ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

Introduction to Machine Learning

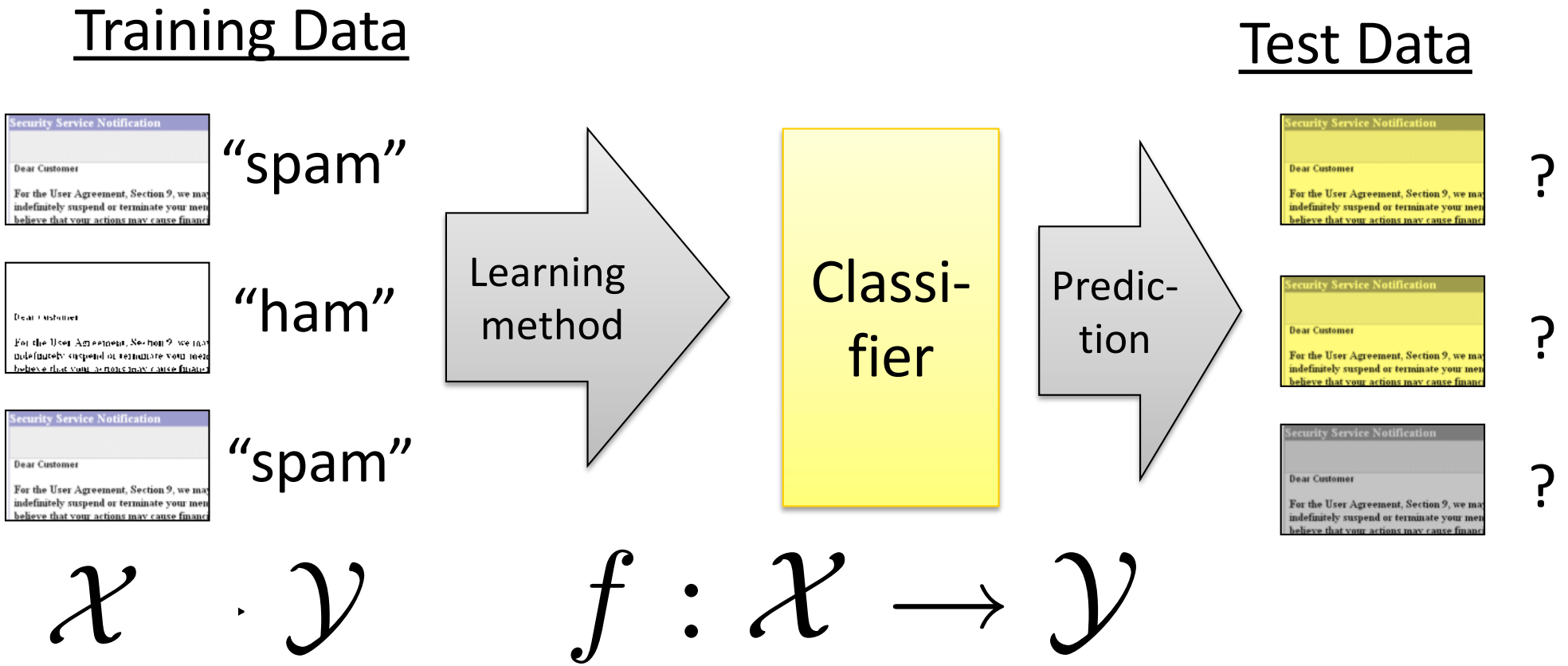
Linear Regression

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

Announcements

- First recordings online:
  - https://www.video.ethz.ch/lectures/d-infk/2019/spring/252-0220-00L.html
- Waitlist updates

Basic Supervised Learning Pipeline



Regression

- Instance of supervised learning
- **Goal**: Predict real valued labels (possibly vectors)
- Examples:

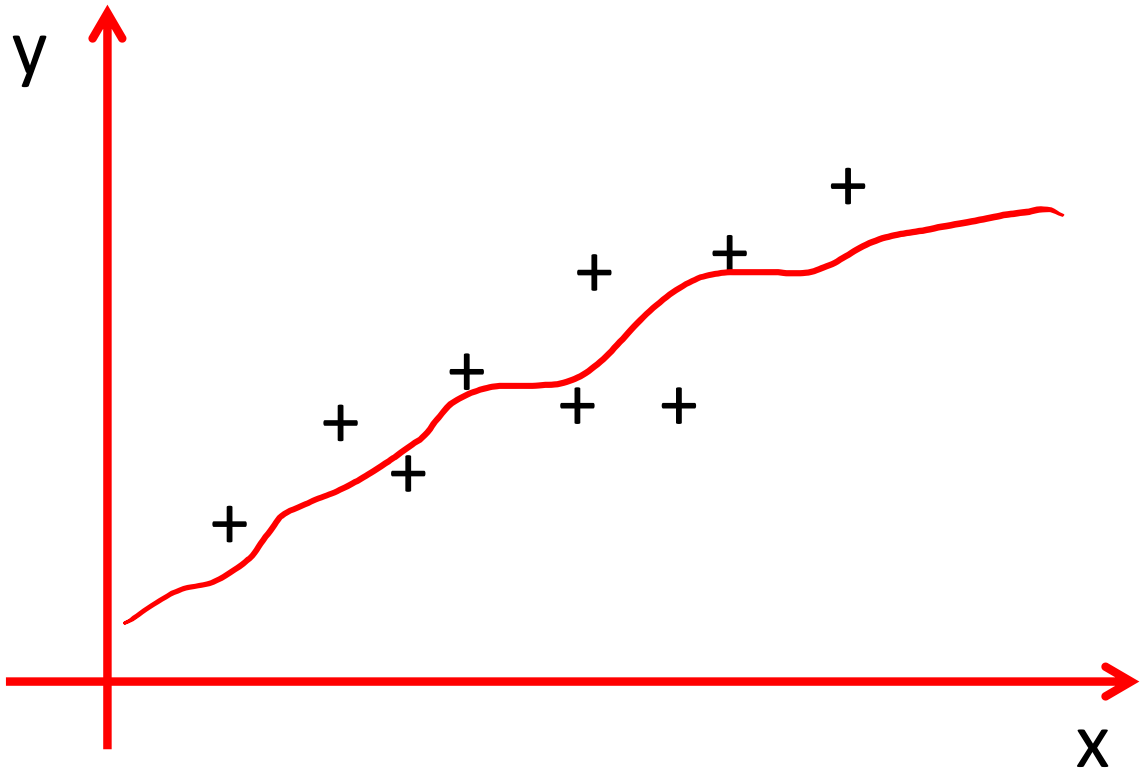
X	**Y**
Flight route	Delay (minutes)
Real estate objects	Price
Customer & ad features	Click-through probability

Running example: Diabetes

[Efron et al '04]

- Features X:
  - Age
  - Sex
  - Body mass index
  - Average blood pressure
  - Six blood serum measurements (S1-S6)
- Label (target) Y: quantitative measure of disease progression

