

Introduction to Artificial Intelligence: Methods, Models, Algorithms

Aleksandr I. Panov and Konstantin S. Yakovlev

National Research University Higher School of Economics

25 July 2017 – Summer University

apanov@hse.ru



Nearest-neighbor method

- It is a task of classification
- Data: $X = (x_i, y_i)_{i=1}^I$
- Learning process: we just remember the data

Nearest-neighbor method

- New object x

- Sorting objects of training data by distance from x :

$$\rho(x, x_{(1)}) \leq \dots \leq \rho(x, x_{(l)})$$

- Choose class that is most popular among k nearest neighbors:

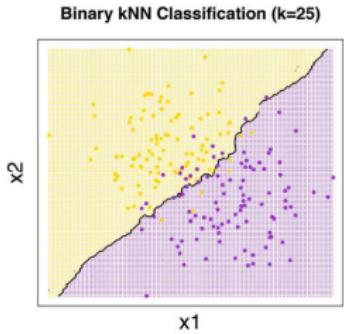
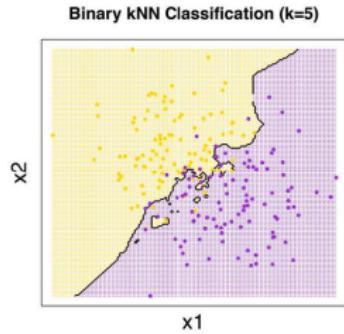
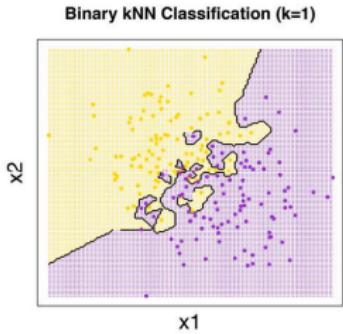
$$a(x) = \arg \min_{y \in \mathbb{Y}} \sum_{i=1}^k [y_{(i)} = y]$$

Nearest-neighbor method

$$a(x) = \arg \min_{y \in \mathbb{Y}} \sum_{i=1}^k [y_{(i)} = y]$$

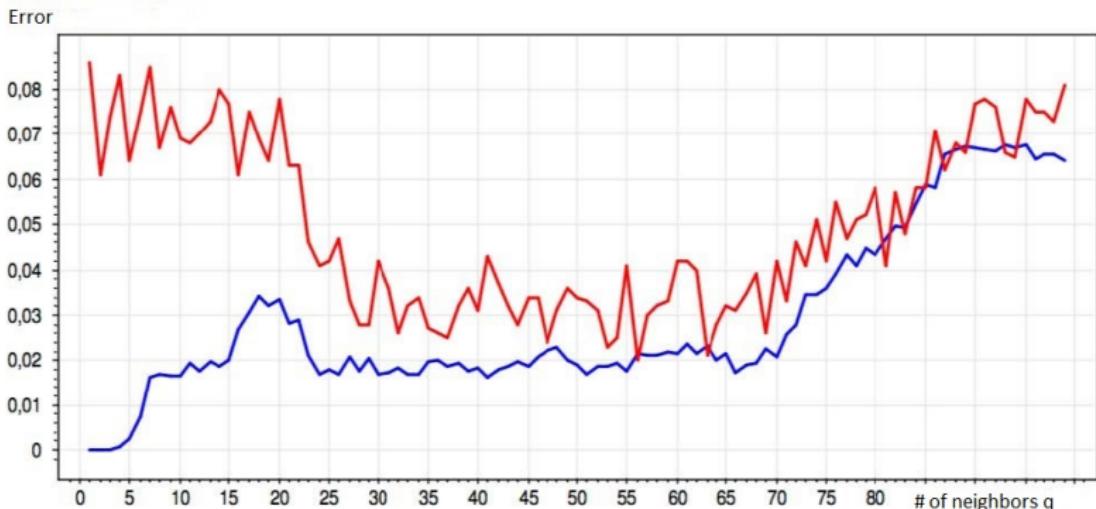
- k - hyperparameter of the algorithm
- we can select it via using of hold-out dataset and cross-validation
- The larger k , the easier the separating surface

How to choose number of neighbors?

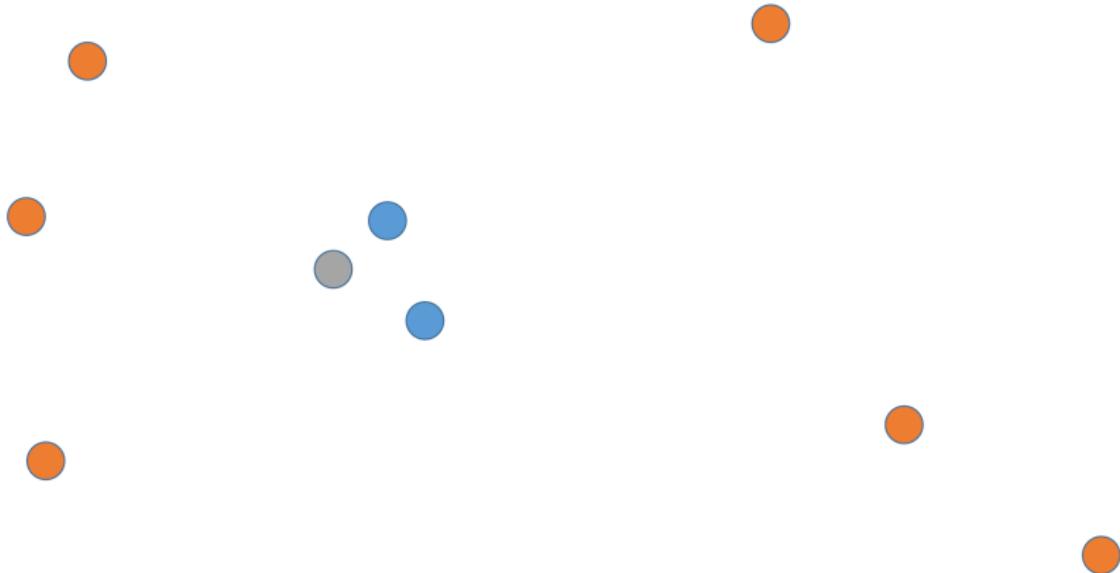


How to choose number of neighbors?

- Blue - learning error
- Red - cross-validation error



Problem of k-NN



Nearest-neighbor method

- Do not take into account the distances to k nearest neighbors
- Closer neighbors should be more important

k-NN with weights

$$a(x) = \arg \min_{y \in \mathbb{Y}} \sum_{i=1}^k w_i [y_{(i)} = y]$$

Options:

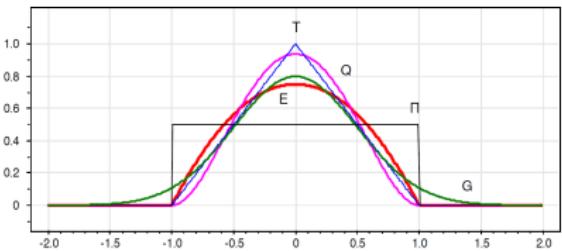
- $w_i = \frac{k+1-i}{k}$
- $w_i = q^i$
- Not a distance weight

k-NN with weights

$$a(x) = \arg \min_{y \in \mathbb{Y}} \sum_{i=1}^k w_i [y(i) = y]$$

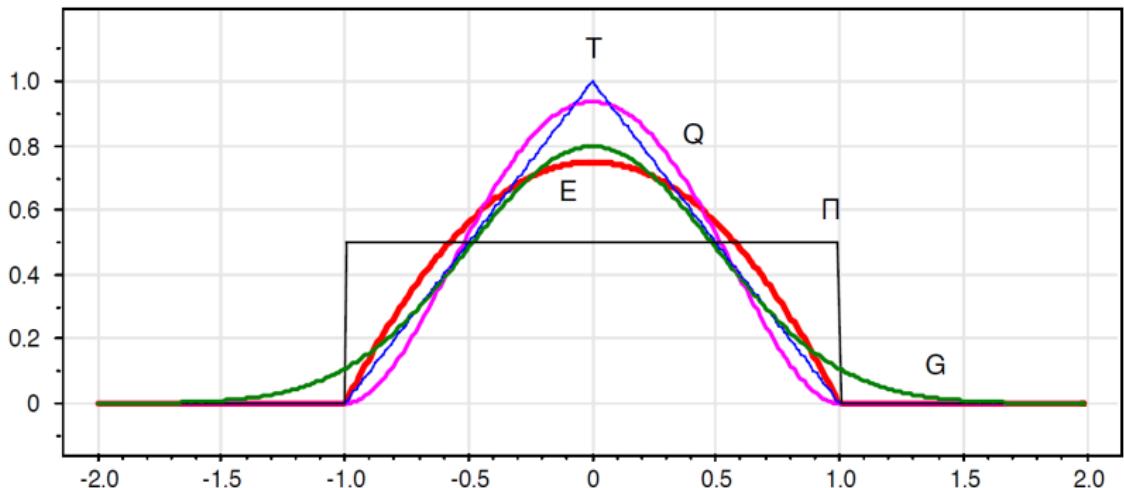
The Parzenov window:

