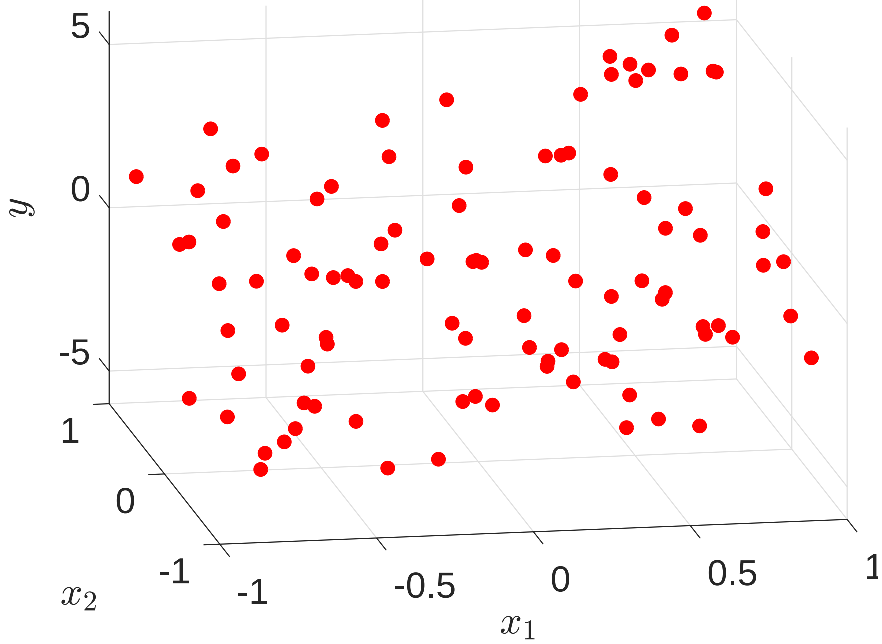
Linear Methods for Regression and Classification

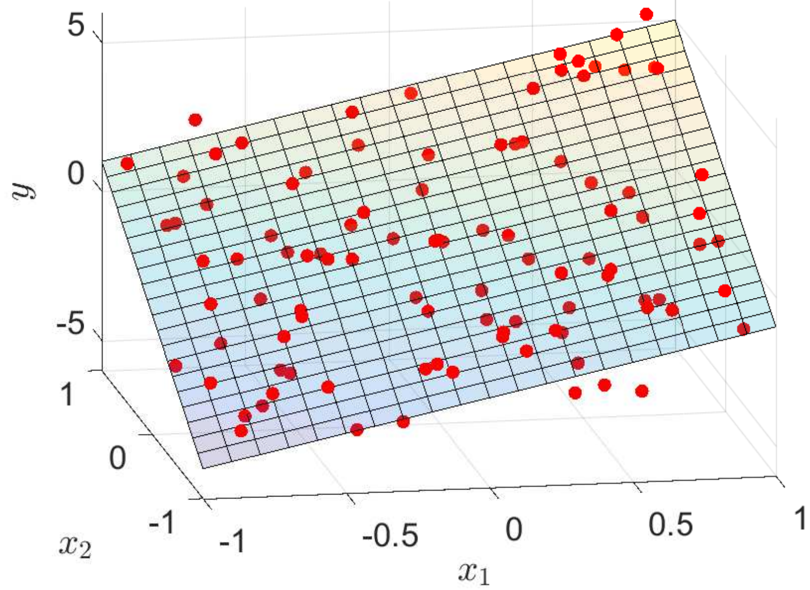
Petr Pošík

Czech Technical University in Prague
Faculty of Electrical Engineering
Dept. of Cybernetics

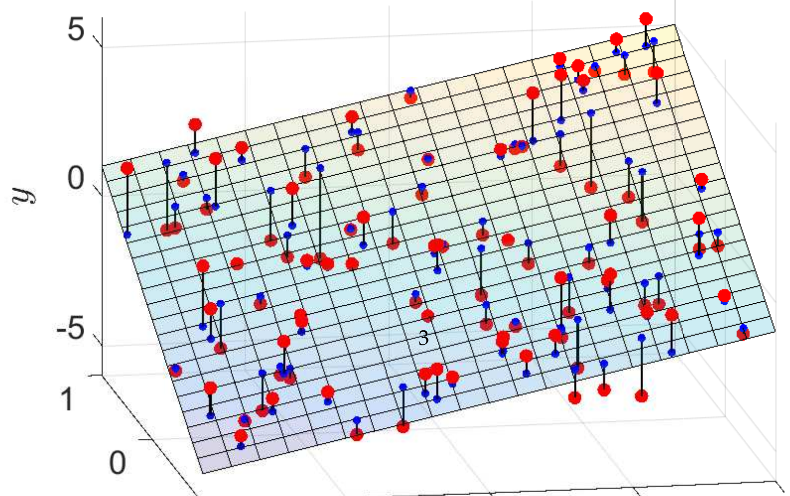
Linear regression: Illustration



Given a dataset of input vectors $x^{(i)}$ and the respective values of output variable $y^{(i)}$ ...



... we would like to find a linear model of this dataset ...



Linear regression

Regression task is a supervised learning task, i.e.

- a training (multi)set $T = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(|T|)}, y^{(|T|)})\}$ is available, where
- the labels $y^{(i)}$ are *quantitative*, often *continuous* (as opposed to classification tasks where $y^{(i)}$ are nominal).
- Its purpose is to model the relationship between independent variables (inputs) $\boldsymbol{x} = (x_1, \ldots, x_D)$ and the dependent variable (output) y.

Linear regression is a particular regression model which assumes (and learns) linear relationship between the inputs and the output:

$$\widehat{y} = h(\boldsymbol{x}) = w_0 + w_1 x_1 + \ldots + w_D x_D = w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle = w_0 + \boldsymbol{x}\boldsymbol{w}^T,$$

where

- $\widehat{y}$ is the model *prediction* (*estimate* of the true value y),
- $h(\boldsymbol{x})$ is the linear model (a *hypothesis*),
- $w_0, \ldots, w_D$ are the coefficients of the linear function, w_0 is the *bias*, organized in a row vector $\boldsymbol{w}$,
- $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ is a *dot product* of vectors $\boldsymbol{w}$ and $\boldsymbol{x}$ (scalar product),
- which can be also computed as a matrix product $\boldsymbol{x}\boldsymbol{w}^T$ if $\boldsymbol{w}$ and $\boldsymbol{x}$ are row vectors.