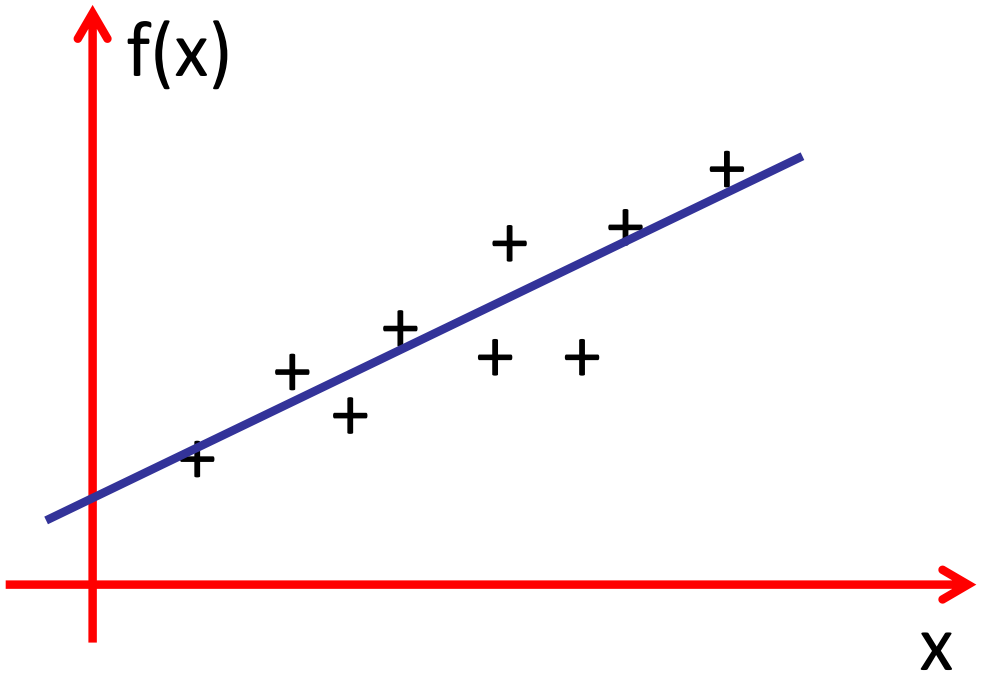
Regression

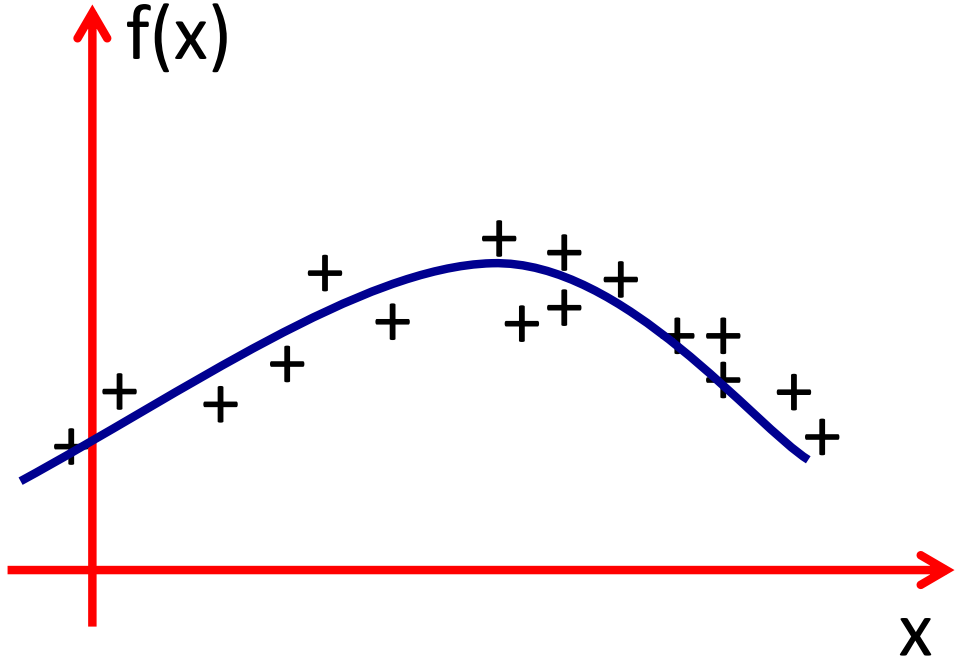


- **Goal**: learn real valued mapping $f : \mathbb{R}^d \rightarrow \mathbb{R}$

Important choices in regression

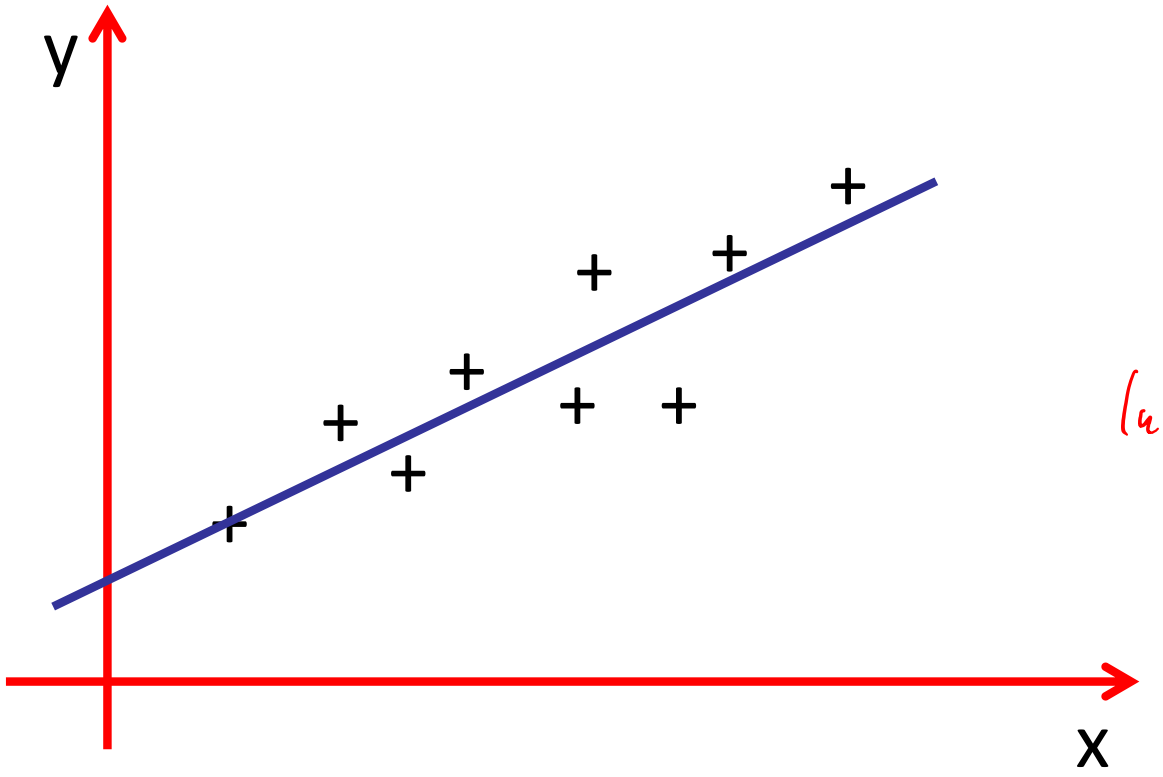
- What types of functions f should we consider? Examples





- How should we measure goodness of fit?

Example: linear regression



$y \approx f(x)$

↑ f is linear (affine) in x

In 1-d: $f(x) = ax + b$

2-d: $f(x) = a x_1 + b x_2 + c$

d-d: $f(x) = w_1 \cdot x_1 + \dots + w_d \cdot x_d + w_0$

$= \sum_i w_i x_i + w_0$

$= w^T x + w_0$

$w = [w_1 \dots w_d]^T$

$x = [x_1 \dots x_d]$

Homogeneous representation

$$w^T x + w_0 = \tilde{w}^T \tilde{x}$$

$$w = [w_1 \dots w_d]^T$$