- $w_i = K\left(\frac{\rho(x, x_{(i)})}{h}\right)$
- K - kernel
- h - width of the window

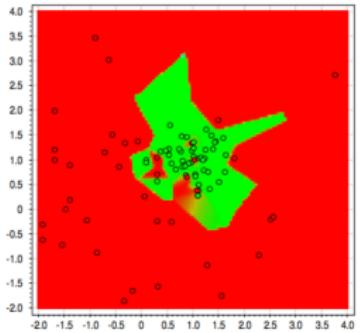


Nearest-neighbor method

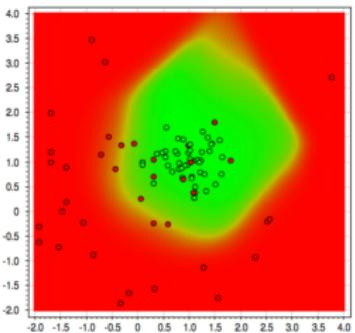
- Gaussian kernel: $K(z) = (2\pi)^{-0.5} \exp^{-\frac{1}{2}z^2}$
- And many others



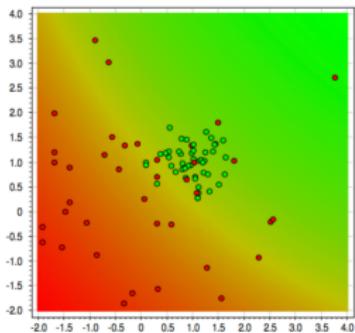
Kernels



$h = 0.05$



$h = 0.5$



$h = 5$

Properties of kNN

- Training is absent - it is only necessary to memorize the training sample
- To apply the model, you need to calculate the distances from the new object to all the training objects
- Application requires $l \times d$ operations
- There are special methods for finding nearest neighbors

kNN for regression

- Classification:

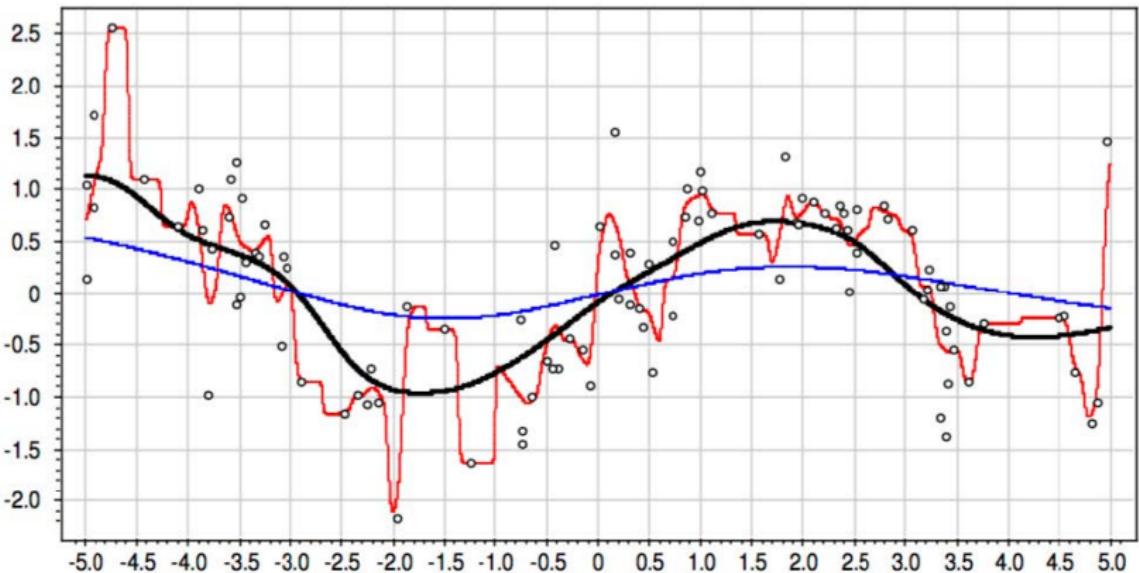
$$a(x) = \arg \min_{y \in \mathbb{Y}} \sum_{i=1}^k [y_{(i)} = y]$$

- Regression:

