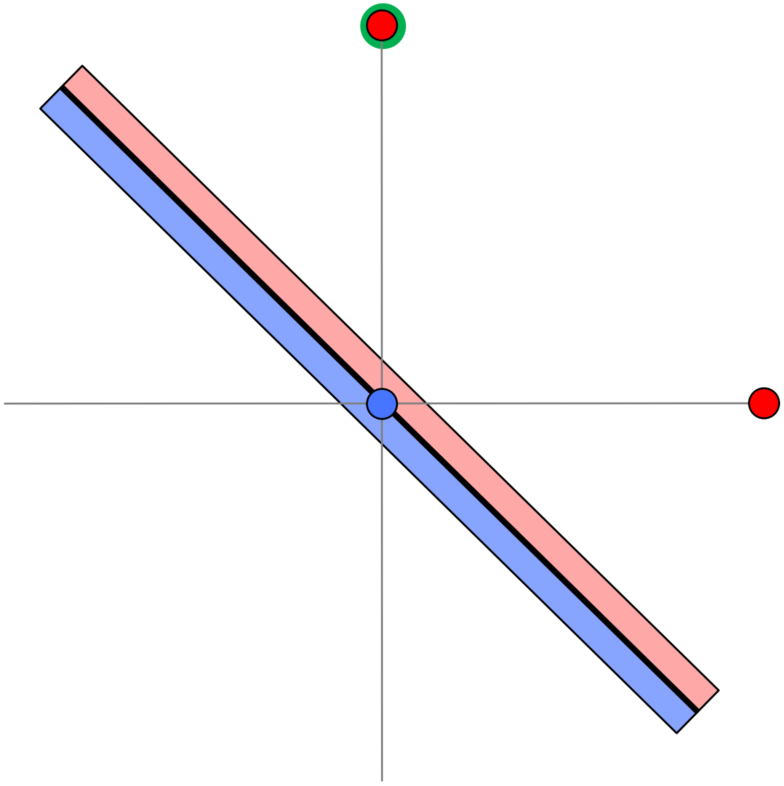
IF	$y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$
THEN	$\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



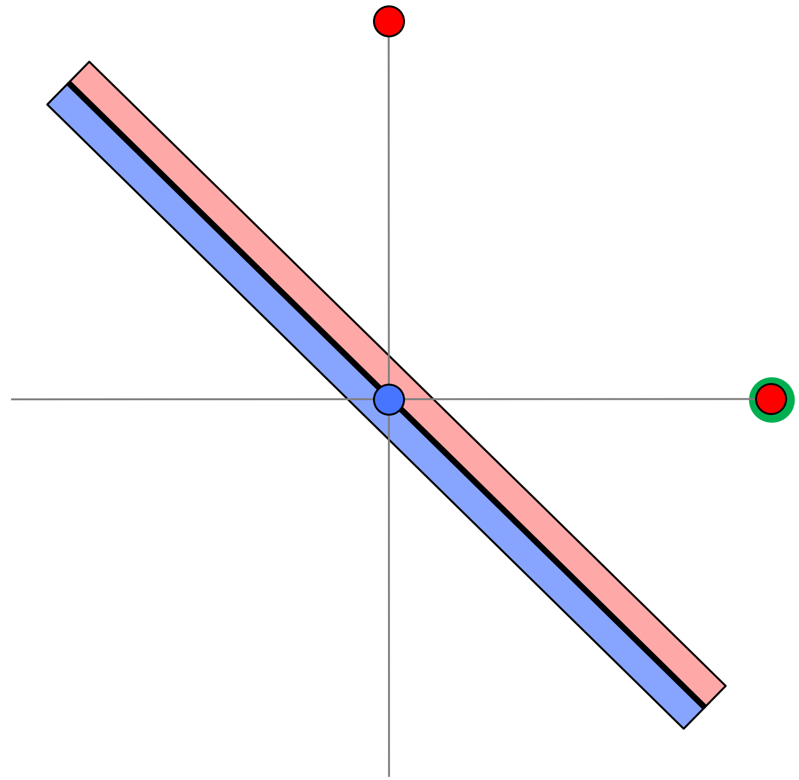
IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \le 0$
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$

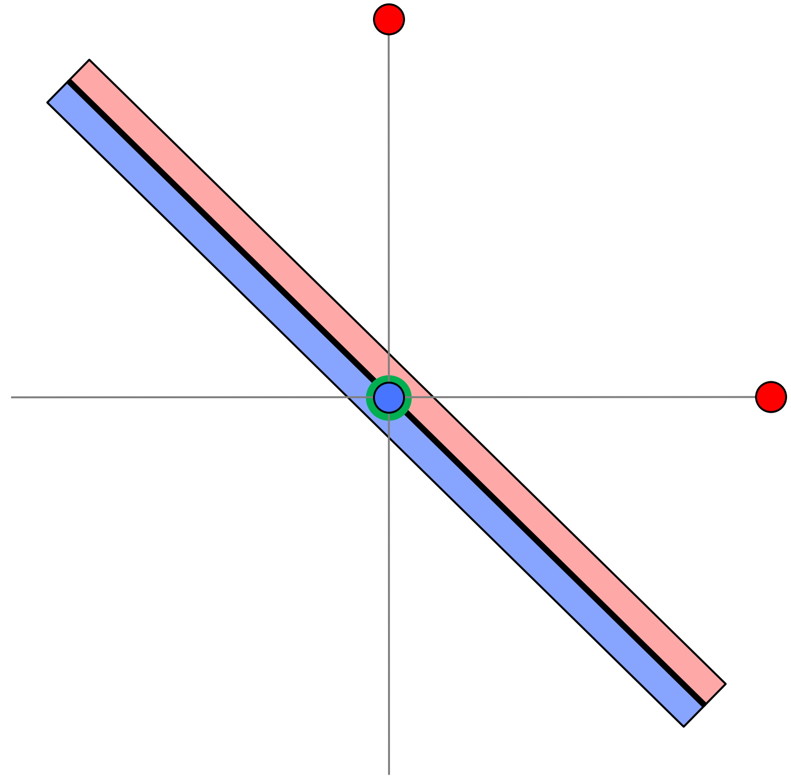
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



