



Linear Methods for Regression and Classification

Petr Pošík

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



Linear regression



Linear regression

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

- a training (multi)set $T = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(|T|)}, y^{(|T|)})\}$ 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) $\mathbf{x} = (x_1, \dots, x_D)$ and the dependent variable (output) y .

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- LSM
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Linear regression

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- LSM
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary

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

- a training (multi)set $T = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(|T|)}, y^{(|T|)})\}$ 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) $\mathbf{x} = (x_1, \dots, x_D)$ 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:

$$\hat{y} = h(\mathbf{x}) = w_0 + w_1 x_1 + \dots + w_D x_D = w_0 + \langle \mathbf{w}, \mathbf{x} \rangle = w_0 + \mathbf{x} \mathbf{w}^T,$$

where

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



Notation remarks

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

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

Linear regression

- Regression
- **Notation remarks**
- Train, apply
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Notation remarks

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

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

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

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

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

$$\hat{\mathbf{y}} = \mathbf{X} \mathbf{w}^T.$$

Linear regression

- Regression
- **Notation remarks**
- Train, apply
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

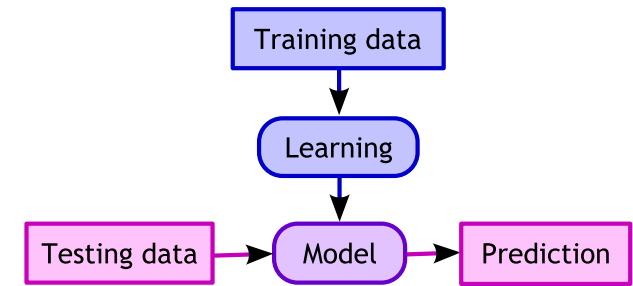
Summary



Two operation modes

Any ML model has 2 operation modes:

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



Linear regression

- Regression
- Notation remarks
- **Train, apply**
- 1D regression
- LSM
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

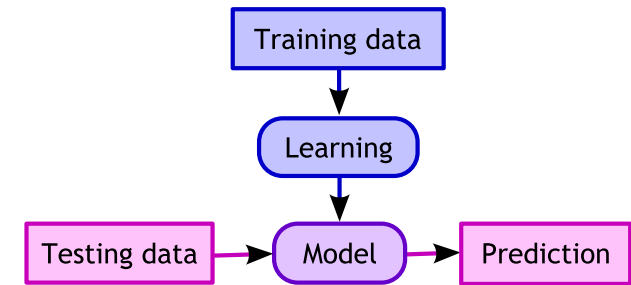
Summary



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: $h(\mathbf{x}, \mathbf{w})$.

Linear regression

- Regression
- Notation remarks
- **Train, apply**
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

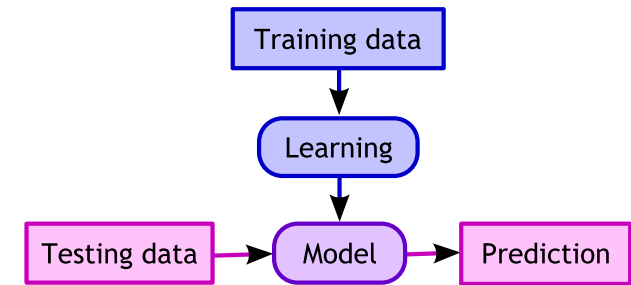
Summary



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: $h(\mathbf{x}, \mathbf{w})$.

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

$$\hat{y} = h(\mathbf{x}, \mathbf{w}) = h_{\mathbf{w}}(\mathbf{x}).$$

Linear regression

- Regression
- Notation remarks
- **Train, apply**
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

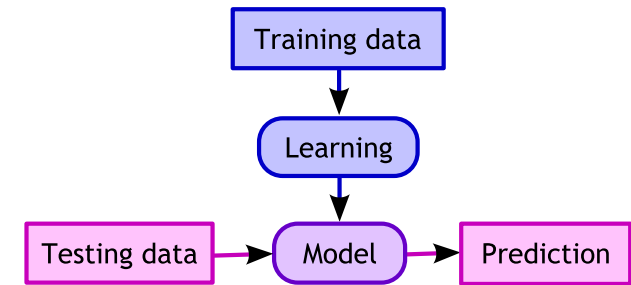
Summary



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: $h(\mathbf{x}, \mathbf{w})$.

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

$$\hat{y} = h(\mathbf{x}, \mathbf{w}) = h_{\mathbf{w}}(\mathbf{x}).$$

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

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

Linear regression

- Regression
- Notation remarks
- **Train, apply**
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

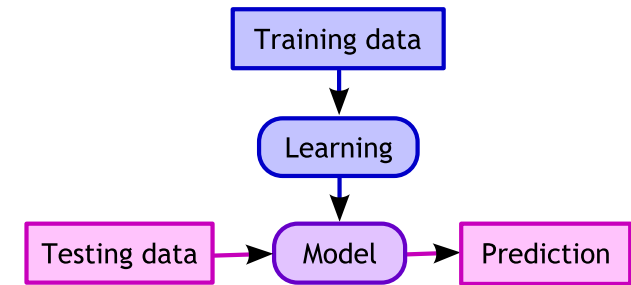
Summary



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: $h(\mathbf{x}, \mathbf{w})$.

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

$$\hat{y} = h(\mathbf{x}, \mathbf{w}) = h_{\mathbf{w}}(\mathbf{x}).$$

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

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

How to train the model?

Linear regression

- Regression
- Notation remarks
- **Train, apply**
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Simple (univariate) linear regression

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- **1D regression**
- LSM
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Simple (univariate) linear regression

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- **1D regression**
- LSM
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary

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 $T = \{(x^{(i)}, y^{(i)})\}_{i=1}^{|T|}$.