Notation remarks

Homogeneous coordinates: If we add "1" as the first element of $\boldsymbol{x}$ so that $\boldsymbol{x} = (1, x_1, \ldots, x_D)$, then we can write the linear model in an even simpler form (without the explicit bias term):

$$\widehat{y} = h(\boldsymbol{x}) = w_0 \cdot 1 + w_1 x_1 + \ldots + w_D x_D = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \boldsymbol{x}\boldsymbol{w}^T.$$

Matrix notation: If we organize the data into matrix $\boldsymbol{X}$ and vector $\boldsymbol{y}$, such that

$$\boldsymbol{X} = \begin{pmatrix} 1 & \boldsymbol{x}^{(1)} \\ \vdots & \vdots \\ 1 & \boldsymbol{x}^{(|T|)} \end{pmatrix} \quad \text{and} \quad \boldsymbol{y} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(|T|)} \end{pmatrix},$$

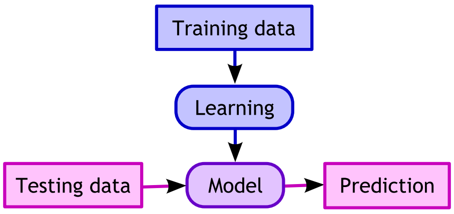
and similarly with $\widehat{\boldsymbol{y}}$, then we can write a batch computation of predictions for all data in $\boldsymbol{X}$ as

$$\widehat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{w}^T.$$

Two operation modes

Any ML model has 2 operation modes:

1. learning (training, fitting) and
2. application (testing, making predictions).



The model h can be viewed as a function of 2 variables: $h(\boldsymbol{x}, \boldsymbol{w})$.

Model application: If the model is given ($\boldsymbol{w}$ is fixed), we can manipulate $\boldsymbol{x}$ to make predictions:

$$\widehat{y} = h(\boldsymbol{x}, \boldsymbol{w}) = h_{\boldsymbol{w}}(\boldsymbol{x}).$$

Model learning: If the data is given (T is fixed), we can manipulate the model parameters $\boldsymbol{w}$ to fit the model to the data:

$$\boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} J(\boldsymbol{w}, T).$$

How to train the model?

Simple (univariate) linear regression

Simple (univariate) regression deals with cases where $\boldsymbol{x}^{(i)} = x^{(i)}$, i.e. the examples are described by a single feature (they are 1-dimensional).

Fitting a line to data:

- find parameters w_0, w_1 of a linear model $\hat{y} = w_0 + w_1 x$
- given a training (multi)set $T = \{(x^{(i)}, y^{(i)})\}_{i=1}^{|T|}$.

How to fit a line depending on the number of training examples $|T|$:

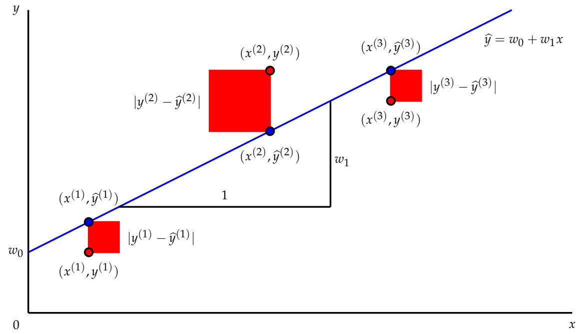
- Given a single example (1 equation, 2 parameters)
  $\Rightarrow$ infinitely many linear functions can be fitted.
- Given 2 examples (2 equations, 2 parameters)
  $\Rightarrow$ exactly 1 linear function can be fitted.
- Given 3 or more examples ($>$ 2 equations, 2 parameters)
  $\Rightarrow$ no line can be fitted with zero error
  $\Rightarrow$ a line which minimizes the "size" of error $y - \hat{y}$ can be fitted:

$$\boldsymbol{w}^* = (w_0^*, w_1^*) = \operatorname*{argmin}_{w_0, w_1} J(w_0, w_1, T).$$

The least squares method

The **least squares method (LSM)** suggests to choose such parameters $\boldsymbol{w}$ which minimize the *mean squared error* (MSE)

$$J_{MSE}(\boldsymbol{w}) = \frac{1}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{1}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)})\right)^2.$$

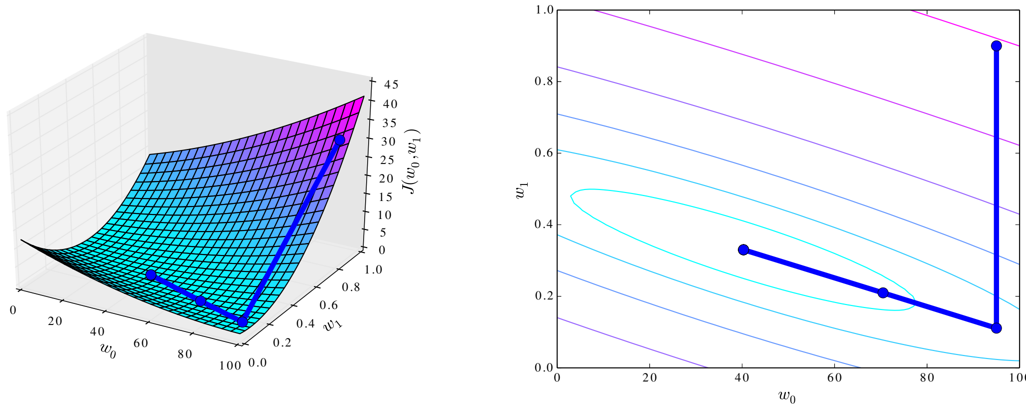


Explicit solution:

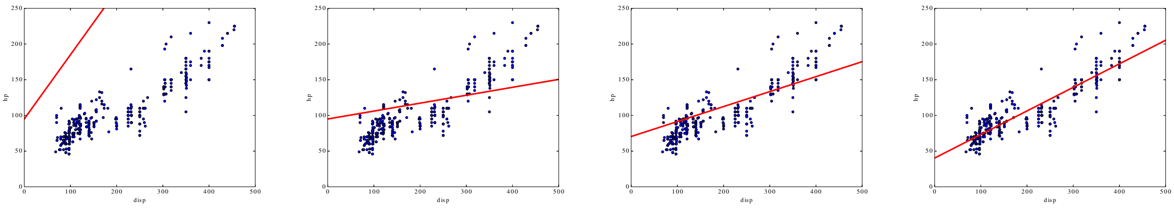
$$w_1 = \frac{\sum_{i=1}^{|T|} (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^{|T|} (x^{(i)} - \bar{x})^2} = \frac{s_{xy}}{s_x^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

Universal fitting method: minimization of cost function J

The landscape of J in the space of parameters w_0 and w_1:



Gradually better linear models found by an optimization method (BFGS):