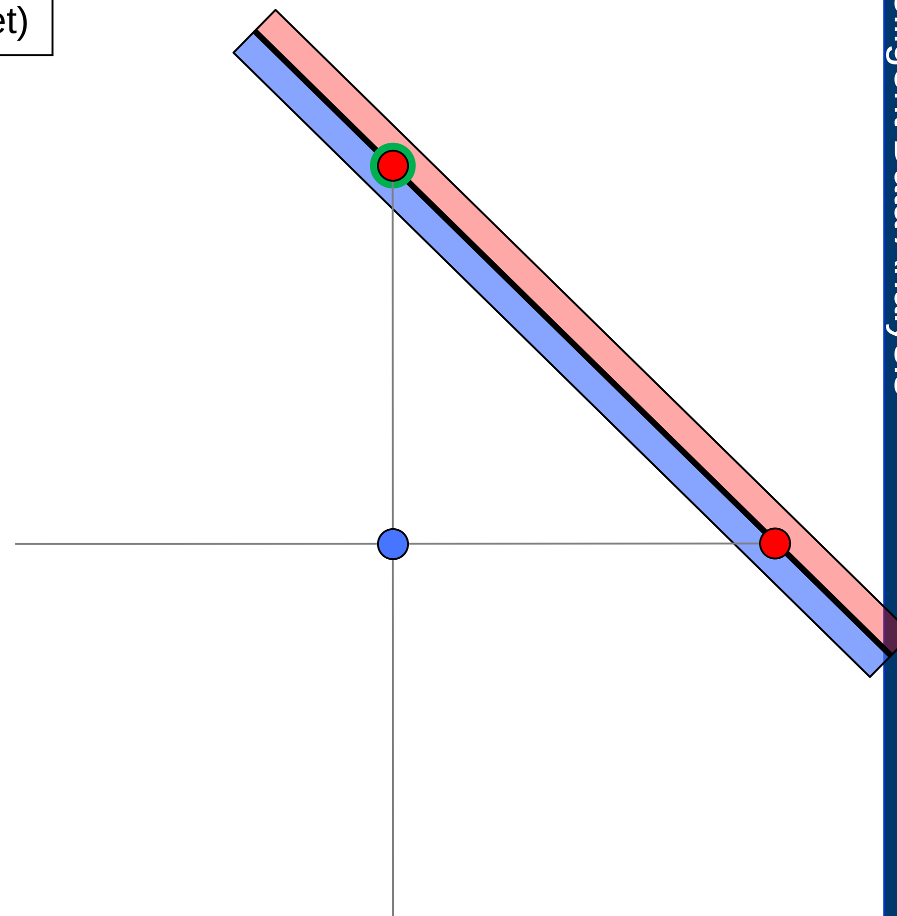
IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	0
2	0	1	1	+	1	1	-1	
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



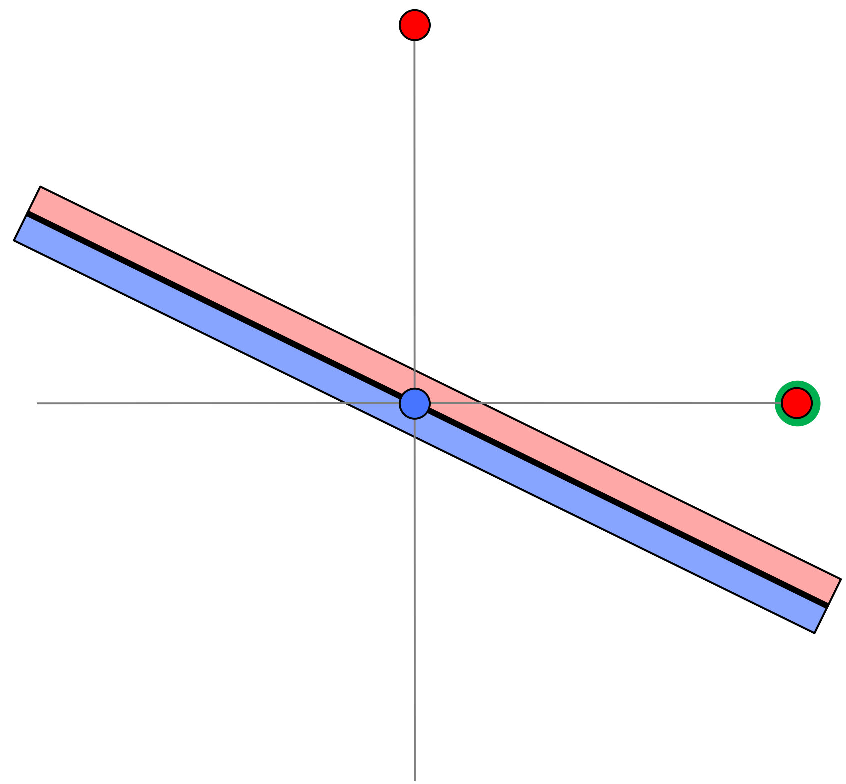
IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	0
2	0	1	1	+	1	1	-1	0
3	1	0	1	+	1	2	0	
1	0	0	1	-				
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