Simple (univariate) linear regression

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- **1D regression**
- LSM
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary

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

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



The least squares method

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- **LSM**
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

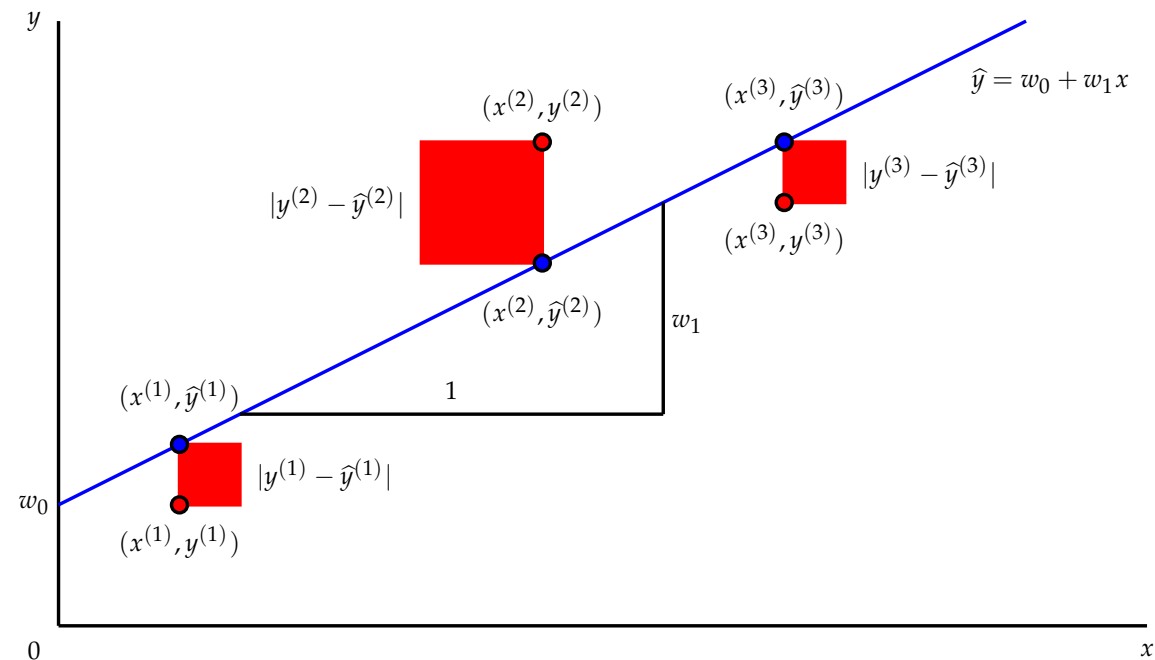
Perceptron

Logistic regression

Optimal separating hyperplane

Summary

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





The least squares method

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- **LSM**
- Minimizing $J(w, T)$
- Gradient descent
- Multivariate linear regression