Gradient descent algorithm

- Given a function $J(w_0, w_1)$ that should be minimized,
- start with a guess of w_0 and w_1 and
- change it, so that $J(w_0, w_1)$ decreases, i.e.
- update our current guess of w_0 and w_1 by taking a step in the direction opposite to the gradient:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla J(w_0, w_1), \text{ i.e.}$$

$$w_d \leftarrow w_d - \alpha \frac{\partial}{\partial w_d} J(w_0, w_1),$$

  where all w_is are updated simultaneously and α is a **learning rate** (step size).
- For the cost function

$$J(w_0, w_1) = \frac{1}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - h_{\boldsymbol{w}}(x^{(i)}) \right)^2 = \frac{1}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - (w_0 + w_1 x^{(i)}) \right)^2,$$

  the gradient can be computed as

$$\frac{\partial}{\partial w_0} J(w_0, w_1) = -\frac{2}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - h_{\boldsymbol{w}}(x^{(i)}) \right) = \frac{2}{|T|} \sum_{i=1}^{|T|} \left(h_{\boldsymbol{w}}(x^{(i)}) - y^{(i)} \right)$$

$$\frac{\partial}{\partial w_1} J(w_0, w_1) = -\frac{2}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - h_{\boldsymbol{w}}(x^{(i)}) \right) x^{(i)} = \frac{2}{|T|} \sum_{i=1}^{|T|} \left(h_{\boldsymbol{w}}(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

Multivariate linear regression

Multivariate linear regression deals with cases where $x^{(i)} = (x_1^{(i)}, \ldots, x_D^{(i)})$, i.e. the examples are described by more than 1 feature (they are D-dimensional).

Model fitting:

- find parameters $w = (w_1, \ldots, w_D)$ of a linear model $\hat{y} = xw^T$
- given the training (multi)set $T = \{(x^{(i)}, y^{(i)})\}_{i=1}^{|T|}$.
- The model is a *hyperplane* in the $D+1$-dimensional space.

Fitting methods:

1. Numeric optimization of $J(w, T)$:
   - Works as for simple regression, it only searches a space with more dimensions.
   - Sometimes one needs to tune some parameters of the optimization algorithm to work properly (learning rate in gradient descent, etc.).
   - May be slow (many iterations needed), but works even for very large D.
2. **Normal equation:**

$$w^* = (X^T X)^{-1} X^T y$$

   - Method to solve for the optimal w^* analytically!
   - No need to choose optimization algorithm parameters.
   - No iterations.
   - Needs to compute $(X^T X)^{-1}$, which is $O(D^3)$. Slow, or intractable, for large D.

Linear classification

Binary classification task (dichotomy)

Let's have the training dataset $T = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(|T|)}, y^{(|T|)})\}$:

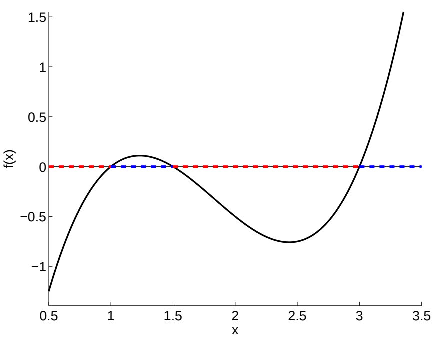
- each example is described by a vector of features $x = (x_1, \ldots, x_D)$,
- each example is labeled with the correct class $y \in \{+1, -1\}$.

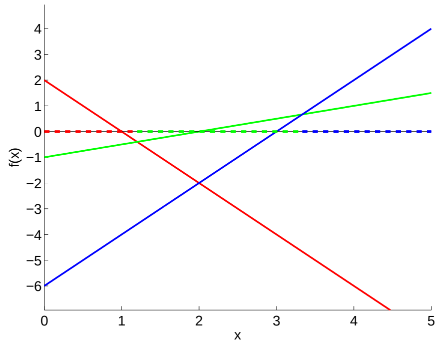
Discrimination function: a function allowing us to *decide* to which class an example x belongs.

- For 2 classes, 1 discrimination function is enough.
- Decision rule:

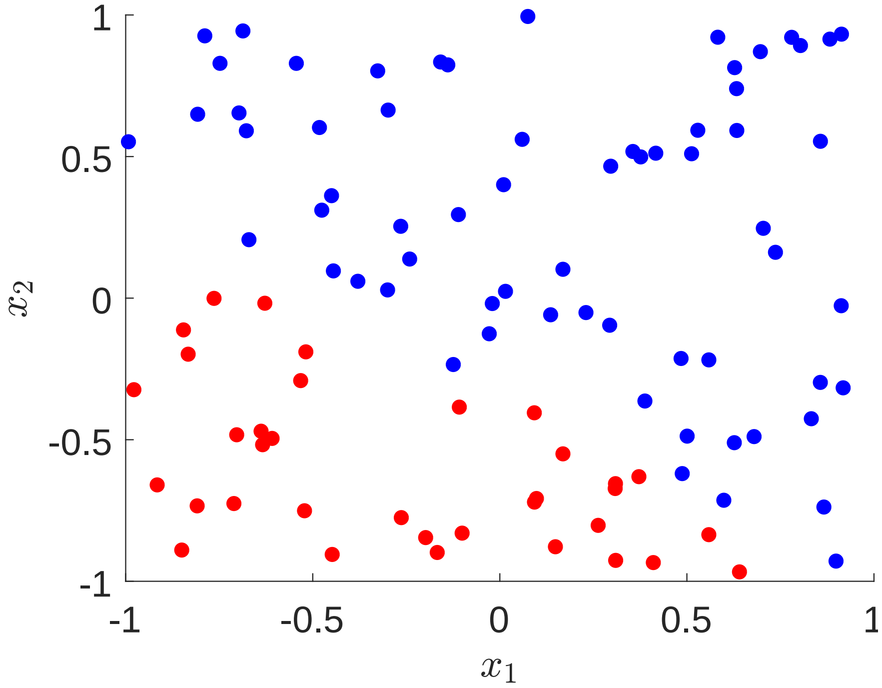
$$\left.\begin{array}{l} f(x) > 0 \iff \hat{y} = +1 \\ f(x) < 0 \iff \hat{y} = -1 \end{array}\right\} \quad \text{i.e.} \quad \hat{y} = \text{sign}(f(x))$$

- **Decision boundary:** $\{x : f(x) = 0\}$
- *Learning* then amounts to finding (parameters of) function f.