$$a(x) = \frac{\sum_i^k w_i y_{(i)}}{\sum_{i=1}^k w_i}$$

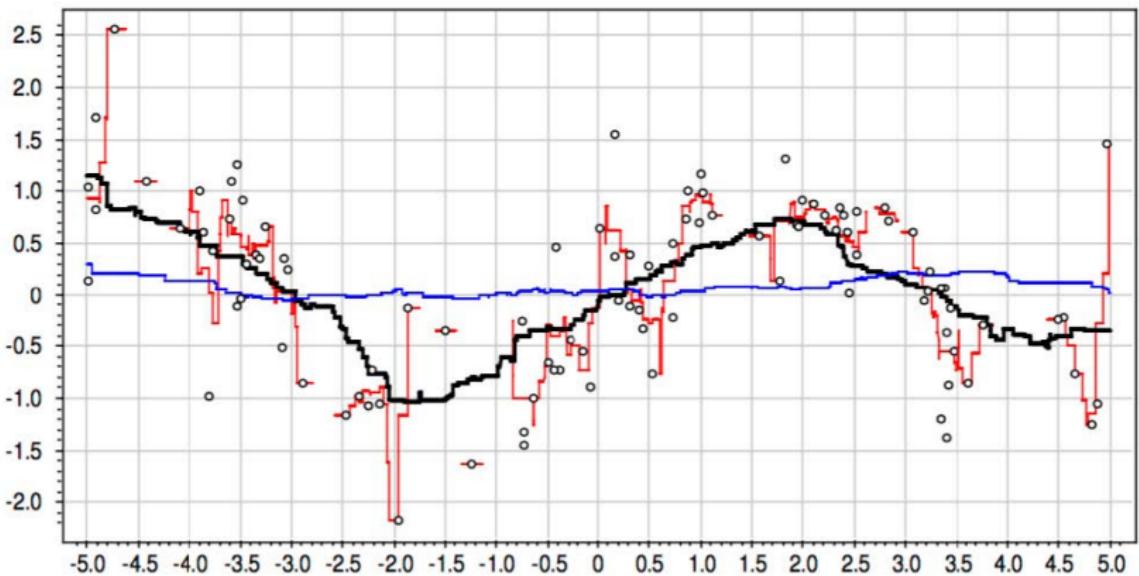
kNN for regression

- Gaussian kernel
- $h \in \{0.1, 1.0, 3.0\}$

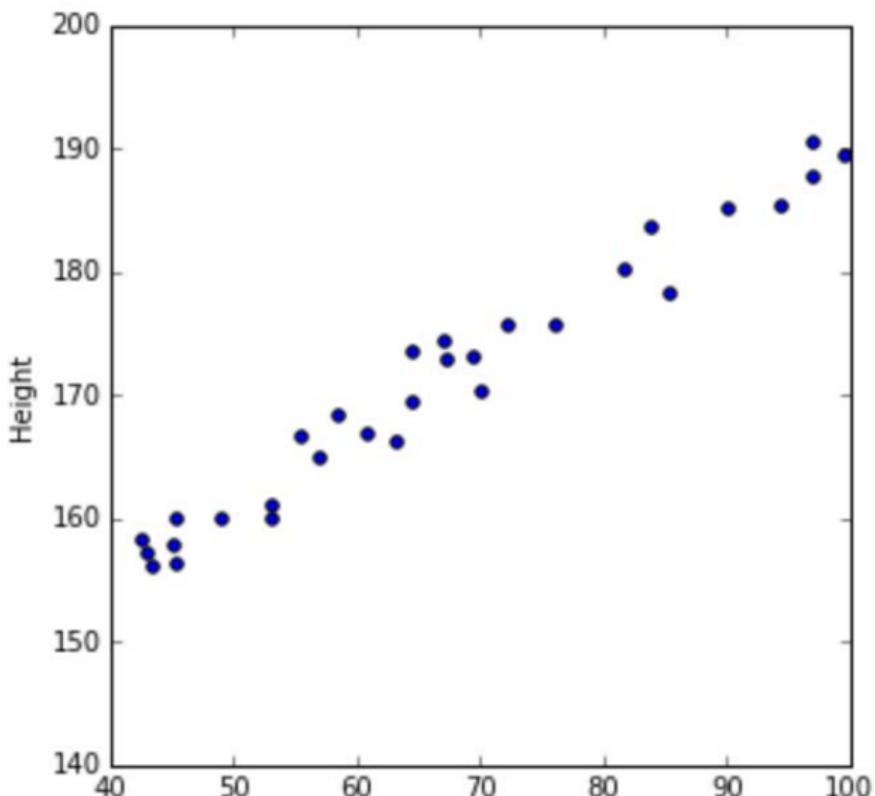


kNN for regression

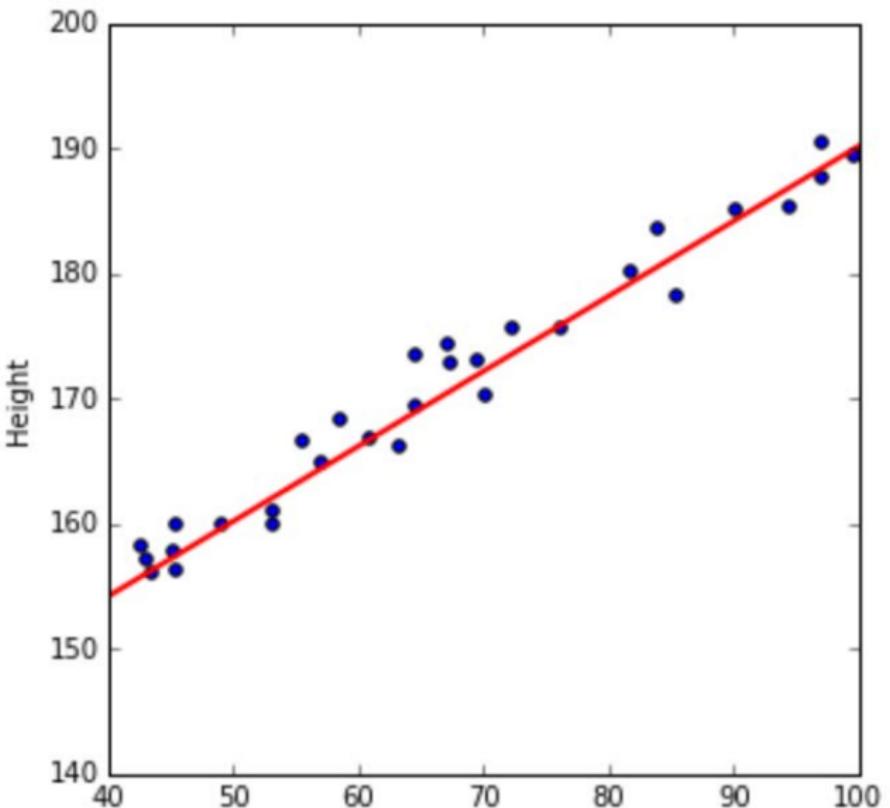
- Rectangle kernel $K(z) = [|z| \leq 1]$
- $h \in \{0.1, 1.0, 3.0\}$



One-dimension sample



One-dimension sample



Paired regression

- The simplest case: only one feature
- Model: $a(x) = w_0 + w_1x$
- Two parameters: w_0 and w_1
- The simplest model

Linear regression

- Weighted sum of features

$$a(x) = w_0 + w_1 x^1 + \cdots + w_d x^d$$

- x^1, x^2, \dots, x^d - values of features
- w_0, w_1, \dots, w_d - parameters
- w_0 - bias

Linear regression

$$a(x) = w_0 \times 1 + w_1 x^1 + \cdots + w_d x^d$$

- w_0 - as a coefficient with a single feature
- and if we add it:

$$\begin{pmatrix} x_{11} & \cdots & x_{1d} & 1 \\ \vdots & & \vdots & \vdots \\ x_{l1} & \cdots & x_{ld} & 1 \end{pmatrix}$$

Linear regression

- It gives as scalar product

$$a(x) = w_0 \times 1 + w_1 x^1 + \cdots + w_d x^d = \langle w, x \rangle$$

- Learning:

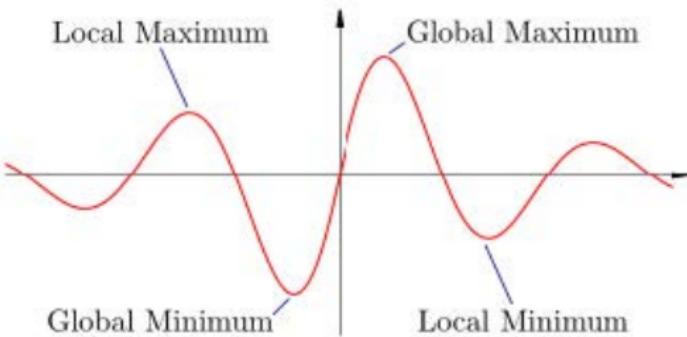
$$\sum_{i=1}^I (\langle w, x_i \rangle - y_i)^2 \rightarrow \min_w$$

- In vector form:

$$Q(w, X) = \frac{1}{q} \|Xw - y\|^2$$

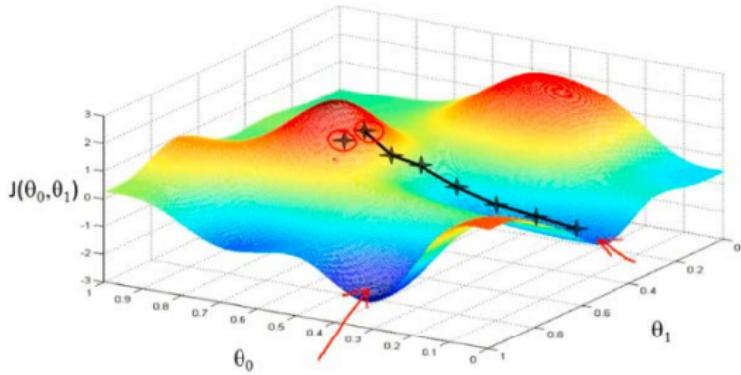
Extremes

- Extreme - minimum or maximum
- The local minimum is less than all values in some neighborhood
- Global minimum - the least of all values



Extremes

- Local minima are one of the main problems in machine learning



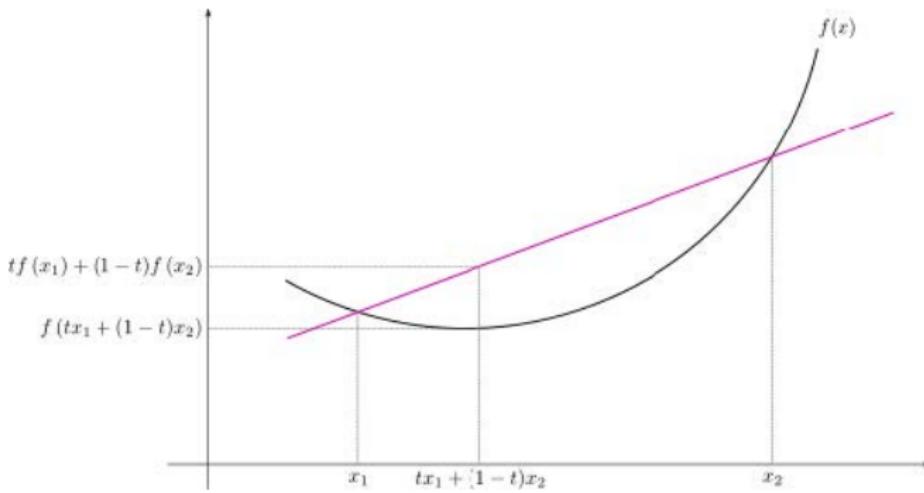
Andrew Ng

The optimality condition

- How do we know if the point x_0 is an extremum?
- Fermat's theorem: if the point x_0 is an extremum, and there is a derivative in it, then $f'(x_0) = 0$
- If the function has a derivative everywhere: we need to solve $f'(x) = 0$
- If there is a problem with the derivative: no luck
- Even if there is a derivative, then what to do with local extremes?