Linear classification

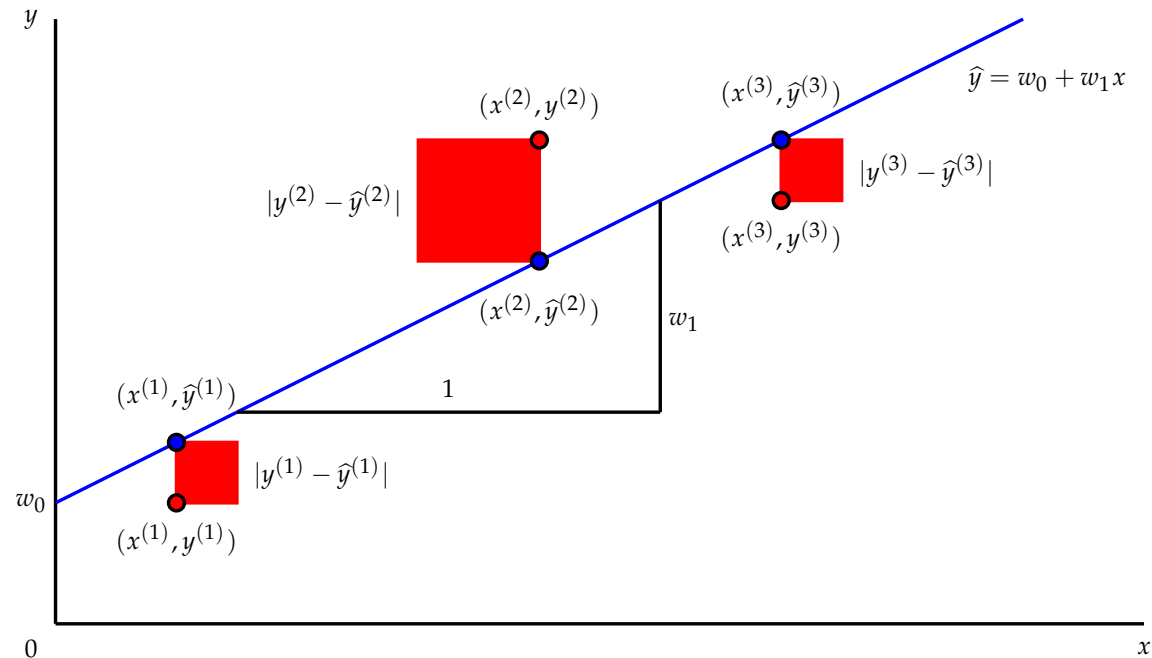
Perceptron

Logistic regression

Optimal separating hyperplane

Summary

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

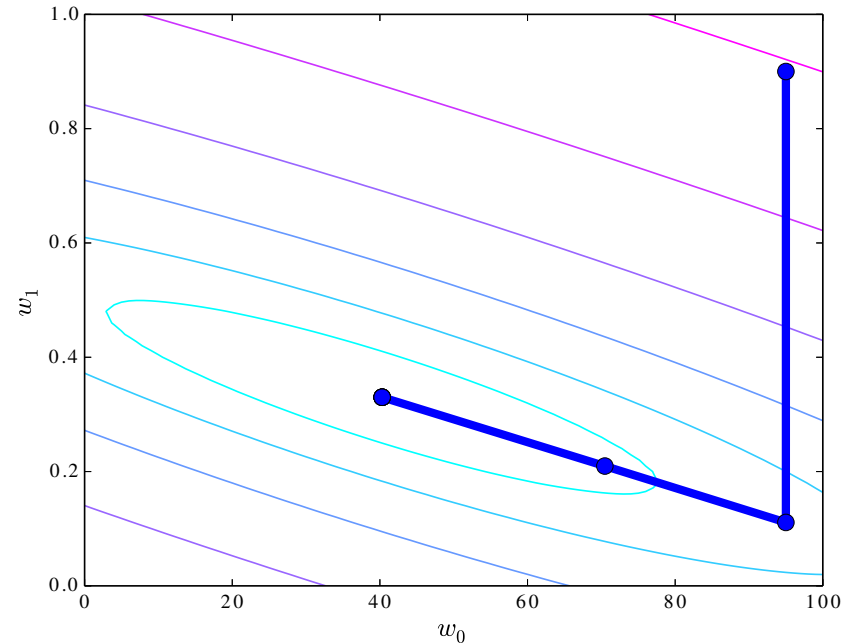
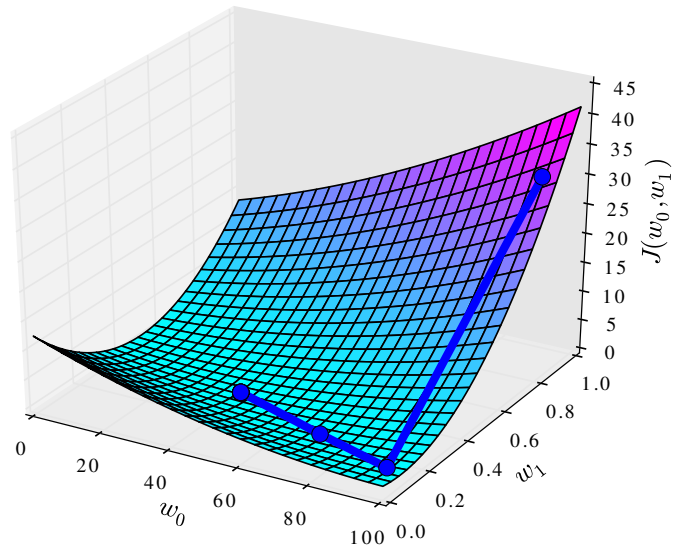


Explicit solution:

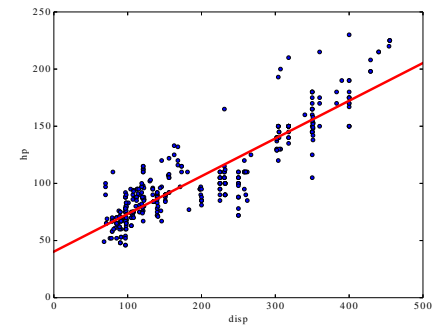
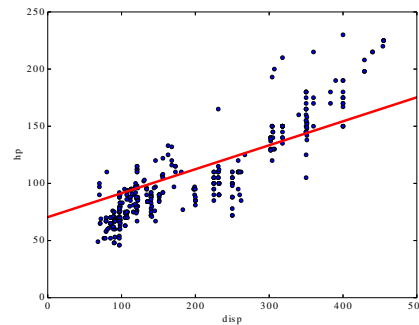
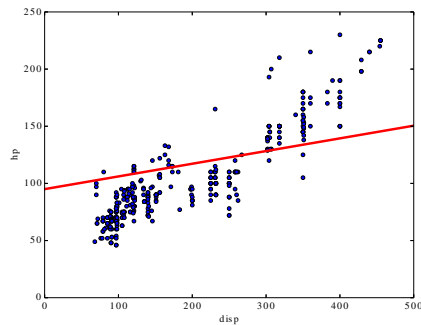
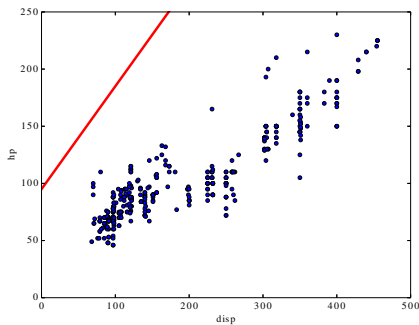
$$w_1 = \frac{\sum_{i=1}^{|T|} (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^{|T|} (x^{(i)} - \bar{x})^2} = \frac{s_{xy}}{s_x^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- LSM
- Minimizing $J(w, T)$
- **Gradient descent**
- Multivariate linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary

- 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:

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

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

where all w_i s are updated simultaneously and α is a **learning rate** (step size).

- For the cost function

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

the gradient can be computed as

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

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



Multivariate linear regression

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- LSM
- Minimizing $J(w, T)$
- Gradient descent
- **Multivariate linear regression**

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Multivariate linear regression

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

Model fitting:

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- **Multivariate linear regression**

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Multivariate linear regression

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

Linear regression

- Regression
- Notation remarks
- Train, apply
- 1D regression
- LSM
- Minimizing $J(\mathbf{w}, T)$
- Gradient descent
- **Multivariate linear regression**

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

Summary

Model fitting:

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

Fitting methods:

1. Numeric optimization of $J(\mathbf{w}, T)$:

- 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:

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

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



Linear classification



Binary classification task (dichotomy)