IF $y_i f_{\boldsymbol\theta}(\mathbf{x}_i) \le 0$

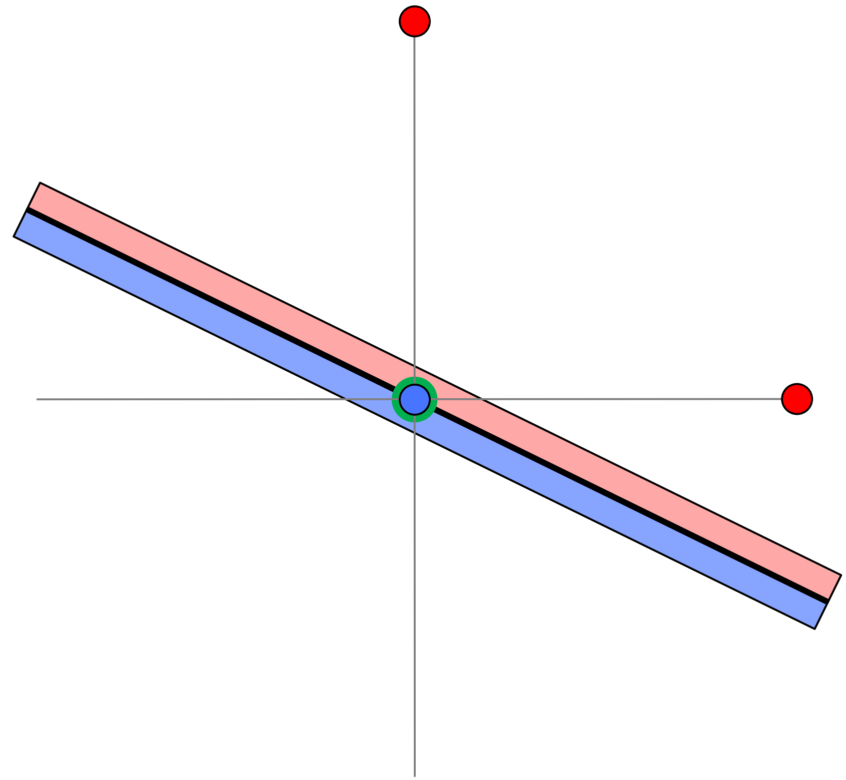
THEN $\boldsymbol\theta = \boldsymbol\theta + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol\theta}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	0
2	0	1	1	+	1	1	-1	0
3	1	0	1	+	1	2	0	1
1	0	0	1	-	1	2	0	
2	0	1	1	+				
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



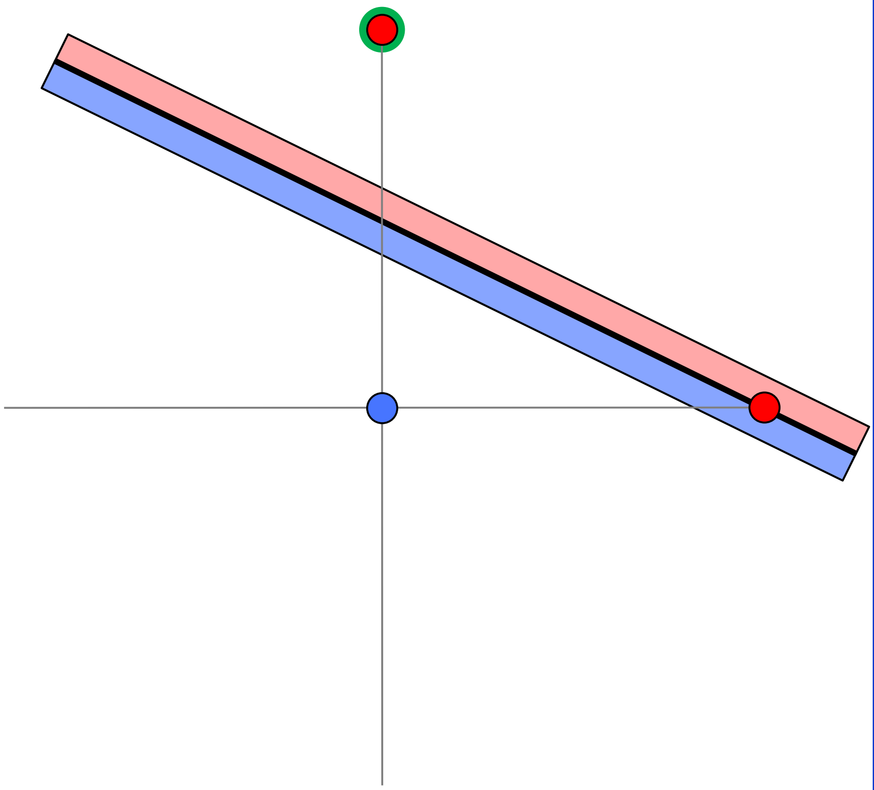
IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \le 0$
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	0
2	0	1	1	+	1	1	-1	0
3	1	0	1	+	1	2	0	1
1	0	0	1	-	1	2	0	0
2	0	1	1	+	1	2	-1	
3	1	0	1	+				
1	0	0	1	-				
2	0	1	1	+				