Convex functions

- A convex function if its graph lies below any segment joining two points



Convex functions

- The function is convex if, at all points $f''(x) > 0$
- An important property: any local extremum of a convex function is global
- Solving the equation $f'(x) = 0$, we obtain global extrema
- Conclusion: we will try to choose convex functionals!

An example

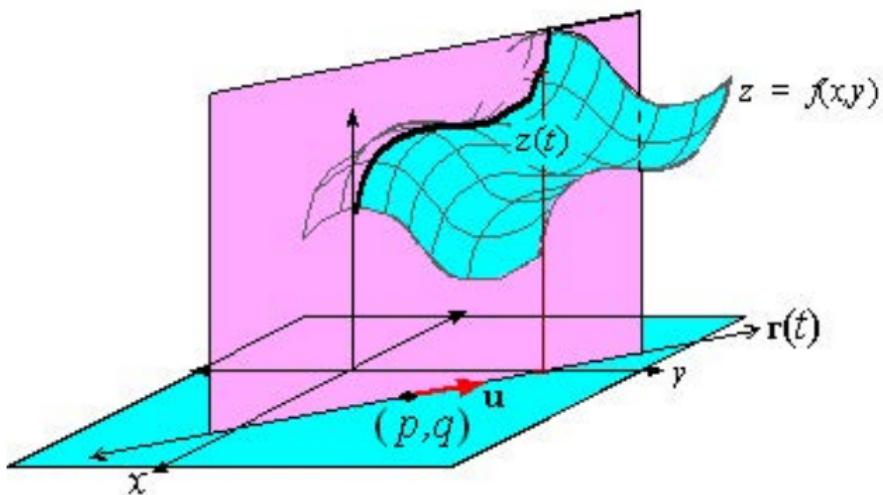
- Quality functional for linear regression:

$$Q(w_1, \dots, w_d) = \sum_{i=1}^I (w_1 x^1 + \dots + w_d x^d - y_i)^2$$

- How to search for its minimum?

Derivative in direction

- How fast is the function growing in a particular direction?



Derivative in direction

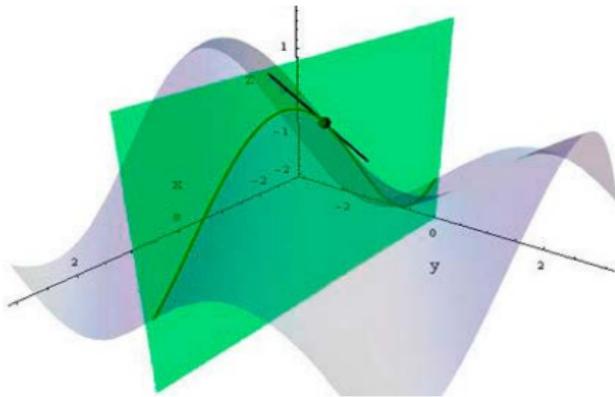
- Direction: v and $\|v\| = 1$
- Derivative:

$$f'_v(x_0) = \lim_{t \rightarrow 0} \frac{f(x_0 + tv) - f(x_0)}{t}$$

Partial derivative

- How fast is the function changing along the variable x_i ?
- The partial derivative with respect to x_i :

$$\frac{\partial f}{\partial x_i} = \lim_{t \rightarrow 0} \frac{f(x_1, \dots, x_i + t, \dots, x_d) - f(x_1, \dots, x_i, \dots, x_d)}{t}$$



Gradient

- Gradient is vector of partial derivatives:

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

- Gradient has a very important property!

Gradient

- Lets fix a point x_0
- In which direction is the function growing fastest?

$$f'_v(x_0) \rightarrow \max_v$$

- Relationship of directional derivative and gradient:

$$f'_v(x_0) = \langle \nabla f(x_0, v) \rangle = \|\nabla f(x_0)\| \times \|v\| \times \cos \phi$$

Gradient

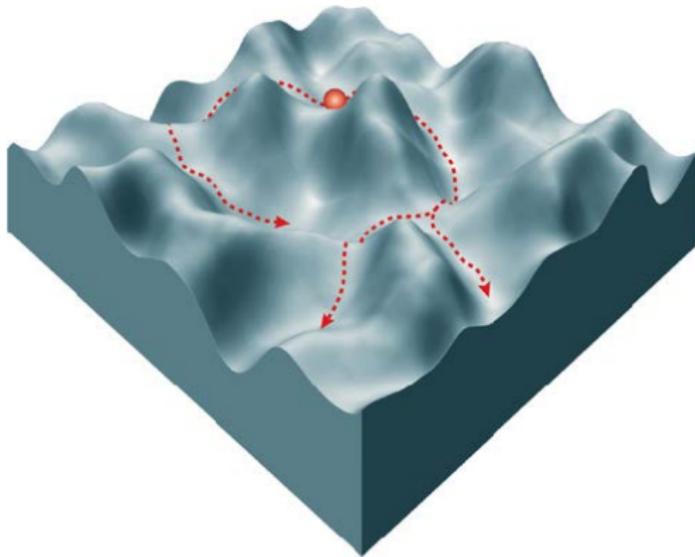
- An arbitrary direction is maximal if the direction coincides with the gradient!
- **Gradient - the direction of the steepest growth of the function**
- Antigradient - the direction of steepest descent

The optimality condition

- How do we know if the point x_0 is an extremum?
- Fermat's theorem: if the point x_0 is an extremum, and there is a gradient in it, then $\nabla f(x_0) = 0$
- If the function has a gradient everywhere: we need to solve $\nabla f(x) = 0$
- If there is a problem with the gradient: no luck

Extremes

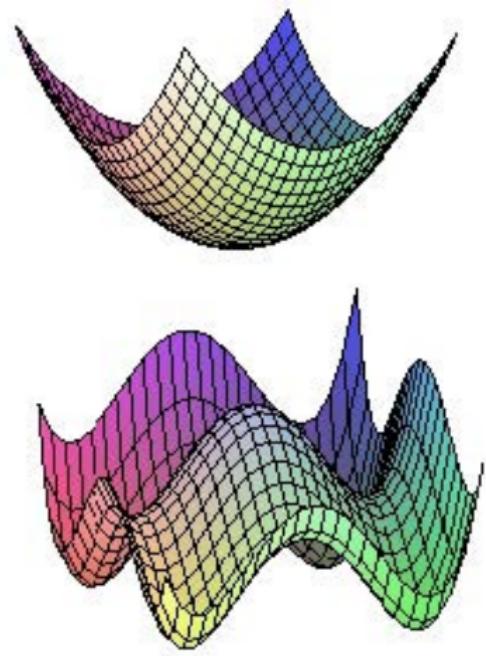
- The problem with local extremes is still relevant



Nature Reviews | Molecular Cell Biology

Convex functions

The function is convex if its graph lies below the segment joining any two points



Optimization problem

$$Q(w, X) = \frac{1}{l} \sum_{i=1}^l l(\langle w, x_i \rangle - y_i)^2 \rightarrow \min_w$$

- A gradient exists at any point
- Convex function
- The only minimum (not always)

Optimization problem

$$\nabla Q(w, X) = \left(\frac{\partial Q}{\partial w_1}, \dots, \frac{\partial Q}{\partial w_d} \right)$$

Derivatives:

$$\frac{\partial Q}{\partial w_j} = \frac{2}{l} \sum_{i=1}^l l x_i^j (\langle w, x_i \rangle - y_i)$$

Learning linear regressor

- In a vector form:

$$Q(w, X) = \frac{1}{l} \|Xw - y\|^2$$

- Condition of minimum:

$$\nabla Q(w, X) = 0$$

- What if we try to solve this system of equations?