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

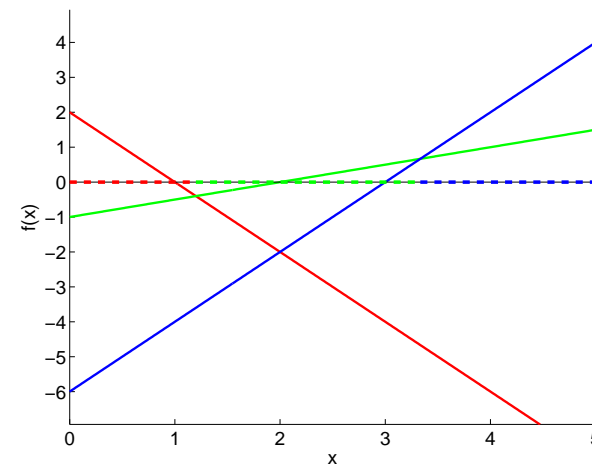
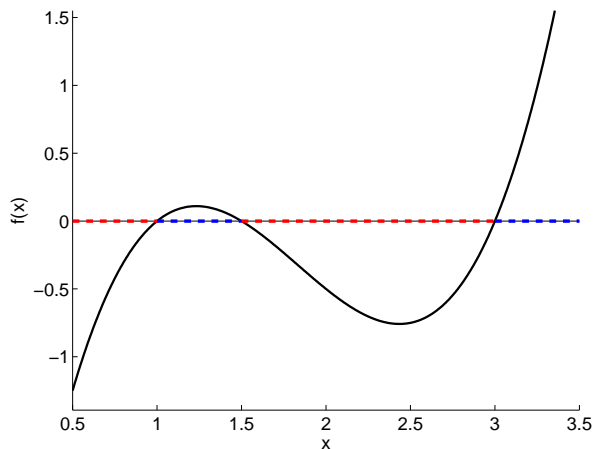
- each example is described by a vector of features $\mathbf{x} = (x_1, \dots, x_D)$,
- each example is labeled with the correct class $y \in \{+1, -1\}$.

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

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

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

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



Linear regression

Linear classification

- Binary class.
- Naive approach

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Naive approach

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

Linear regression

Linear classification

- Binary class.
- Naive approach

Perceptron

Logistic regression

Optimal separating
hyperplane

Summary



Naive approach

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

Naive solution: fit linear regression model to the data!

- Use cost function

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

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

Linear regression

Linear classification

- Binary class.
- Naive approach

Perceptron

Logistic regression

Optimal separating
hyperplane

Summary



Naive approach

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

Naive solution: fit linear regression model to the data!

- Use cost function

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

- minimize it with respect to \mathbf{w} ,
- and use $\hat{y} = \text{sign}(f(\mathbf{x}))$.
- 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:

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

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.

Linear regression

Linear classification

- Binary class.
- Naive approach

Perceptron

Logistic regression

Optimal separating hyperplane

Summary



Perceptron



Perceptron algorithm

Perceptron [Ros62]:

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

Linear regression

Linear classification

Perceptron

- Algorithm
- Demo
- Features
- Result

Logistic regression

Optimal separating hyperplane

Summary

Algorithm 1: Perceptron algorithm

Input: Linearly separable training dataset: $\{\mathbf{x}^{(i)}, y^{(i)}\}, \mathbf{x}^{(i)} \in \mathcal{R}^{D+1}$ (homogeneous coordinates), $y^{(i)} \in \{+1, -1\}$

Output: Weight vector w such that $\mathbf{x}^{(i)} w^T > 0$ iff $y^{(i)} = +1$ and $\mathbf{x}^{(i)} w^T < 0$ iff $y^{(i)} = -1$