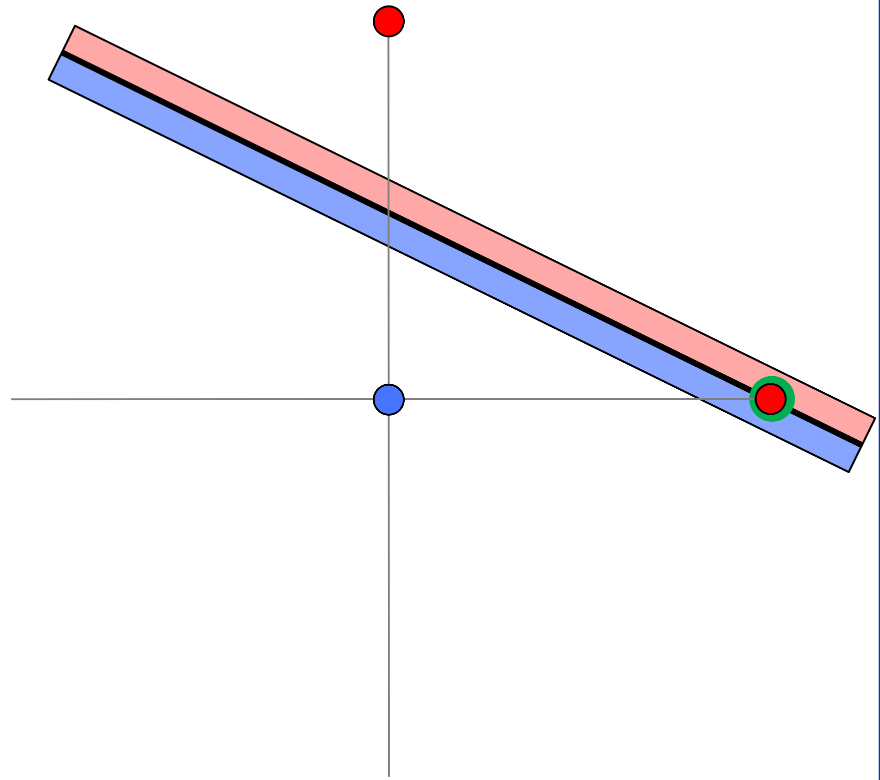
IF	$y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$
THEN	$\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	0
2	0	1	1	+	1	1	-1	0
3	1	0	1	+	1	2	0	1
1	0	0	1	-	1	2	0	0
2	0	1	1	+	1	2	-1	1
3	1	0	1	+	1	2	-1	
1	0	0	1	-				
2	0	1	1	+				



IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$

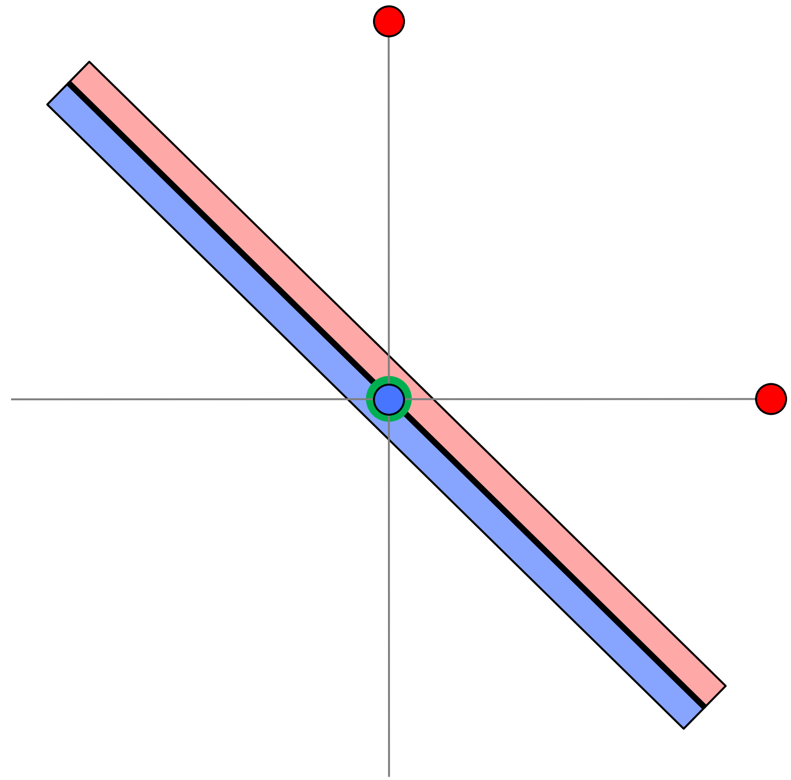
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	0
2	0	1	1	+	1	1	-1	0
3	1	0	1	+	1	2	0	1
1	0	0	1	-	1	2	0	0
2	0	1	1	+	1	2	-1	1
3	1	0	1	+	1	2	-1	0
1	0	0	1	-	2	2	0	
2	0	1	1	+				