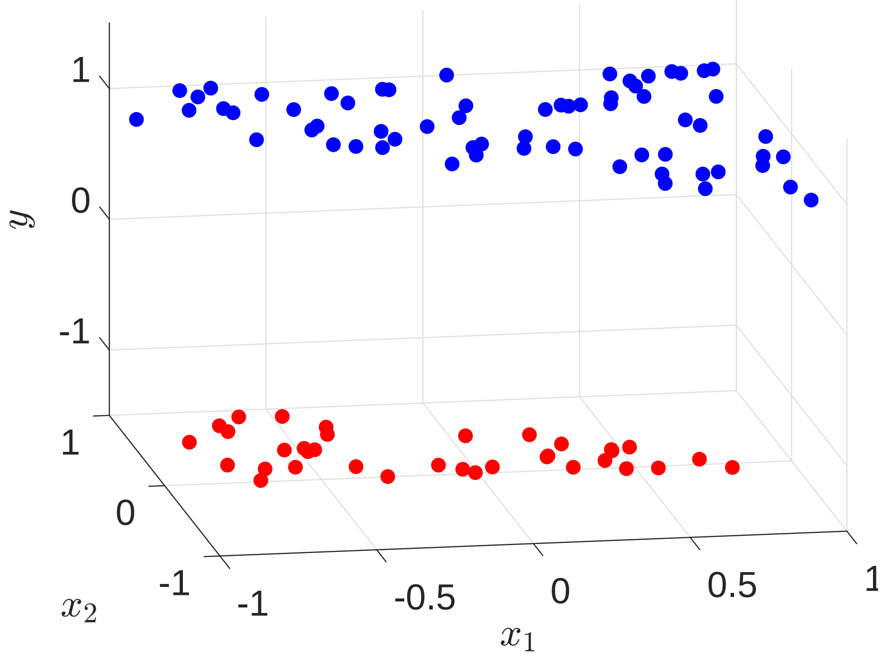




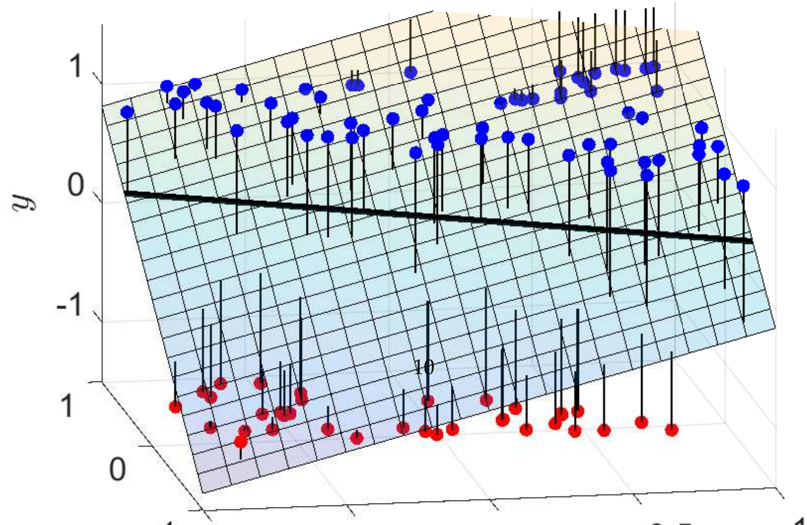
Naive approach: Illustration



Given a dataset of input vectors $x^{(i)}$ and their classes $y^{(i)}$ ...



...we shall encode the class label as $y = -1$ and $y = 1$ ...



Naive approach

Problem: Learn a linear discrimination function f from data T.

Naive solution: fit linear regression model to the data!

- Use cost function

$$J_{MSE}(\boldsymbol{w}, T) = \frac{1}{|T|}\sum_{i=1}^{|T|}\left(y^{(i)} - f(\boldsymbol{w}, \boldsymbol{x}^{(i)})\right)^2,$$

- minimize it with respect to $\boldsymbol{w}$,
- and use $\widehat{y} = \text{sign}(f(\boldsymbol{x}))$.
- Issue: Points far away from the decision boundary have *huge effect* on the model!

Better solution: fit a linear discrimination function which minimizes the number of errors!

- Cost function:

$$J_{01}(\boldsymbol{w}, T) = \frac{1}{|T|}\sum_{i=1}^{|T|}\mathbb{I}(y^{(i)} \neq \widehat{y}^{(i)}),$$

  where $\mathbb{I}$ is the indicator function: $\mathbb{I}(a)$ returns 1 iff a is True, 0 otherwise.
- The cost function is non-smooth, contains plateaus, not easy to optimize, but there are algorithms which attempt to solve it, e.g. perceptron, Kozinec's algorithm, etc.

Perceptron algorithm

Perceptron [Ros62]:

- a simple model of a neuron
- a linear classifier (in this case, a classifier with a linear discrimination function)

Algorithm 1: Perceptron algorithm

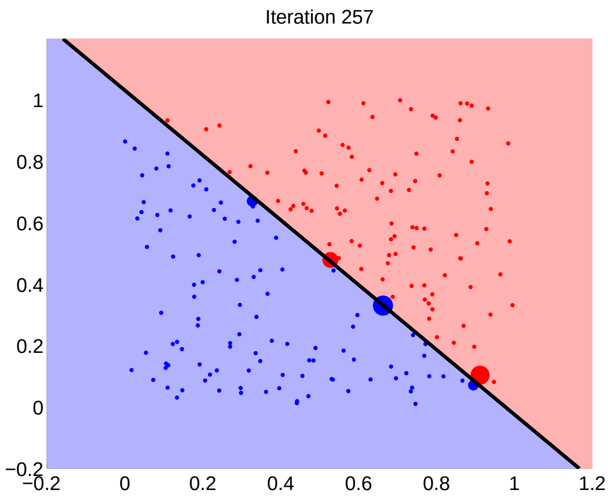
Input: Linearly separable training dataset: $\{x^{(i)}, y^{(i)}\}$, $x^{(i)} \in \mathcal{R}^{D+1}$ (homogeneous coordinates), $y^{(i)} \in \{+1, -1\}$
Output: Weight vector w such that $x^{(i)} w^T > 0$ iff $y^{(i)} = +1$ and $x^{(i)} w^T < 0$ iff $y^{(i)} = -1$

```
1 begin
2     Initialize the weight vector, e.g. w = 0.
3     Invert all examples x belonging to class -1: x^(i) = −x^(i) for all i, where y^(i) = −1.
4     Find an incorrectly classified training vector, i.e. find j such that x^(j) w^T ≤ 0, e.g. the worst classified vector: x^(j) = argmin_{x^(i)} (x^(i) w^T).
5     if all examples classified correctly then
6         Return the solution w. Terminate.
7     else
8         Update the weight vector: w = w + x^(j).
9     Go to 4.
```

Instead of using the worst classified point, the algorithm may go over the training set (several times) and use all encountered wrongly classified points to update w.