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

Linear regression

Linear classification

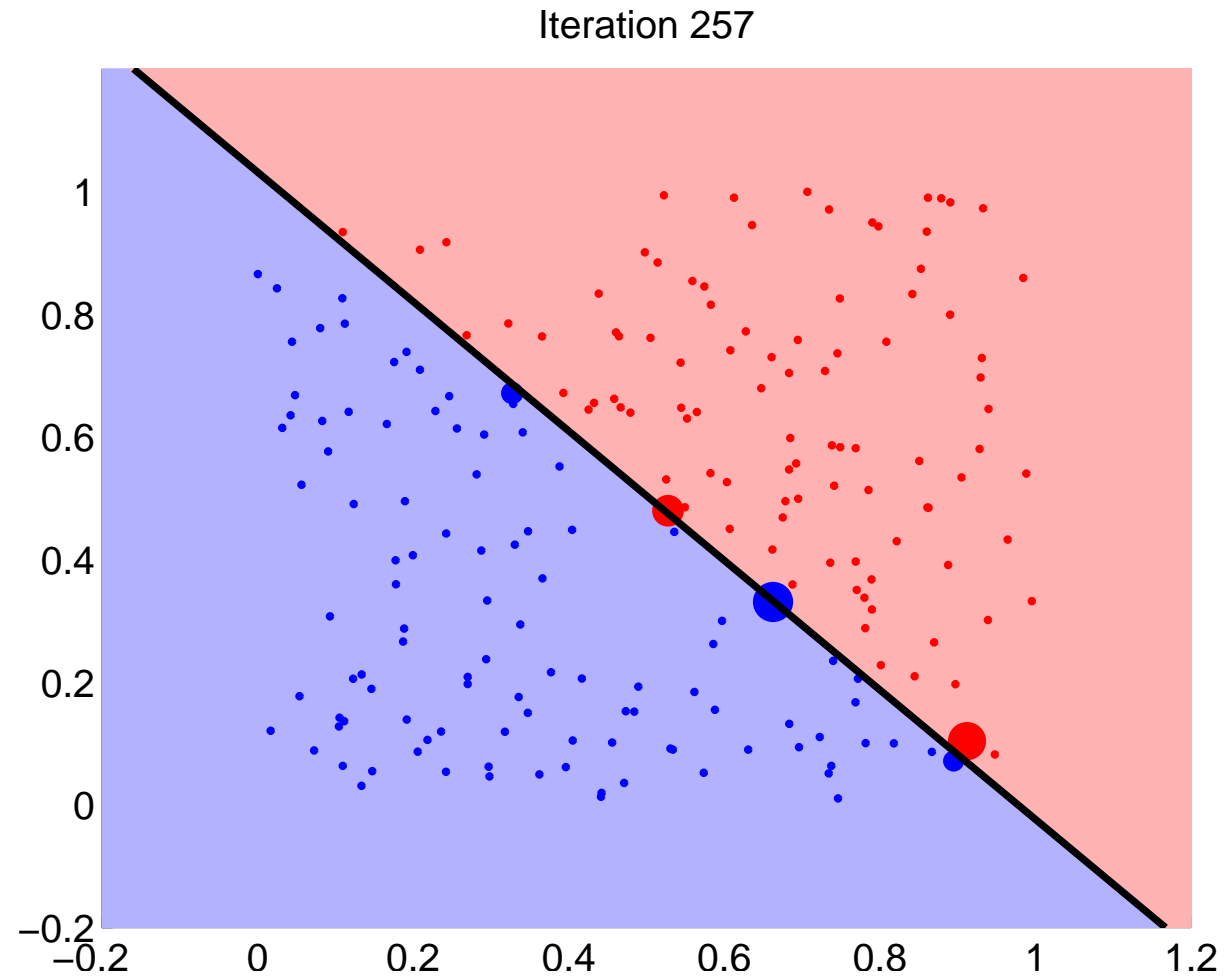
Perceptron

- Algorithm
- Demo
- Features
- Result

Logistic regression

Optimal separating hyperplane

Summary





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.

Linear regression

Linear classification

Perceptron

- Algorithm
- Demo
- **Features**
- Result

Logistic regression

Optimal separating hyperplane

Summary

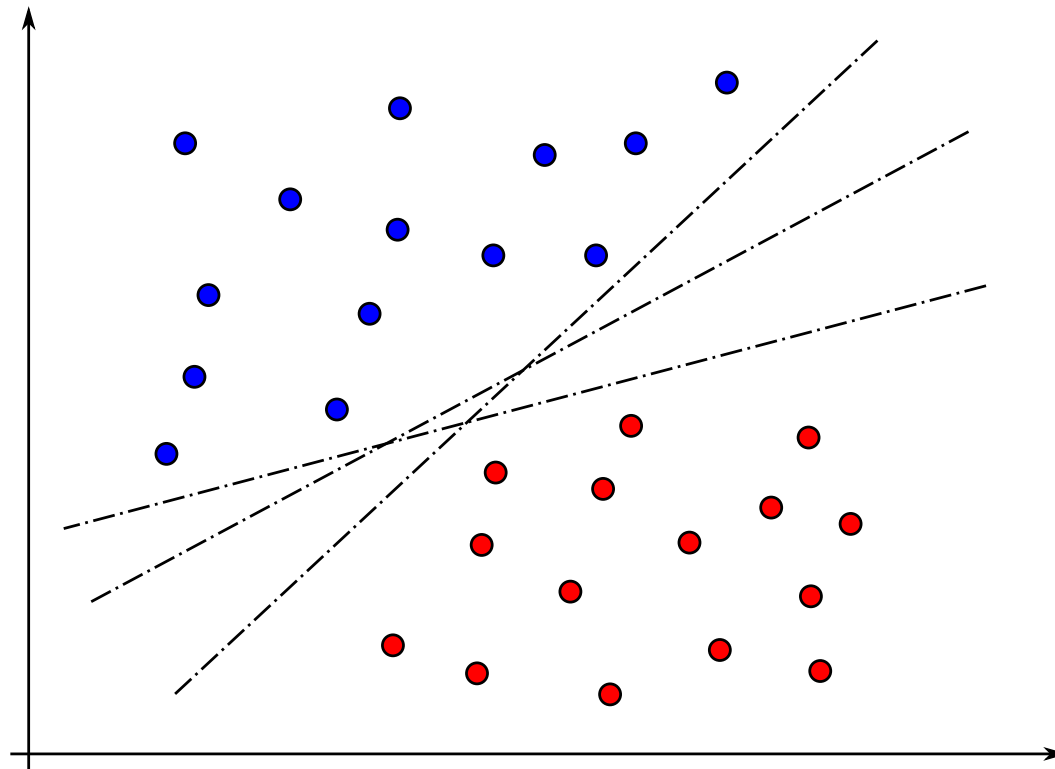
[Nov62] Albert B. J. Novikoff. On convergence proofs for perceptrons. In *Proceedings of the Symposium on Mathematical Theory of Automata*, volume 12, Brooklyn, New York, 1962.



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?

Linear regression

Linear classification

Perceptron

- Algorithm
- Demo
- Features
- Result

Logistic regression

Optimal separating hyperplane

Summary



Logistic regression



Logistic regression model

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

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

Linear regression

Linear classification

Perceptron

Logistic regression

- Model
- Cost function

Optimal separating hyperplane

Summary



Logistic regression model

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

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

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

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

Linear regression

Linear classification

Perceptron

Logistic regression

- Model
- Cost function

Optimal separating hyperplane

Summary



Logistic regression model

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

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

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

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

Interpretation of the model:

- $f_w(\mathbf{x})$ is interpreted as an estimate of the probability that \mathbf{x} belongs to class 1.
- The **decision boundary** is defined using a different level-set: $\{\mathbf{x} : f_w(\mathbf{x}) = 0.5\}$.
- Logistic *regression* is a *classification model*!
- The discrimination function $f_w(\mathbf{x})$ 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!

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

Linear regression

Linear classification

Perceptron

Logistic regression

- Model
- Cost function

Optimal separating hyperplane

Summary



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.