Learning linear regressor

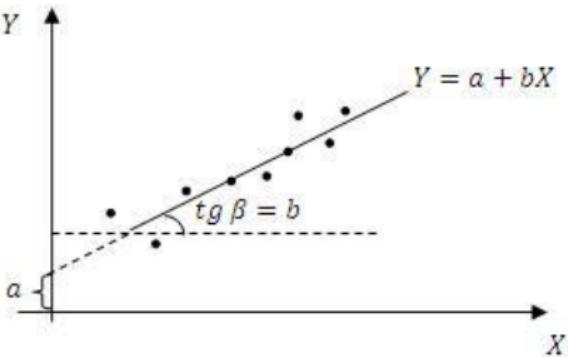
- We can solve it analytically:

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

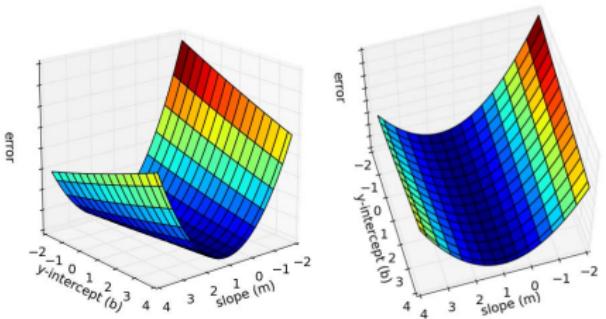
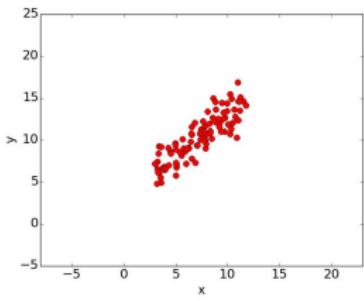
- But the matrix inversion is a very complicated operation
- Gradient descent much faster

Paired regression

- The simplest case: only one feature
- Model: $a(x) = w_0 + w_1x$
- Two parameters: w_0 and w_1
- Quality functional: $Q(w_0, w_1) = \sum_{i=1}^l (w_1 x_i + w_0 - y_i)^2$



Paired regression

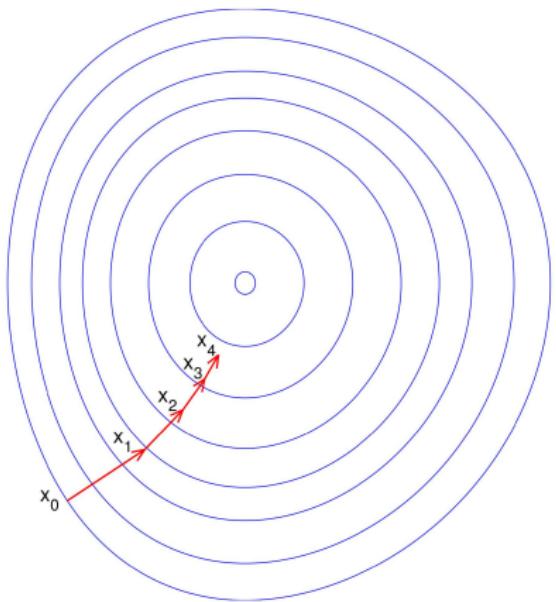


Sample

Quality functional

Gradient descent

- Suppose we have chosen the initial approximation $w^0 = (w_0^0, w_1^0)$
- How to improve it?
- Stepping towards the steepest descent
- That is, towards the anti-gradient!



Gradient descent

- Repeat until convergence

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \eta \nabla Q(\mathbf{w}^{t-1})$$

- convergence: $\|\mathbf{w}^t - \mathbf{w}^{t-1}\| < \epsilon$

Gradient for pair regression

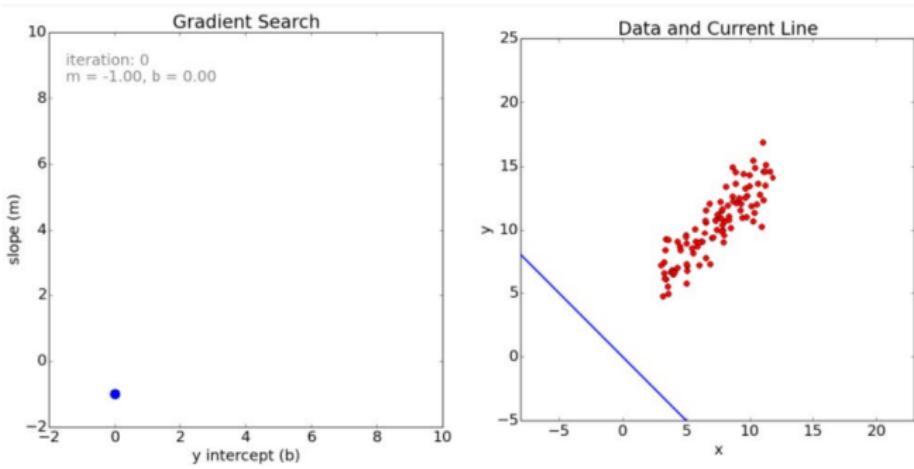
$$Q(w_0, w_1) = \sum_{i=1}^l (w_1 x_i + w_0 - y_i)^2$$

Partial derivatives:

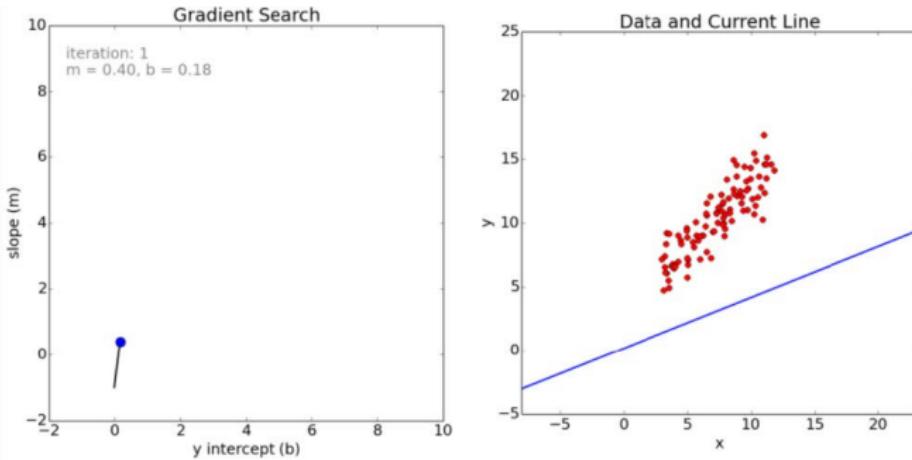
$$\frac{\partial Q}{\partial w_1} = 2 \sum_{i=1}^l (w_1 x_i + w_0 - y_i) x_i$$