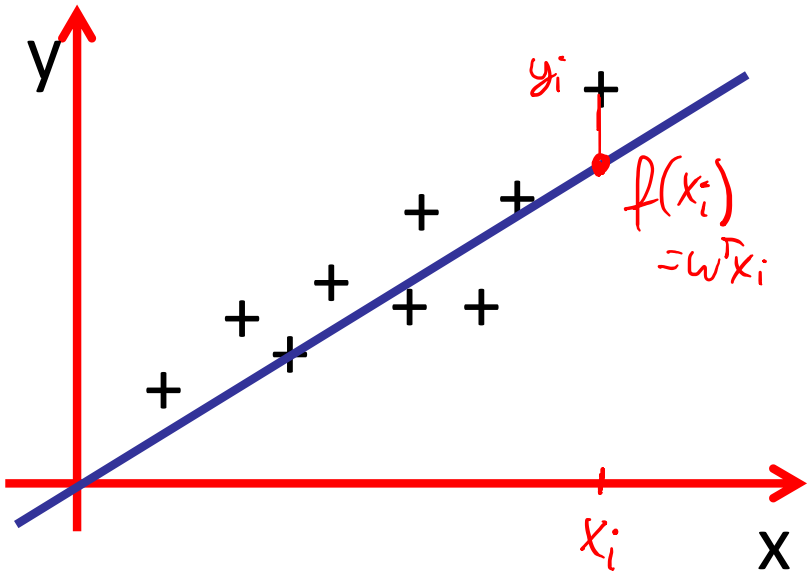
$$x = [x_1 \dots x_d]^T$$

$$\tilde{w} = [w_1 \dots w_d \; w_0]^T$$

$$\tilde{x} = [x_1 \dots x_d \; 1]^T$$

Quantifying goodness of fit

$$D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \qquad \mathbf{x}_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$



Residual $r_i = y_i - w^T x_i$

Cost $\hat{R}(w) = \sum_{i=1}^{n} r_i^2$

$= \sum_{i=1}^{n} (y_i - w^T x_i)^2$

Note: we have made 2 decisions here:

1) quantify error via squared residuals

2) sum over these

Least-squares linear regression optimization

[Legendre 1805, Gauss 1809]

- Given data set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

- How do we find the optimal weight vector?