Linear regression

Linear classification

Perceptron

Logistic regression

- Model
- **Cost function**

Optimal separating
hyperplane

Summary



Cost function

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

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

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*):

$$J(\mathbf{w}, T) = \frac{1}{|T|} \sum_{i=1}^{|T|} \text{cost}(y^{(i)}, f_{\mathbf{w}}(\mathbf{x}^{(i)})), \text{ where}$$
$$\text{cost}(y, \hat{y}) = \begin{cases} -\log(\hat{y}) & \text{if } y = 1 \\ -\log(1 - \hat{y}) & \text{if } y = 0 \end{cases} ,$$

which can be rewritten in a single expression as

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

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

Linear regression

Linear classification

Perceptron

Logistic regression

- Model
- Cost function

Optimal separating
hyperplane

Summary



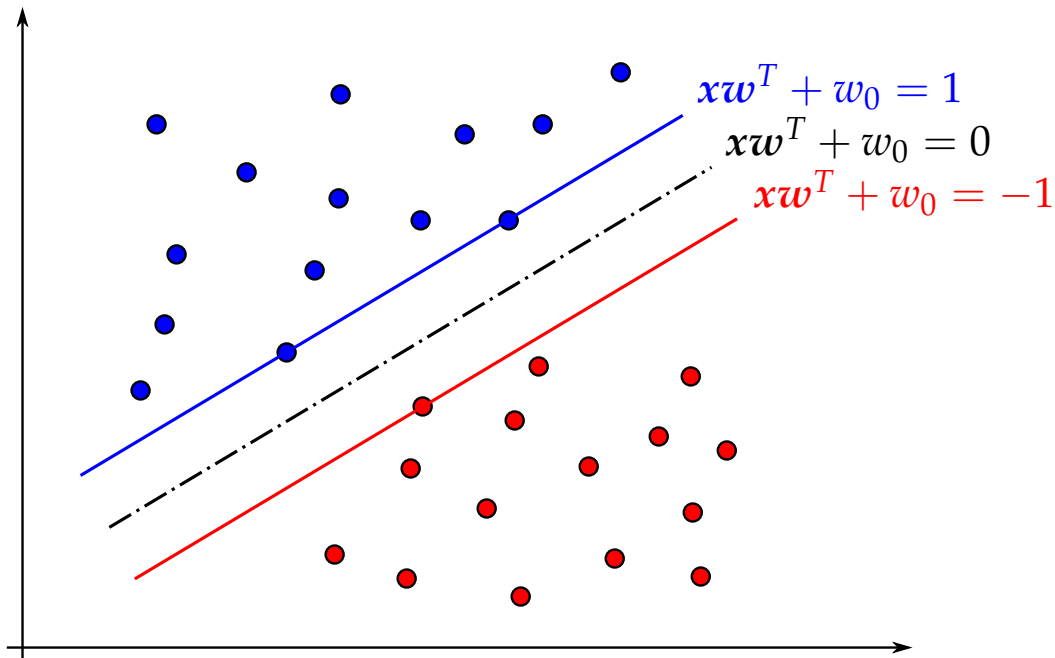
Optimal separating hyperplane

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: $\{x : xw^T + w_0 = 1\}$

Minus 1 level: $\{x : xw^T + w_0 = -1\}$

Decision boundary

(separating hyperplane): $\{x : xw^T + w_0 = 0\}$

Support vectors:

- Data points 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?

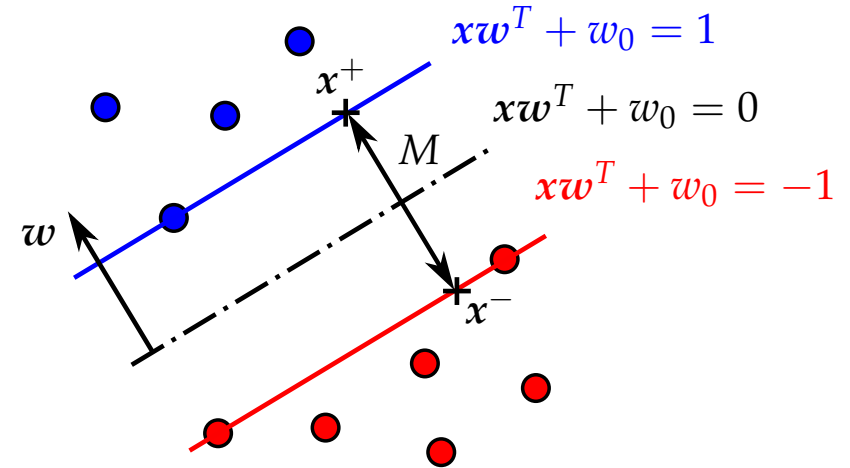
- Intuitively, it is safe.
- If we make a small error in estimating the boundary, the classification will likely stay correct.
- The model is invariant with respect to the training set changes, except the changes of support vectors.
- There are sound theoretical results that having a maximum margin classifier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



Margin size

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

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



We know that:

$$x^+ w^T + w_0 = 1$$

$$x^- w^T + w_0 = -1$$

$$x^- + \lambda w = x^+$$

And we can derive:

$$(x^+ - x^-) w^T = 2$$

$$(x^- + \lambda w - x^-) w^T = 2$$

$$\lambda w w^T = 2$$

$$\lambda = \frac{2}{w w^T} = \frac{2}{\|w\|^2}$$

Thus the margin size is

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

Linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

- Optimal SH
- **Margin size**
- OSH learning
- Non-separable case
- OSH learning (2)
- OSH: remarks
- Demo

Summary



Optimal separating hyperplane learning

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

Linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating
hyperplane

- Optimal SH
- Margin size
- **OSH learning**
- Non-separable case
- OSH learning (2)
- OSH: remarks
- Demo

Summary



Optimal separating hyperplane learning

We want to maximize margin $M = \frac{2}{\|w\|}$ 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:

minimize ww^T with respect to w_0, \dots, w_D

subject to $y^{(i)}(x^{(i)}w^T + w_0) \geq 1 \quad \forall i \in 1, \dots, |T|$.

Linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

- Optimal SH
- Margin size
- **OSH learning**
- Non-separable case
- OSH learning (2)
- OSH: remarks
- Demo

Summary



Optimal separating hyperplane learning

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

Linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

- Optimal SH
- Margin size
- **OSH learning**
- Non-separable case
- OSH learning (2)
- OSH: remarks
- Demo

Summary

■ Primary QP task:

minimize $w w^T$ with respect to w_0, \dots, w_D

subject to $y^{(i)} (x^{(i)} w^T + w_0) \geq 1 \quad \forall i \in 1, \dots, |T|$.