[Ros62] Frank Rosenblatt. *Principles of Neurodynamics: Perceptron and the Theory of Brain Mechanisms*. Spartan Books, Washington, D.C., 1962.

Demo: Perceptron



Features of the perceptron algorithm

Perceptron convergence theorem [Nov62]:

- Perceptron algorithm eventually finds a hyperplane that separates 2 classes of points in a finite number of steps, if such a hyperplane exists.
- If no separating hyperplane exists, the algorithm does not converge and will iterate forever.

Possible solutions:

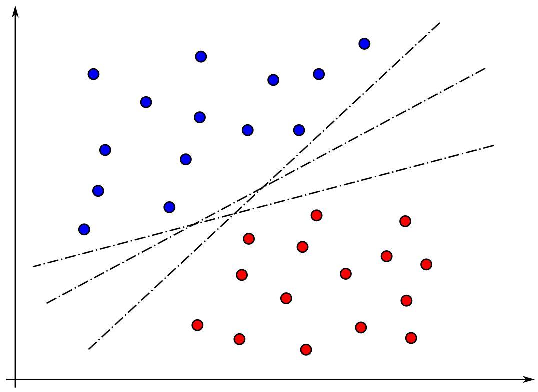
- Pocket algorithm — track the error the perceptron makes in each iteration and store the best weights found so far in a separate memory (pocket).
- Use a different learning algorithm, which finds an approximate solution, if the classes are not linearly separable.

[Nov62] Albert B. J. Novikoff. On convergence proofs for perceptrons. In *Proceedings of the Symposium on Mathematical Theory of Automata*, volume 12, Brooklyn, New York, 1962.

The hyperplane found by perceptron

The perceptron algorithm

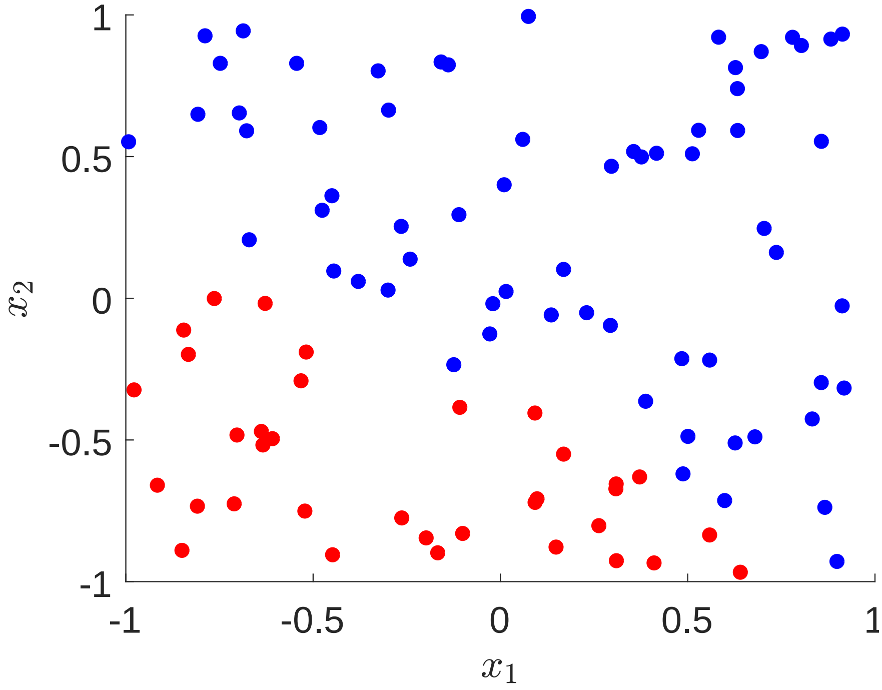
- finds a separating hyperplane, if it exists;
- but if a single separating hyperplane exists, then there are infinitely many (equally good?) separating hyperplanes.



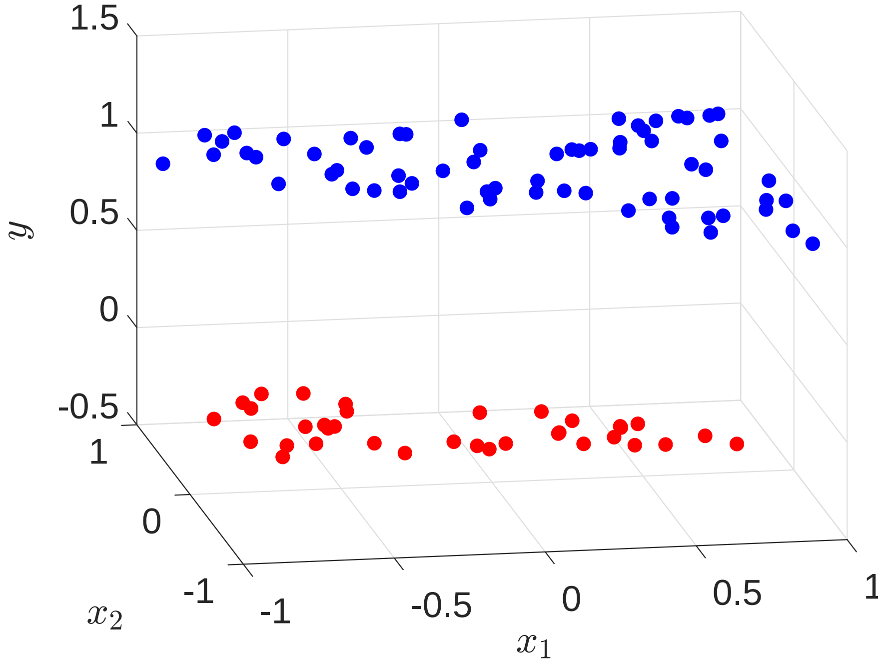
- and perceptron finds *any* of them!

Which separating hyperplane is the optimal one? What does "optimal" actually mean?