IF $y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0$

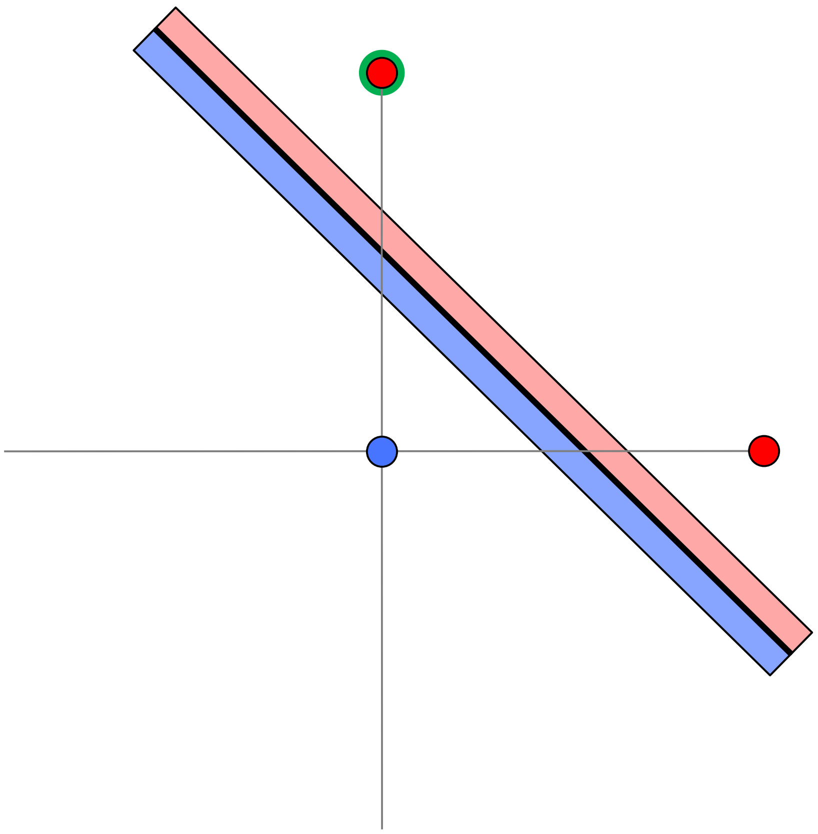
THEN $\boldsymbol{\theta} = \boldsymbol{\theta} + y_i \mathbf{x}_i$