$$\frac{\partial Q}{\partial w_0} = 2 \sum_{i=1}^l (w_1 x_i + w_0 - y_i)$$

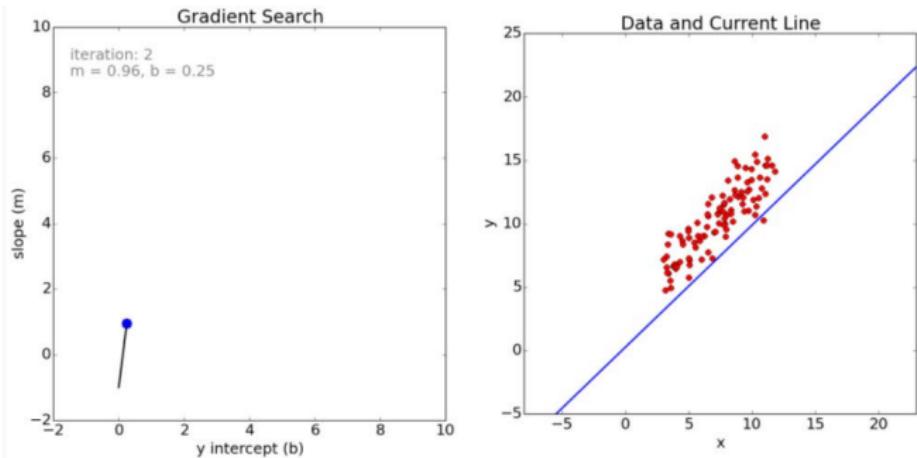
Paired regression



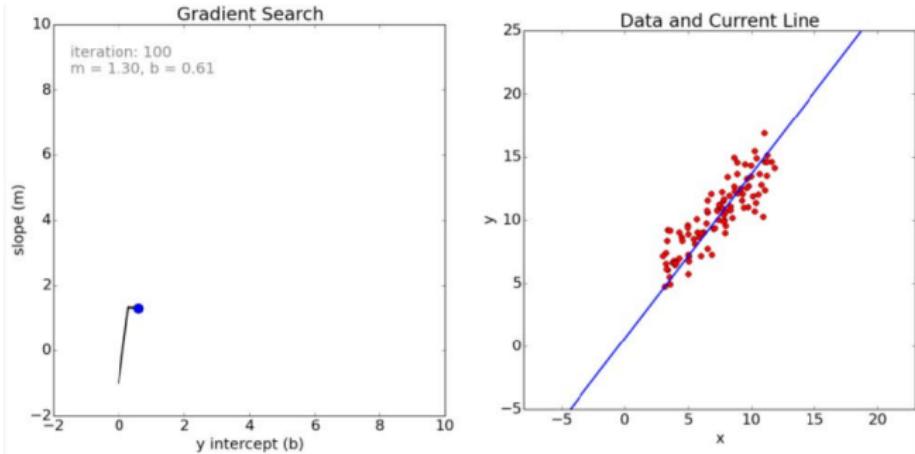
Paired regression



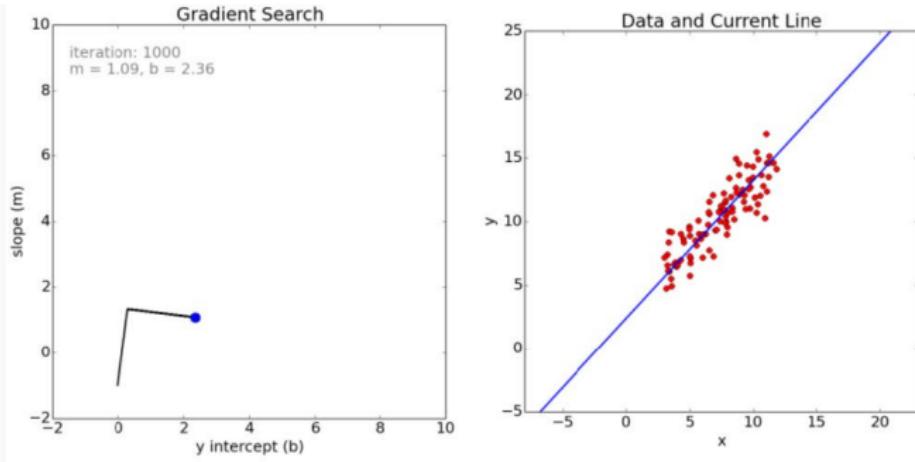
Paired regression



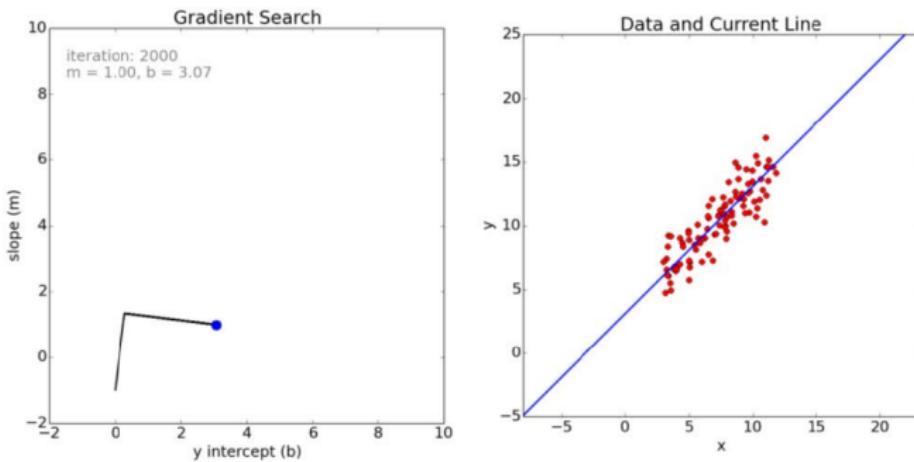
Paired regression



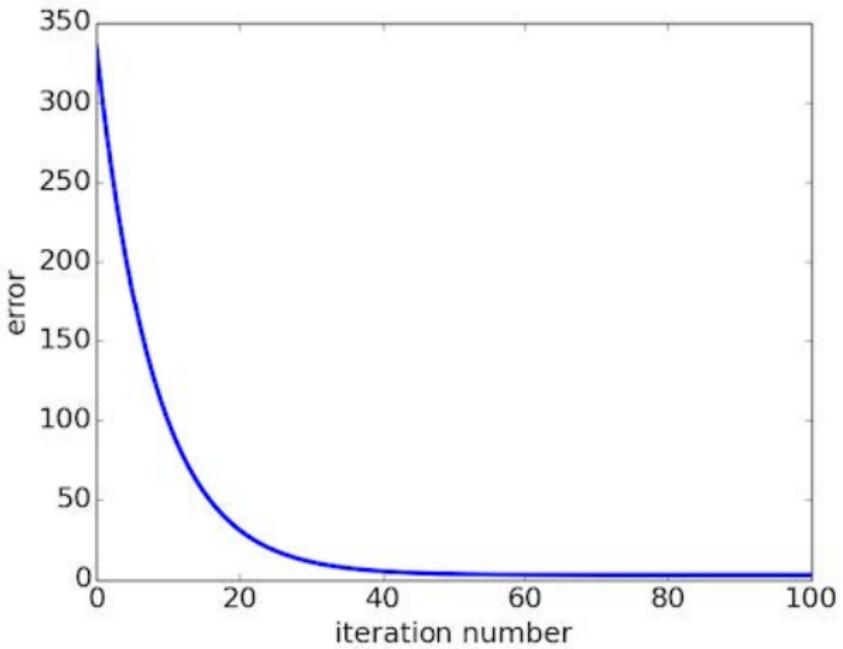
Paired regression



Paired regression

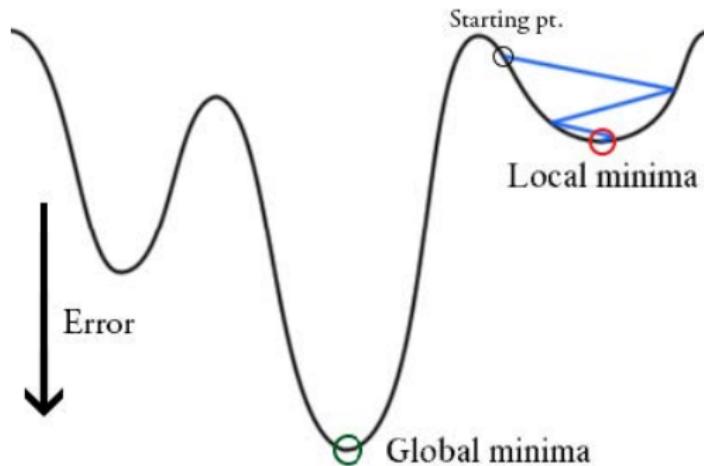


Paired regression



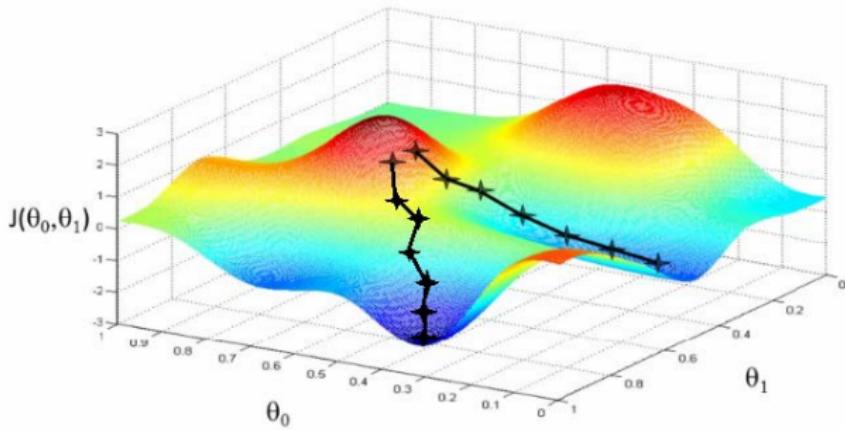
Local minima

Gradient descent can find only local minima



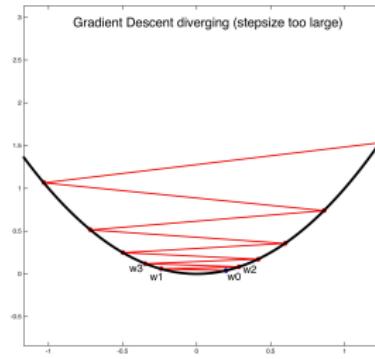
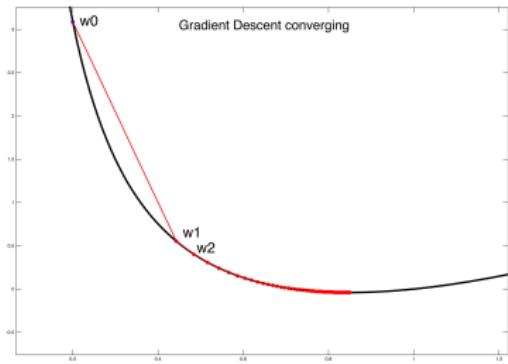
Local minima

- The result depends on the initial approximation
- Multi-start



Size of step

- Choosing of the stepsize η is art



Size of step

- Small step - more chance of convergence, but required more iterations
- Big step - there is a risk of lack of convergence
- Steepest gradient descent:

$$\eta_t = \arg \min_{\eta} Q(w^{t-1} - \eta \nabla Q(w^{t-1}))$$

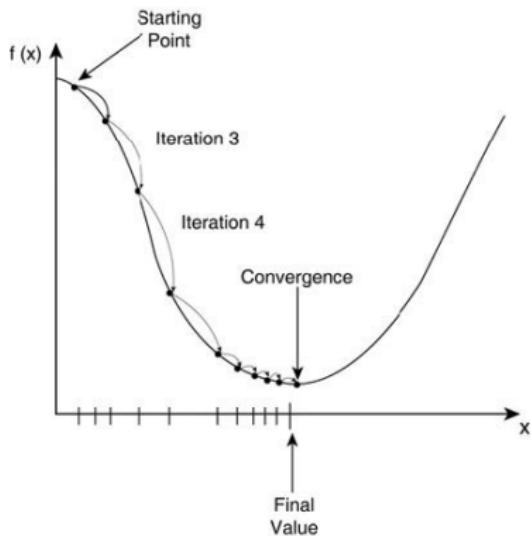
- It is necessary to do a one-dimensional search at each iteration

Size of step

- Usually use heuristics
- The closer to a minimum, the less it is necessary to tread
- Works well

$$\eta_t = \frac{1}{t}$$

- Even better: $\eta_t = \lambda \left(\frac{s}{s+t} \right)^p$



Other optimization methods

- First-order methods - use first derivatives
 - Gradient descent
 - Stochastic gradient descent
 - Quasi-Newtonian methods, BFGS
 - Stochastic Average Gradient, Nesterov momentum, ...
- Second-order methods - use second derivatives
 - The Newton method
- Zero-order methods - without derivatives
 - Coordinate descent
 - Stochastic optimization

Nuances in learning linear regressors

- Selecting the step length η - we try different values
- The sample must be scaled