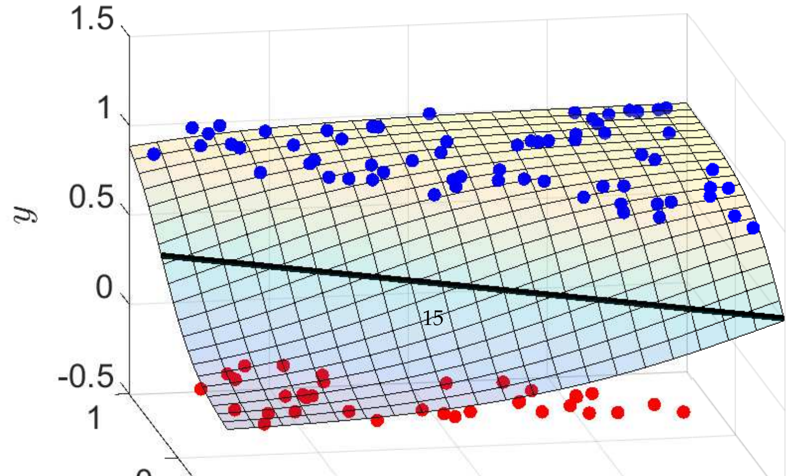
Logistic regression: Illustration



Given a dataset of input vectors $x^{(i)}$ and their classes $y^{(i)}$ …



…we shall encode the class label as $y = 0$ and $y = 1$ …



Logistic regression model

Problem: Learn a binary **classifier** for the dataset $T = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}$, where $y^{(i)} \in \{0, 1\}$.[a]

To reiterate: when using linear regression, the examples far from the decision boundary have a huge impact on f. How to limit their influence?

Logistic regression uses a discrimination function which is a nonlinear transformation of the values of a linear function

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = g(\boldsymbol{x}\boldsymbol{w}^T) = \frac{1}{1 + e^{-\boldsymbol{x}\boldsymbol{w}^T}},$$

where $g(z) = \frac{1}{1 + e^{-z}}$ is the **sigmoid** function (a.k.a **logistic** function).

Interpretation of the model:

- $f_{\boldsymbol{w}}(\boldsymbol{x})$ is interpretted as an estimate of the probability that $\boldsymbol{x}$ belongs to class 1.
- The **decision boundary** is defined using a different level-set: $\{\boldsymbol{x} : f_{\boldsymbol{w}}(\boldsymbol{x}) = 0.5\}$.
- Logistic *regression* is a *classification model!*
- The discrimination function $f_{\boldsymbol{w}}(\boldsymbol{x})$ itself is not linear anymore; but the *decision boundary is still linear!*
- Thanks to the sigmoidal transformation, logistic regression is much less influenced by examples far from the decision boundary!

[a] Previously, we have used $y^{(i)} \in \{-1, +1\}$, but the values can be chosen arbitrarily, and $\{0, 1\}$ is convenient for logistic regression.

Cost function

To train the logistic regression model, one can use the J_{MSE} criterion:

$$J(\boldsymbol{w}, T) = \frac{1}{|T|} \sum_{i=1}^{|T|} \left(y^{(i)} - f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) \right)^2.$$

However, this results in a non-convex multimodal landscape which is hard to optimize.

Logistic regression uses a modified cost function (sometimes called *cross-entropy*):

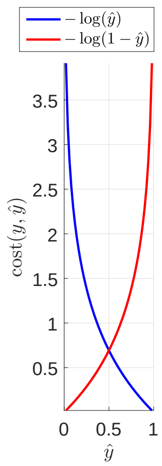
$$J(\boldsymbol{w}, T) = \frac{1}{|T|} \sum_{i=1}^{|T|} \text{cost}(y^{(i)}, f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)})), \text{ where}$$

$$\text{cost}(y, \widehat{y}) = \begin{cases} -\log(\widehat{y}) & \text{if } y = 1 \\ -\log(1-\widehat{y}) & \text{if } y = 0 \end{cases},$$

which can be rewritten in a single expression as

$$\text{cost}(y, \widehat{y}) = -y \cdot \log(\widehat{y}) - (1-y) \cdot \log(1-\widehat{y}).$$

Such a cost function is simpler to optimize for numerical solvers.

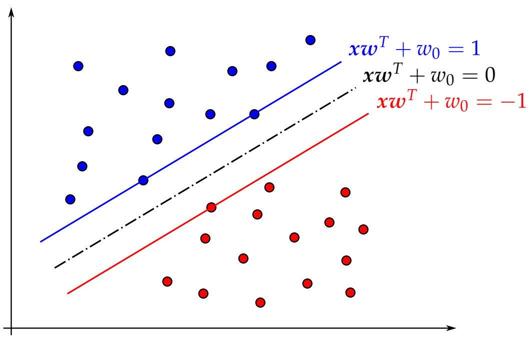


Optimal separating hyperplane (separable case)

Margin (cz:odstup):

- "The width of the band in which the decision boundary can move (in the direction of its normal vector) without touching any data point."

Maximum margin linear classifier



Plus 1 level: $\{x : \boldsymbol{xw}^T + w_0 = 1\}$
Minus 1 level: $\{x : \boldsymbol{xw}^T + w_0 = -1\}$
Decision boundary
(separating hyperplane): $\{x : \boldsymbol{xw}^T + w_0 = 0\}$

Support vectors:

- Data points $\boldsymbol{x}$ lying at the plus 1 level or minus 1 level.
- Only these points influence the decision boundary!

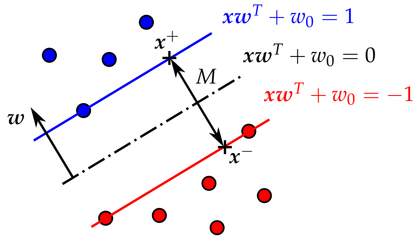
Why we would like to maximize the margin?

- Intuitively, it is safe.
- If we make a small error in estimating the boundary, the classification will likely stay correct.
- The model is invariant with respect to the training set changes, except the changes of support vectors.
- There are sound theoretical results that having a maximum margin classifier is good.
- Maximal margin works well in practice.

Margin size

How to compute the margin M given $\boldsymbol{w} = (w_1, \ldots, w_D)$, w_0 of certain sep. hyperplane?

- Let's choose two points $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$, lying in the plus 1 level and minus 1 level, respectively.
- Let's compute the margin M as their distance.



We know that:

$$\boldsymbol{x}^+\boldsymbol{w}^T + w_0 = 1$$
$$\boldsymbol{x}^-\boldsymbol{w}^T + w_0 = -1$$
$$\boldsymbol{x}^- + \lambda\boldsymbol{w} = \boldsymbol{x}^+$$

And we can derive:

$$(\boldsymbol{x}^+ - \boldsymbol{x}^-)\boldsymbol{w}^T = 2$$
$$(\boldsymbol{x}^- + \lambda\boldsymbol{w} - \boldsymbol{x}^-)\boldsymbol{w}^T = 2$$
$$\lambda\boldsymbol{w}\boldsymbol{w}^T = 2$$
$$\lambda = \frac{2}{\boldsymbol{w}\boldsymbol{w}^T} = \frac{2}{\|\boldsymbol{w}\|^2}$$

Thus the margin size is

$$M = \|\boldsymbol{x}^+ - \boldsymbol{x}^-\| = \|\lambda\boldsymbol{w}\| = \lambda\|\boldsymbol{w}\| = \frac{2}{\|\boldsymbol{w}\|^2}\|\boldsymbol{w}\| = \frac{2}{\|\boldsymbol{w}\|}$$

Optimal separating hyperplane learning

We want to maximize margin $M = \frac{2}{\|\boldsymbol{w}\|}$ subject to the constraints ensuring correct classification of the training set T. This optimization problem can be formulated as a *quadratic programming* (QP) task.