Perceptron-Algorithm: Example

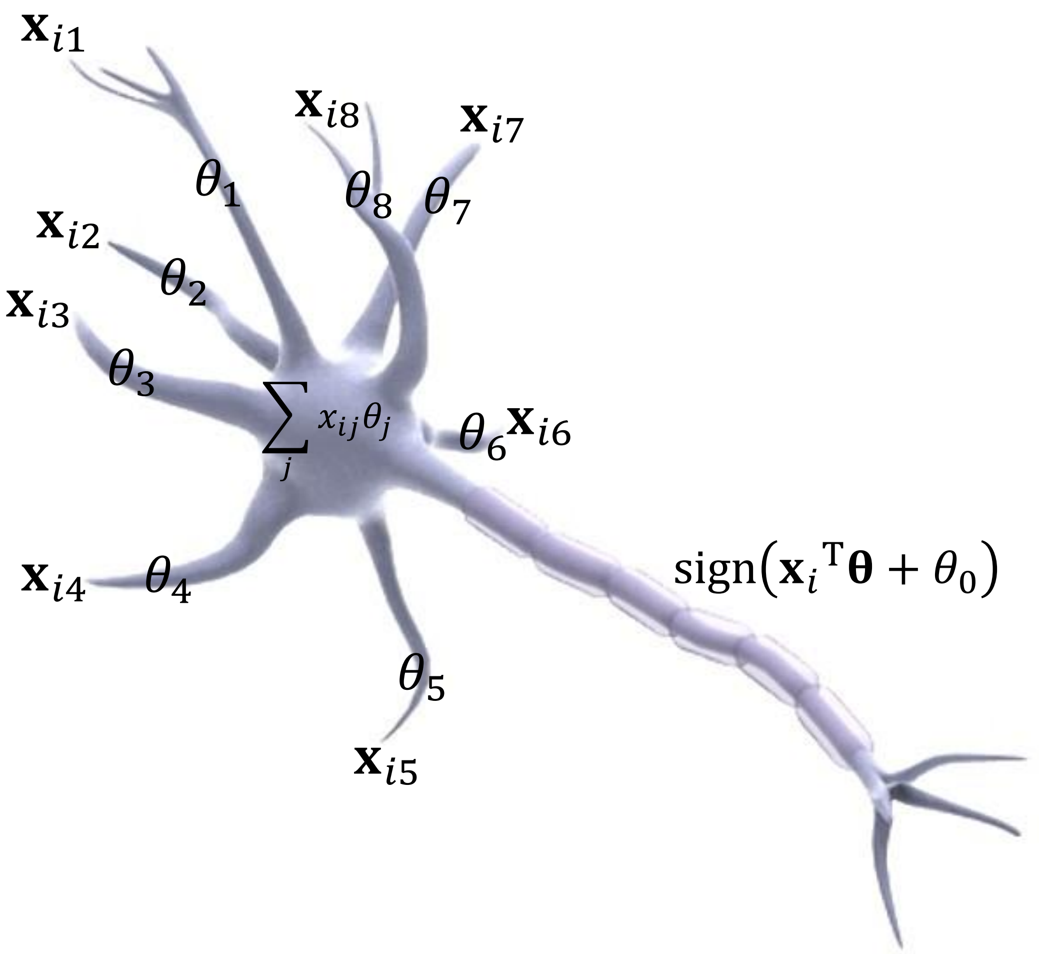
constant 1

$\theta_3 = b$ (Offset)

i	$\mathbf{x}_{i1}$	$\mathbf{x}_{i2}$	$\mathbf{x}_{i0}$	y_i	θ_1	θ_2	θ_0	$f_{\boldsymbol{\theta}}(\mathbf{x}_i)$
1	0	0	1	-	0	0	0	0
2	0	1	1	+	0	0	-1	-1
3	1	0	1	+	0	1	0	0
1	0	0	1	-	1	1	1	1
2	0	1	1	+	1	1	0	1
3	1	0	1	+	1	1	0	1
1	0	0	1	-	1	1	0	0
2	0	1	1	+	1	1	-1	0
3	1	0	1	+	1	2	0	1
1	0	0	1	-	1	2	0	0
2	0	1	1	+	1	2	-1	1
3	1	0	1	+	1	2	-1	0
1	0	0	1	-	2	2	0	0
2	0	1	1	+	2	2	-1	



ERM: Perceptron



Perceptron

- Perceptron algorithm minimizes the sum of the perceptron loss over all samples
- No regularizer
- Update rules realized as stochastic gradient search
- Fixed step size of 1; hence there is no guarantee that it will converge (unless data is separable).
- Perceptron finds (some) separating hyperplane between positive and negative samples
- Perceptron converges if such a hyperplane exists.

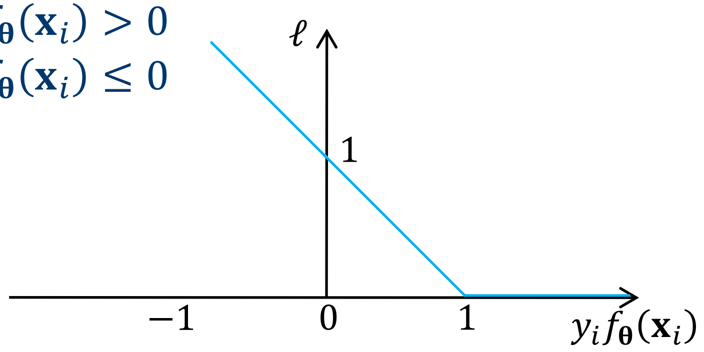
Overview

- Linear classification models
- Empirical risk minimization
  - Gradient descent method
  - Inexact line search
  - Stochastic gradient descent methods
- Loss functions and regularizers for classification
- Special cases
  - Perceptron
  - Support vector machines
- Multi-class classification

ERM: Support Vector Machine (SVM)

- Class $y \in \{-1, +1\}$
- Loss function:
  - $\ell_h(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) = \begin{cases} 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) & \text{if } 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) > 0 \\ 0 & \text{if } 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \leq 0 \end{cases}$

    $= \max\big(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\big)$



- Regularizer:
  - $\Omega_2(\boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} = \sum_{j=1}^{m} |\theta_j|^2 = \|\boldsymbol{\theta}\|_2^2$

Support Vector Machine (SVM)

- Loss function is 0, if…

$$\sum_{i=1}^{n} \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) = 0$$

$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq 1$$

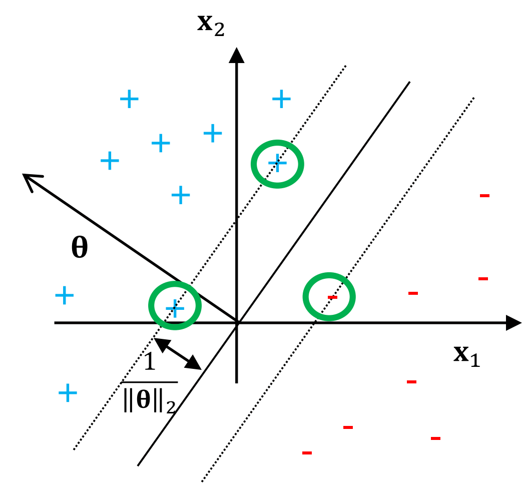
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta} \geq 1$$

$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i \underbrace{\mathbf{x}_i{}^{\mathrm{T}} \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2}}_{\mathrm{proj}_{\boldsymbol{\theta}} \mathbf{x}_i} \geq \frac{1}{\|\boldsymbol{\theta}\|_2}$$

Hessian normal form: normal vector has length 1

$$\Leftrightarrow \quad \forall_{i=1}^{n}: \mathrm{proj}_{\boldsymbol{\theta}} \mathbf{x}_i \begin{cases} \geq \dfrac{1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = +1 \\ \leq \dfrac{-1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = -1 \end{cases}$$