■ Dual QP task:

maximize $\sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} x^{(j)T}$ with respect to $\alpha_1, \dots, \alpha_{|T|}$

subject to $\alpha_i \geq 0$

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



Optimal separating hyperplane learning

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

Linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

- Optimal SH
- Margin size
- **OSH learning**
- Non-separable case
- OSH learning (2)
- OSH: remarks
- Demo

Summary

- Primary QP task:

minimize $w w^T$ with respect to w_0, \dots, w_D

subject to $y^{(i)} (\mathbf{x}^{(i)} w^T + w_0) \geq 1 \quad \forall i \in 1, \dots, |T|$.

- Dual QP task:

maximize $\sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \mathbf{x}^{(j)T}$ with respect to $\alpha_1, \dots, \alpha_{|T|}$

subject to $\alpha_i \geq 0$

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:

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

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



Non-separable case

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

Slack variables ζ_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”.

Linear regression

Linear classification

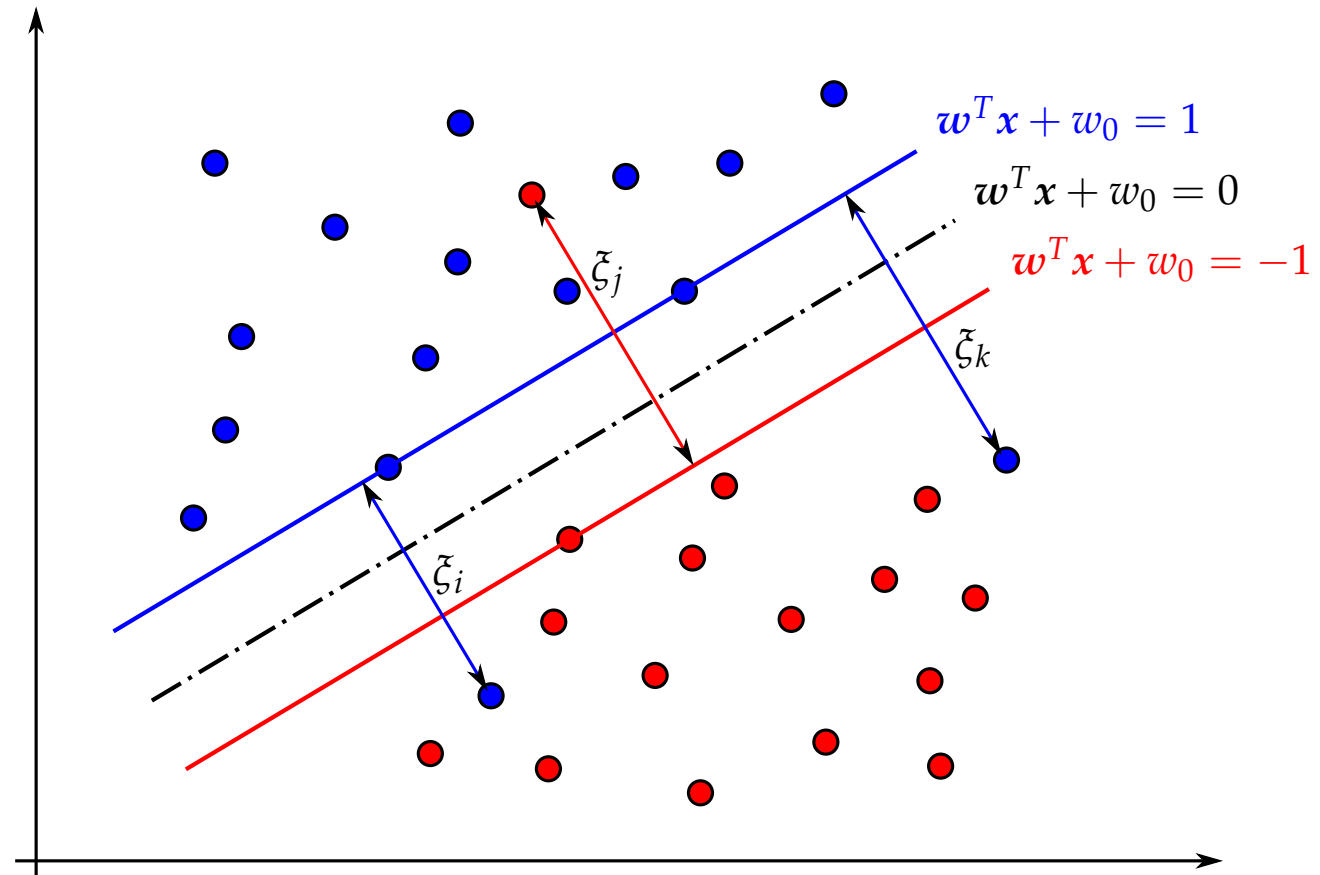
Perceptron

Logistic regression

Optimal separating hyperplane

- Optimal SH
- Margin size
- OSH learning
- **Non-separable case**
- OSH learning (2)
- OSH: remarks
- Demo

Summary



Optimal separating hyperplane learning for non-separable data

- Primary QP task with **slack variables**:

minimize $(\mathbf{w}\mathbf{w}^T + C \sum_{i=1}^{|T|} \tilde{\zeta}_i)$ with respect to $w_0, \dots, w_D, \tilde{\zeta}_1, \dots, \tilde{\zeta}_{|T|}$

subject to $y^{(i)}(\mathbf{x}^{(i)}\mathbf{w}^T + w_0) \geq 1 - \tilde{\zeta}_i \quad \forall i \in 1, \dots, |T|,$

and $\tilde{\zeta}_i \geq 0 \quad \forall i \in 1, \dots, |T|.$

Optimal separating hyperplane learning for non-separable data

- Primary QP task with **slack variables**:

$$\text{minimize } (\mathbf{w}\mathbf{w}^T + C \sum_{i=1}^{|T|} \zeta_i) \text{ with respect to } w_0, \dots, w_D, \zeta_1, \dots, \zeta_{|T|}$$