- Primary QP task:

$$\text{minimize } \frac{1}{2}\boldsymbol{w}\boldsymbol{w}^T \text{ with respect to } w_0, \ldots, w_D$$
$$\text{subject to } y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) \geq 1 \quad \forall i \in 1, \ldots, |T|.$$

- Dual QP task:

$$\text{maximize } \sum_{i=1}^{|T|}\alpha_i - \sum_{i=1}^{|T|}\sum_{j=1}^{|T|}\alpha_i\alpha_j y^{(i)}y^{(j)}\boldsymbol{x}^{(i)}\boldsymbol{x}^{(j)T} \text{ with respect to } \alpha_1, \ldots, \alpha_{|T|}$$
$$\text{subject to } \alpha_i \geq 0$$
$$\text{and } \sum_{i=1}^{|T|}\alpha_i y^{(i)} = 0.$$

- From the solution of the dual task, we can compute the solution of the primal task:

$$\boldsymbol{w} = \sum_{i=1}^{|T|}\alpha_i y^{(i)}\boldsymbol{x}^{(i)}, \qquad w_0 = y^{(k)} - \boldsymbol{x}^{(k)}\boldsymbol{w}^T,$$

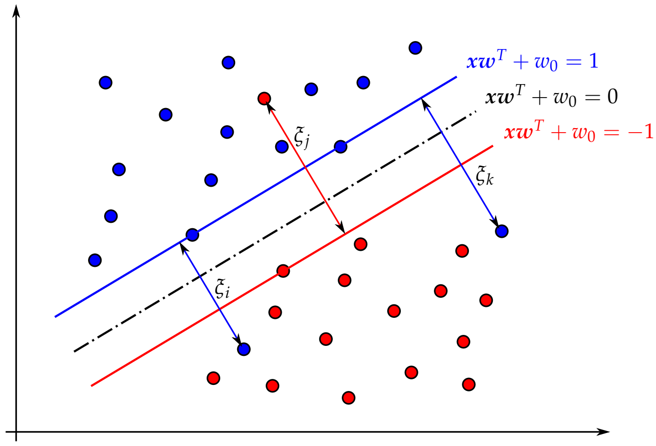
where $(\boldsymbol{x}^{(k)}, y^{(k)})$ is any *support vector*, i.e. $\alpha_k > 0$.

Non-separable case

Soft margin: Allows for incorrect classification of some data points.

Slack variables ξ_i: The shortest distances of data points to their "correct place":

- 0 for correctly classified data "outside the margin",
- positive for incorrectly classified data and data "inside the margin".



Optimal separating hyperplane learning for non-separable data

- Primary QP task with slack variables:

$$\text{minimize } \left(\frac{1}{2}ww^T + C\sum_{i=1}^{|T|}\xi_i\right) \text{ with respect to } w_0, \ldots, w_D, \xi_1, \ldots, \xi_{|T|}$$

$$\text{subject to } y^{(i)}(x^{(i)}w^T + w_0) \geq 1 - \xi_i \quad \forall i \in 1, \ldots, |T|,$$

$$\text{and } \xi_i \geq 0 \ \ \forall i \in 1, \ldots, |T|.$$

- Dual QP task:

$$\text{maximize } \sum_{i=1}^{|T|}\alpha_i - \sum_{i=1}^{|T|}\sum_{j=1}^{|T|}\alpha_i\alpha_j y^{(i)}y^{(j)}x^{(i)}x^{(j)T} \text{ with respect to } \alpha_1, \ldots, \alpha_{|T|}, \mu_1, \ldots, \mu_{|T|},$$

$$\text{subject to } \alpha_i \geq 0, \ \mu_i \geq 0, \ \alpha_i + \mu_i = C,$$

$$\text{and } \sum_{i=1}^{|T|}\alpha_i y^{(i)} = 0.$$

- Variables α_i are more constrained than in the separable case, but the solution is the same:

$$w = \sum_{i=1}^{|T|}\alpha_i y^{(i)}x^{(i)}, \qquad w_0 = y^{(k)} - x^{(k)}w^T,$$

where $(x^{(k)}, y^{(k)})$ is any *support vector*, i.e. $\alpha_k > 0$.

Lagrange function

Primary QP task:

$$(\boldsymbol{w}^*, w_0^*, \xi^*) = \arg\min_{\boldsymbol{w}, w_0, \xi} \left(\frac{1}{2}\boldsymbol{w}\boldsymbol{w}^T + C\sum_{i=1}^{|T|} \xi_i \right)$$

with constraints $\forall i, i = 1, \ldots, |T|$

$$y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 + \xi_i \geq 0$$
$$\xi_i \geq 0$$

Method of Lagrange multipliers

- replaces the search for stationary points of function of D variables with K constraints by the search for stationary points of unconstrained function of $D + K$ variables;
- creates a new variable — *Langrange multiplier* — for each constraint and defines a new function, *Langrangian*, formed by the original function, constraints and multipliers.

$$L(\boldsymbol{w}, w_0, \xi_i, \alpha_i, \mu_i) = \frac{1}{2}|\boldsymbol{w}|^2 + C\sum_{i=1}^{|T|} \xi_i - \sum_{i=1}^{|T|} \alpha_i \{y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 + \xi_i\} - \sum_{i=1}^{|T|} \mu_i \xi_i$$

where

- $\alpha_i \geq 0$ are Lagrange multipliers for constraints ensuring the correct classification of points, and
- $\mu_i \geq 0$ are Lagrange multipliers for constraint on positivity of ξ_i.

The Lagrangian must be minimized w.r.t. the *primary variables* $\boldsymbol{w}$*,* w_0 *and* ξ_i and maximized w.r.t. the *dual variables* α_i *and* μ_i.

Dual QP Task

The dual QP task is obtained when we take the Lagrangian

$$L(\boldsymbol{w}, w_0, \xi_i, \alpha_i, \mu_i) = \frac{1}{2}|\boldsymbol{w}|^2 + C\sum_{i=1}^{|T|} \xi_i - \sum_{i=1}^{|T|} \alpha_i \{y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 + \xi_i\} - \sum_{i=1}^{|T|} \mu_i \xi_i$$

and we substitute for the primary variables $\boldsymbol{w}$, w_0 and ξ_i.