Support Vector Machine (SVM)



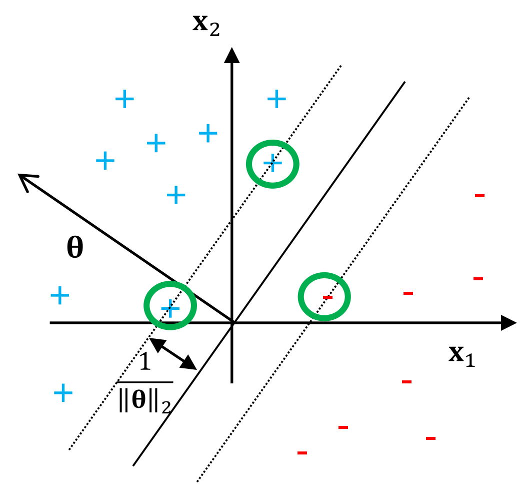
- Loss function is 0, if…

$$\sum_{i=1}^{n} \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) = 0$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i) \geq 1$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i {\mathbf{x}_i}^{\mathrm{T}} \boldsymbol{\theta} \geq 1$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: y_i {\mathbf{x}_i}^{\mathrm{T}} \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2} \geq \frac{1}{\|\boldsymbol{\theta}\|_2}$$
$$\Leftrightarrow \quad \forall_{i=1}^{n}: {\mathbf{x}_i}^{\mathrm{T}} \frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|_2} \begin{cases} \geq \frac{1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = +1 \\ \leq \frac{-1}{\|\boldsymbol{\theta}\|_2} & \text{if } y_i = -1 \end{cases}$$

Distance of closest point to plane (margin)

- For loss to be 0, all training samples must
  - lie on the correct side of separating plane,
  - and have a minimal margin (gap) of $\frac{1}{\|\boldsymbol{\theta}\|_2}$ to the plane.

Support Vector Machine (SVM)

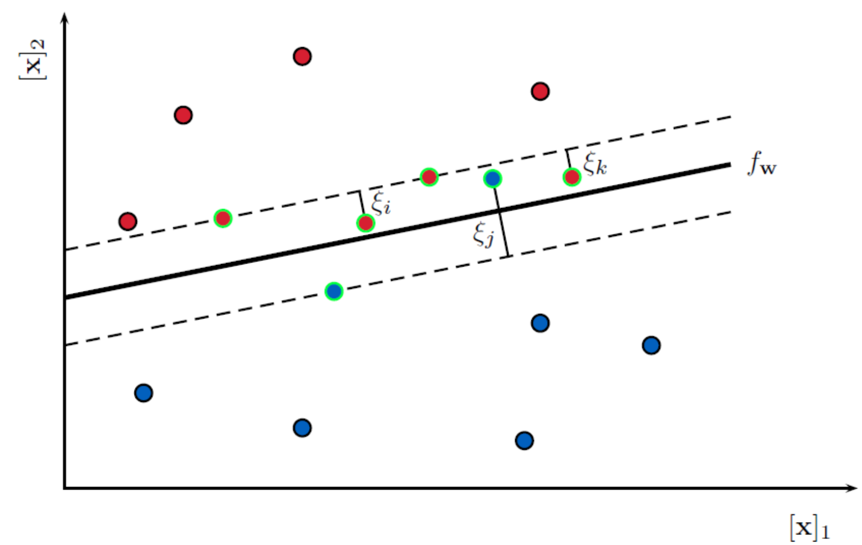


- Loss function is 0 if all training samples have a margin of at least $\frac{1}{\|\boldsymbol{\theta}\|_2}$.
  - Points that lie $\frac{1}{\|\boldsymbol{\theta}\|_2}$ from the plane are *support vectors*
- Regularizer
  - $\Omega_2(\boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} = \|\boldsymbol{\theta}\|_2^2$; is zero only if $\boldsymbol{\theta} = \mathbf{0}$
  - Minimizing $\Omega_2(\boldsymbol{\theta}) \Leftrightarrow$ maximizing margin $\frac{1}{\|\boldsymbol{\theta}\|_2}$
- SVM is also referred to as a *large margin classifier* because its optimization criterion is minimized by the plane with the largest margin from any sample.

Support Vector Machine (SVM)

- If loss function >0, some instances violate margin.
- Loss function as a sum of slack terms
  - $\sum_{i=1}^{n} \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right) = \sum_{i=1}^{n} \xi_i$
    $\xi_i = \max\left(0, 1 - y_i f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)$

Slack term or margin violation



- Points with non-zero slack are *support vectors*

Support Vector Machine (SVM)

- Minimize hinge loss and L2-norm $||\boldsymbol{\theta}||_2^2 = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}$ of the parameter vector.
- Hinge loss is positive for a sample if the sample has a distance (margin) of less than $\frac{1}{||\boldsymbol{\theta}||_2}$ to the separating hyperplane.
- SVM thereby finds the hyperplane with the greatest margin that separates the most possible samples. It trades off between
  - The size of the margin $\frac{1}{||\boldsymbol{\theta}||_2}$
  - And the sum of the slack errors $\sum_{i=1}^{n} \xi_i$