- Features should not be correlated

An example

$$X = \begin{pmatrix} 1 & 1000 & 5 & 3 & 4 \\ 9 & 9000 & 10 & 5 & 7.5 \\ 5 & 5000 & 1 & 3 & 2 \end{pmatrix}$$

- The task of predicting the profit of the store in the next month
- Consider vectors of the matrix (features) as vectors

An example

$$X = \begin{pmatrix} 1 & 1000 & 5 & 3 & 4 \\ 9 & 9000 & 10 & 5 & 7.5 \\ 5 & 5000 & 1 & 3 & 2 \end{pmatrix}$$

- The first and second signs: $x_2 = 1000x_1$
- First - the total weight of goods in tonnes, the second - in kilograms
- $x_5 = 0.5x_3 + 0.5x_4$
- Fifth - average profit for the last two months
- Third and fourth - profit in the past and the year before last

Linear dependence

Linear dependence - one of the vectors is equal to the sum of the weights of the remaining vectors

- Redundant information
- Extra storage costs
- Damages some machine learning methods

Linear dependence

- Suppose we found a solution w^*
- Modify: $w_1 = w^* + t\alpha$, where t - a number such that:

$$\alpha_1 x_1 + \cdots + \alpha_d x_d = \langle \alpha, x \rangle = 0$$

- The worst case is linearly dependent features
- The answer of the new algorithm on any object:

$$\langle w_1, x \rangle = \langle w^* + t\alpha, x \rangle = \langle w^*, x \rangle + t\langle \alpha, x \rangle = \langle w^*, x \rangle$$

- w_1 is also a solution!

Correlating features

- Also bad!

$$\rho(\xi\eta) = \frac{\mathbb{E}(\xi - \bar{\xi})(\eta - \mathbb{E}\eta)}{\sqrt{\mathbb{D}\xi\mathbb{D}\eta}}$$

$$\rho(x, z) = \frac{\sum_{i=1}^l (x_i - \bar{x})(z_i - \bar{z})}{\sqrt{\sum_{i=1}^l (x_i - \bar{x})^2 \sum_{i=1}^l (z_i - \bar{z})^2}}$$

$$\bar{x} = \frac{1}{l} \sum_{j=1}^l x_j, \bar{z} = \frac{1}{l} \sum_{j=1}^l z_j$$

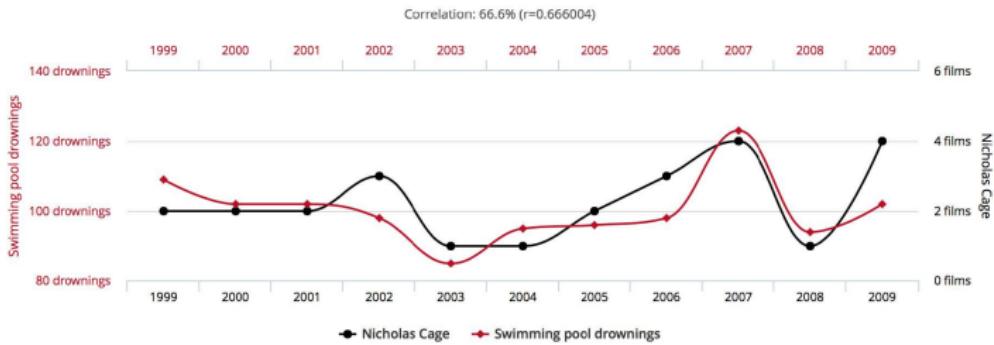
- $\rho(x, z) \in [-1, +1]$
- Very roughly: the closer to +1 or -1, the more accurately equation