For a stationary point:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{|T|} \alpha_i y^{(i)} \boldsymbol{x}^{(i)} = 0 \quad \Longrightarrow \quad \boldsymbol{w} = \sum_{i=1}^{|T|} \alpha_i y^{(i)} \boldsymbol{x}^{(i)}$$

$$\frac{\partial L}{\partial w_0} = -\sum_{i=1}^{|T|} \alpha_i y^{(i)} = 0 \quad \Longrightarrow \quad \sum_{i=1}^{|T|} \alpha_i y^{(i)} = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \Longrightarrow \quad C = \alpha_i + \mu_i$$

After substituting back to L and simplification we get the criterion of the dual task:

$$L_D = \frac{1}{2}\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} \alpha_i\alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \boldsymbol{x}^{(j)T} + \sum_{i=1}^{|T|} \alpha_i \xi_i + \sum_{i=1}^{|T|} \mu_i \xi_i - \sum_{i=1}^{N}\sum_{j=1}^{|T|} \alpha_i\alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \boldsymbol{x}^{(j)T}$$
$$- \sum_{i=1}^{|T|} \alpha_i y^{(i)} w_0 + \sum_{i=1}^{|T|} \alpha_i - \sum_{i=1}^{|T|} \alpha_i \xi_i - \sum_{i=1}^{|T|} \mu_i \xi_i = \sum_{i=1}^{|T|} \alpha_i - \frac{1}{2}\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} \alpha_i\alpha_j y^{(i)} y^{(j)} \boldsymbol{x}^{(i)} \boldsymbol{x}^{(j)T}$$

Relations of the variables in Lagrangian

Lagrangian

$$L(\boldsymbol{w}, w_0, \xi_i, \alpha_i, \mu_i) = \frac{1}{2}|\boldsymbol{w}|^2 + C\sum_{i=1}^{|T|}\xi_i - \sum_{i=1}^{|T|}\alpha_i\{y^{(i)}(\boldsymbol{x}^{(i)}w^T + w_0) - 1 + \xi_i\} - \sum_{i=1}^{|T|}\mu_i\xi_i$$

shall be minimized w.r.t. the *primary variables $\boldsymbol{w}$, w_0 and ξ_i* and maximized w.r.t. the *dual variables α_i and μ_i*.

1. If a point $\boldsymbol{x}^{(i)}$ lies on an incorrect side of plus- or minus-plane:
   - $y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 < 0$, then $\xi_i > 0$ so that $y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 + \xi_i = 0$
   - $\xi_i > 0$ and L must be maximized w.r.t. μ_i, so that μ_i must be as small as possible, i.e. $\mu_i = 0$
   - $C = \alpha_i + \mu_i$ a $\mu_i = 0$, so that $\alpha_i = C$
2. If a point $\boldsymbol{x}^{(i)}$ lies on a correct side of plus- or minus-plane:
   - $y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 > 0$, so that $\xi_i = 0$
   - $y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 + \xi_i > 0$ and L must be maximized w.r.t. α_i, so that α_i must be as small as possible, i.e. $\alpha_i = 0$
   - $C = \alpha_i + \mu_i$ and $\alpha_i = 0$, so that $\mu_i = C$
3. If a point $\boldsymbol{x}^{(i)}$ lies directly on plus- or minus-plane:
   - $y^{(i)}(\boldsymbol{x}^{(i)}\boldsymbol{w}^T + w_0) - 1 = 0$, so that $\xi_i = 0$
   - $0 < \mu_i < C$
   - $0 < \alpha_i < C$

Optimal separating hyperplane: remarks

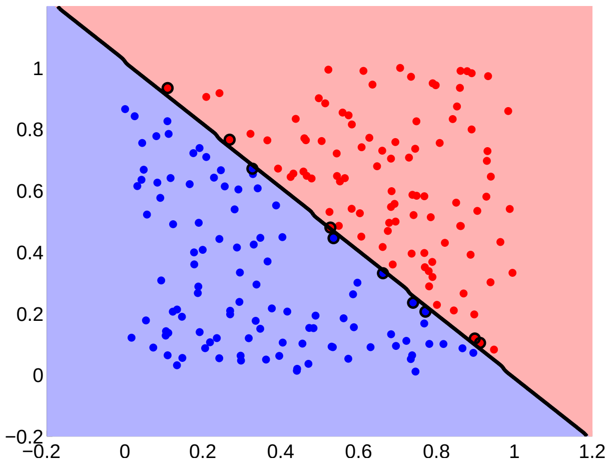
The importance of dual formulation:

- The QP task in dual formulation is easier to solve for QP solvers than the primal formulation.
- New, unseen examples can be classified using function

  $$f(\boldsymbol{x}, \boldsymbol{w}, w_0) = \text{sign}(\boldsymbol{x}\boldsymbol{w}^T + w_0) = \text{sign}\left(\sum_{i=1}^{|T|}\alpha_i y^{(i)}\boldsymbol{x}^{(i)}\boldsymbol{x}^T + w_0\right),$$

  i.e. the discrimination function contains the examples $\boldsymbol{x}$ only in the form of dot products (which will be useful later).
- The examples with $\alpha_i > 0$ are *support vectors*, thus the sums may be carried out only over the support vectors.
- The dual formulation contains the data only in the form of dot products which allows for other tricks you will learn later.
- The primal task with soft margin has double the number of constraints, the task is more complex, but
- the results for the QP task with soft margin are of the same type as in the separable case.

Optimal separating hyperplane: demo



Summary

Competencies

After this lecture, a student shall be able to ...

- define and recognize linear regression model (with scalar parameters, in scalar product form, in matrix form, non-homogenous and homogenous coordinates);
- define the loss function suitable for fitting a regression model;
- explain the least squares metod, draw an illustration;
- compute coefficients of simple (1D) linear regression by hand, write a computer program computing coefficients for multiple regression;
- explain the concept of discrimination function for binary and multinomial classification;
- define a loss function suitable for fitting a classification model;
- describe a perceptron algorithm, perform a few iterations by hand;
- explain the characteristics of perceptron algorithm;
- describe logistic regression, the interpretation of its outputs, and why we classify it as a linear model;
- define loss functions suitable for fitting logistic regression;
- define optimal separating hyperplane, explain in what sense it is optimal;
- define what a margin is, what support vectors are, and explain their relation;
- compute the margin given the parameters of separating hyperplane for which $\min_{i:y^{(i)}=+1}(x^{(i)}w^T + w_0) = 1$ and $\max_{i:y^{(i)}=-1}(x^{(i)}w^T + w_0) = -1$;
- formulate the primary quadratic programming task which results in the optimal separating hyperplane (including the soft-margin version);
- compute the parameters of optimal hyperplane given the set of support vectors and their weights.