Support Vector Machine (SVM)

- Linear classification model: minimize

$$\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\max\left(0, 1 - y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta}\right) + \frac{\lambda}{n} \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta} \right]$$

- Gradient:

$$\boldsymbol{\nabla} \boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \boldsymbol{\nabla}_{\mathbf{x}_i} \boldsymbol{L}(\boldsymbol{\theta})$$

- Stochastic gradient for $\mathbf{x}_i$:

$$\boldsymbol{\nabla}_{\mathbf{x}_i} \boldsymbol{L}(\boldsymbol{\theta}) = \left\{ \right.$$

Support Vector Machine (SVM)

- Linear classification model: minimize

$$\boldsymbol{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left[\max\left(0, 1 - y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta}\right) + \frac{\lambda}{n} \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta} \right]$$

- Gradient:

$$\boldsymbol{\nabla L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \boldsymbol{\nabla}_{\mathbf{x}_i} \boldsymbol{L}(\boldsymbol{\theta})$$

- Stochastic gradient for $\mathbf{x}_i$:

$$\boldsymbol{\nabla}_{\mathbf{x}_i} \boldsymbol{L}(\boldsymbol{\theta}) = \begin{cases} \dfrac{2\lambda}{n} \boldsymbol{\theta} & \text{if } y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta} > 1 \\ \dfrac{2\lambda}{n} \boldsymbol{\theta} - y_i \mathbf{x}_i & \text{if } y_i \mathbf{x}_i{}^{\mathrm{T}} \boldsymbol{\theta} < 1 \end{cases}$$

Support Vector Machine (SVM)

- $\boldsymbol{L}(\boldsymbol{\theta})$ can be minimized using stochastic gradient descent method ("Pegasos")
  - Very fast, often used in practice
- $\boldsymbol{L}(\boldsymbol{\theta})$ can be minimized using gradient descent method ("Primal SVM")

Overview

- Linear classification models
- Empirical risk minimization
  - Gradient descent method
  - Inexact line search
  - Stochastic gradient descent methods
- Loss functions and regularizers for classification
- Special cases
  - Perceptron
  - Support vector machines
- Multi-class classification

Multi-Class Classification

- Motivation: we would like to extend classification to problems with more than 2 classes.
  - $Y = \{1, \ldots, k\}$

- Problem: we cannot separate k classes with a single hyperplane.

- Idea: Each class y has a separate function $f_{\boldsymbol{\theta}}(\mathbf{x}, y)$ that is used to predict how likely y is given $\mathbf{x}$.
  - Each function is modeled as linear.
  - We predict class y with the highest scoring function for $\mathbf{x}$.

Multi-Class Classification

- Decision functions:

$$f_{\boldsymbol{\theta}}(\mathbf{x}, y) = \mathbf{x}^{\mathrm{T}} \boldsymbol{\theta}^{y}$$

- Classifier:

$$y_{\boldsymbol{\theta}}(\mathbf{x}) = \underset{y \in Y}{\operatorname{argmax}} f_{\boldsymbol{\theta}}(\mathbf{x}, y)$$

- Model parameters:

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}^{1} \\ \vdots \\ \boldsymbol{\theta}^{k} \end{pmatrix}$$

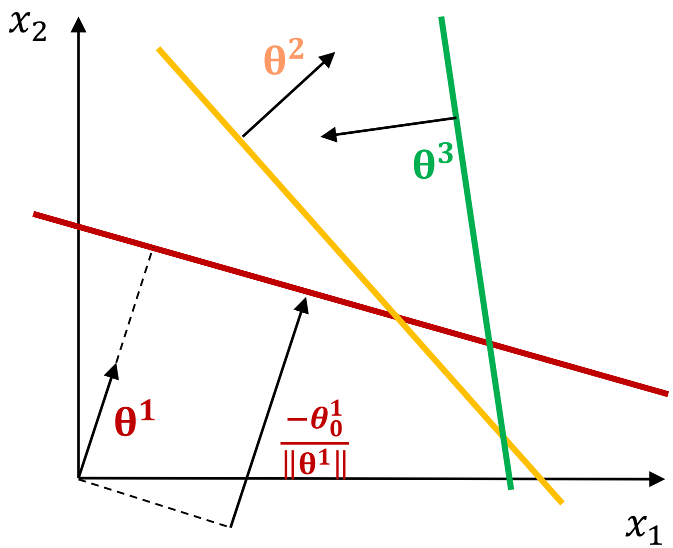
Multi-Class Classification

- Decision functions:

$$f_{\boldsymbol{\theta}}(\mathbf{x}, y) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\theta}^{y}$$

- Classifier:

$$y_{\boldsymbol{\theta}}(\mathbf{x}) = \underset{y \in Y}{\operatorname{argmax}}\, f_{\boldsymbol{\theta}}(\mathbf{x}, y)$$



Linear Classification Methods

- Linear hyperplane separates classes.
- Empirical risk minimization
  - Gradient descent method
  - Inexact line search
  - Stochastic gradient descent methods
- Perceptron
  - Stochastic gradient, perceptron loss, no regularizer
- Support vector machines
  - Gradient or stochastic gradient, hinge loss, L2-regularizer.
  - Maximizes margin between instances and plane.
- Multi-class classification: multiple planes.