$$z = az + b$$

- The measure of linear dependence

Examples

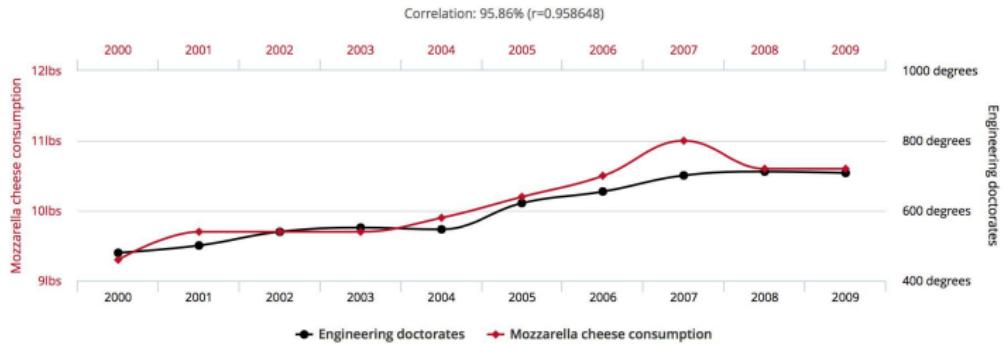
Number of people who drowned by falling into a pool
correlates with
Films Nicolas Cage appeared in



Data sources: Centers for Disease Control & Prevention and Internet Movie Database

Examples

Per capita consumption of mozzarella cheese correlates with Civil engineering doctorates awarded



Data sources: U.S. Department of Agriculture and National Science Foundation

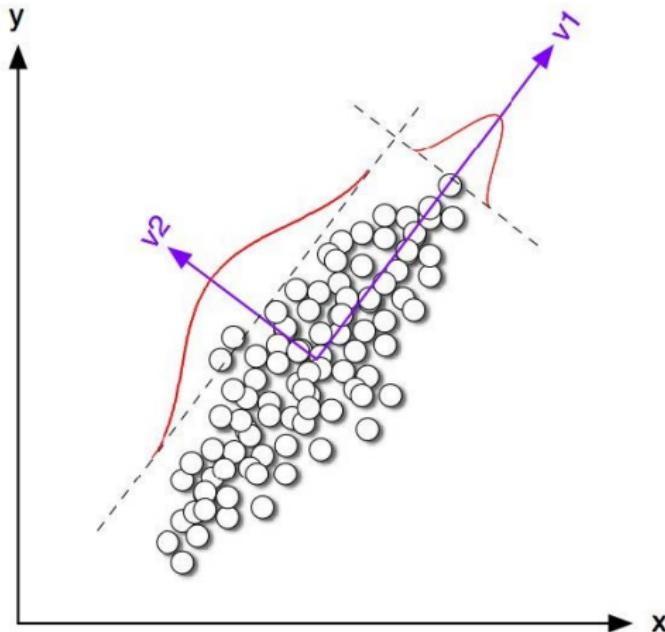
tylenogen.com

A common misconception

- It may seem that correlation follows a causal relationship
- This is not true!
- Correlation means that events often occur together
- But they do not follow each other in any way
- More examples: <http://tylervigen.com/spurious-correlations>

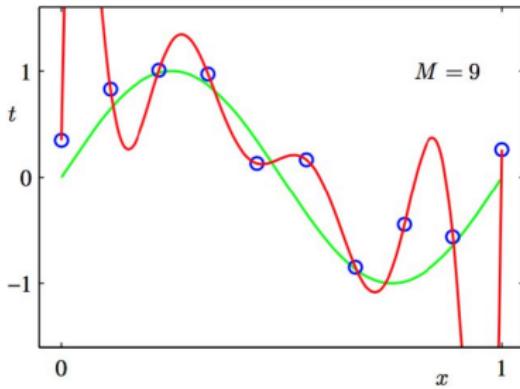
Correlating features

- It is bad if there are correlating signs
- Solution: selection of features or decorrelation



Correlating features

- One feature x
- $a(x) = w_0 + w_1x + w_2x^2 + \cdots + w_9x^9$



- Coefficients:

$$a(x) = 0.5 + 13458922x - 43983740x^2 + \cdots + 2740x^9$$

- Large coefficients - a symptom of retraining (empirical observation)

Regularization

- We will fine for the big weights!
- Functionality:

$$Q(w, x) = \frac{1}{l} \sum_{i=1}^l (\langle w, x_i \rangle - y_i)^2 \rightarrow \min_w$$

- Regularizer:

$$\|w\|^2 = \sum_{j=1}^d w_j^2$$

- The regularized error functional:

$$\frac{1}{l} \sum_{i=1}^l (\langle w, x_i \rangle - y_i)^2 + \lambda \|w\|^2 \rightarrow \min_w$$

- Still smooth and convex

Regularization

- λ - new parameter, it is necessary to select
- High λ - simple models
- Low λ - risk of retraining
- It is necessary to balance
- Selection of λ - using cross-validation

The meaning of regularization

- Minimization of a regularized functional is equivalent to solving a conditional problem:

$$\begin{cases} \frac{1}{l} \sum_{i=1}^l (\langle w, x_i \rangle - y_i)^2 + \lambda \|w\|^2 \rightarrow \min_w \\ \|w\|^2 \leq C \end{cases}$$

L_1 regularization

- L_1 regularizer:

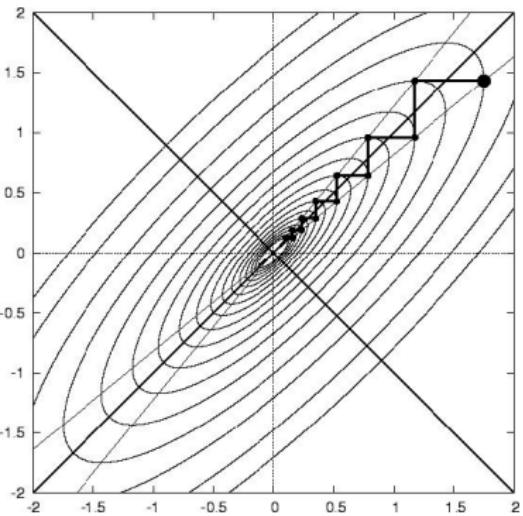
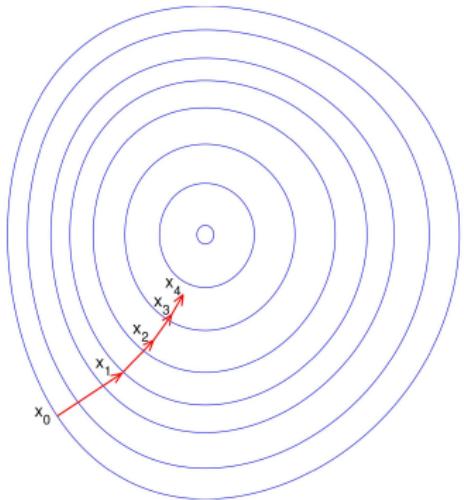
$$\|w\|_1 = \sum_{j=1}^d |w_j|$$

- The regularized error functional:

$$\frac{1}{l} \sum_{i=1}^l (\langle w, x_i \rangle - y_i)^2 + \lambda \|w\|_1 \rightarrow \min_w$$

- Functionality becomes non-smooth
- It is more difficult to optimize
- But the selection of characteristics
- Some of the weights in the solution will be zero

Scaling the sample



Scaling the sample

- Task: Will the grant application be approved?
- 1st feature: how many successful applications did the applicant have before
- 2nd feature: the year of birth of the applicant
- Scale: units and thousands
- All features must have the same scale

Scaling the sample

- We scale the j -th feature
- Calculate the mean and standard deviation of the trait on the training sample:

$$\mu_j = \frac{1}{I} \sum_{i=1}^I x_j^i$$

$$\sigma_j = \sqrt{\frac{1}{I} \sum_{i=1}^I (x_j^i - \mu_j)^2}$$

- We scale the j -th feature
- Subtract from each characteristic value the average and divide by the standard deviation:

$$x_i^j := \frac{x_i^j - \mu_j}{\sigma_j}$$