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

$$\text{and } \zeta_i \geq 0 \quad \forall i \in 1, \dots, |T|.$$

- Dual QP task:

$$\text{maximize } \sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j \mathbf{y}^{(i)} \mathbf{y}^{(j)} \mathbf{x}^{(i)} \mathbf{x}^{(j)T} \text{ with respect to } \alpha_1, \dots, \alpha_{|T|}, \mu_1, \dots, \mu_{|T|},$$

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

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

Optimal separating hyperplane learning for non-separable data

- Primary QP task with **slack variables**:

$$\text{minimize } (\mathbf{w}\mathbf{w}^T + C \sum_{i=1}^{|T|} \zeta_i) \text{ with respect to } w_0, \dots, w_D, \zeta_1, \dots, \zeta_{|T|}$$

$$\text{subject to } y^{(i)} (\mathbf{x}^{(i)} \mathbf{w}^T + w_0) \geq 1 - \zeta_i \quad \forall i \in 1, \dots, |T|,$$
$$\text{and } \zeta_i \geq 0 \quad \forall i \in 1, \dots, |T|.$$

- Dual QP task:

$$\text{maximize } \sum_{i=1}^{|T|} \alpha_i - \frac{1}{2} \sum_{i=1}^{|T|} \sum_{j=1}^{|T|} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \mathbf{x}^{(j)T} \text{ with respect to } \alpha_1, \dots, \alpha_{|T|}, \mu_1, \dots, \mu_{|T|},$$

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

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

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

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

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



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

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

i.e. the discrimination function contains the examples \mathbf{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.

Linear regression

Linear classification

Perceptron

Logistic regression

Optimal separating hyperplane

- Optimal SH
- Margin size
- OSH learning
- Non-separable case
- OSH learning (2)
- **OSH: remarks**
- Demo

Summary



Optimal separating hyperplane: demo

Linear regression

Linear classification

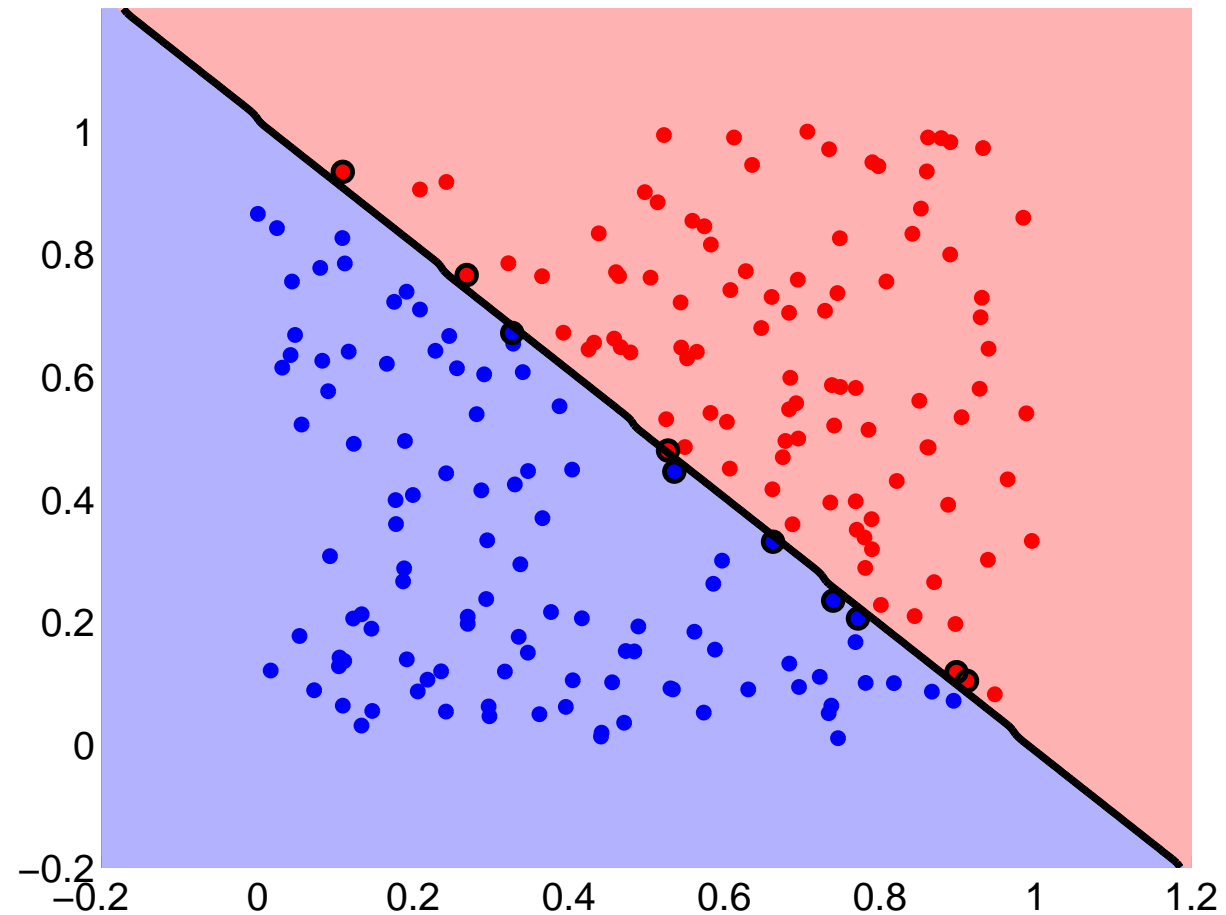
Perceptron

Logistic regression

Optimal separating hyperplane

- Optimal SH
- Margin size
- OSH learning
- Non-separable case
- OSH learning (2)
- OSH: remarks
- Demo

Summary





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 method, 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.