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

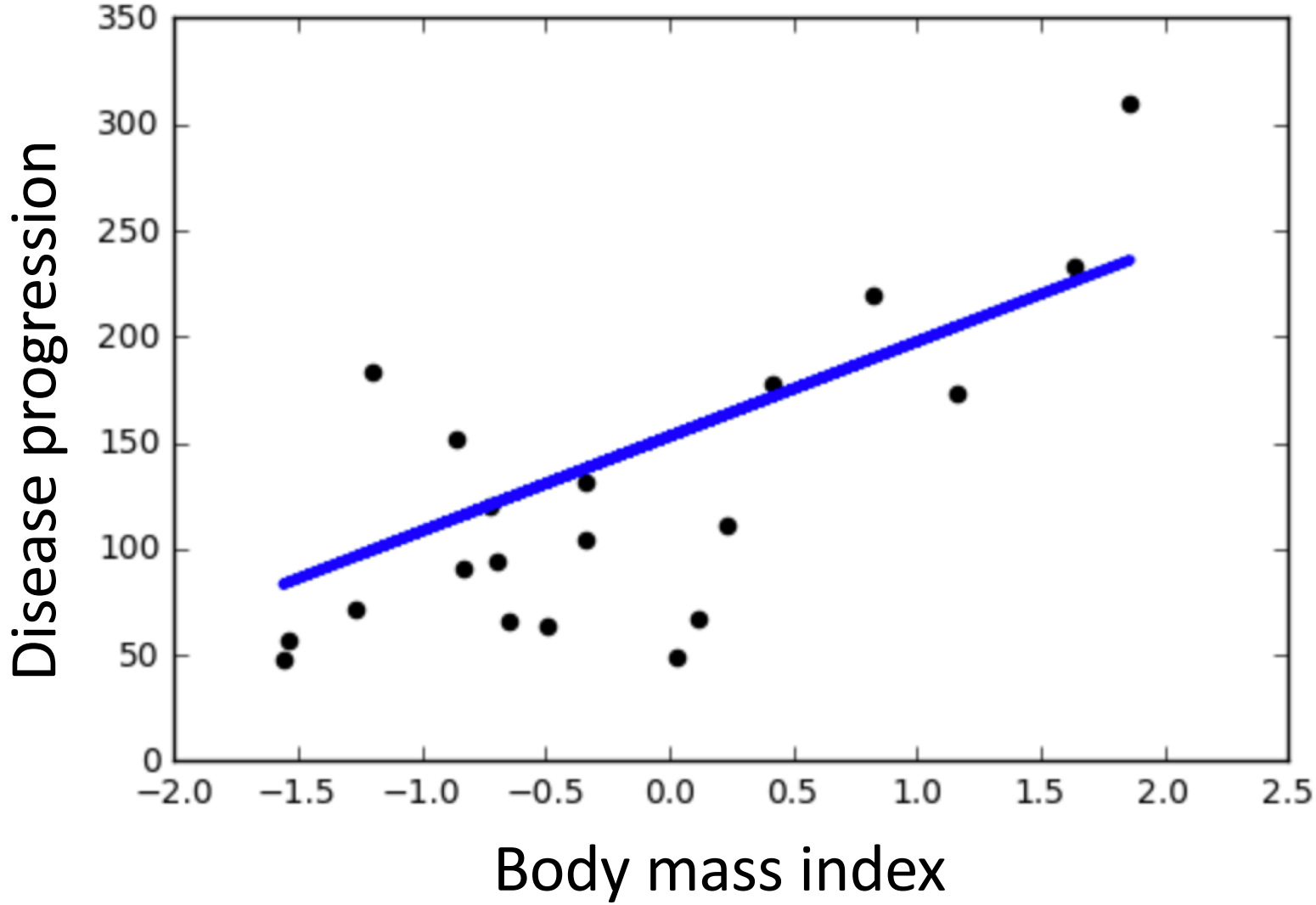
How to solve? Example: Scikit Learn

```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training set
regr.fit(X_train, Y_train)

# Make predictions on the testing set
Y_pred = regr.predict(X_test)
```

Demo


Disease progression
350
300
250
200
150
100
50
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
Body mass index

Method 1: Closed form solution

- The problem $$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

  can be solved in closed form:

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

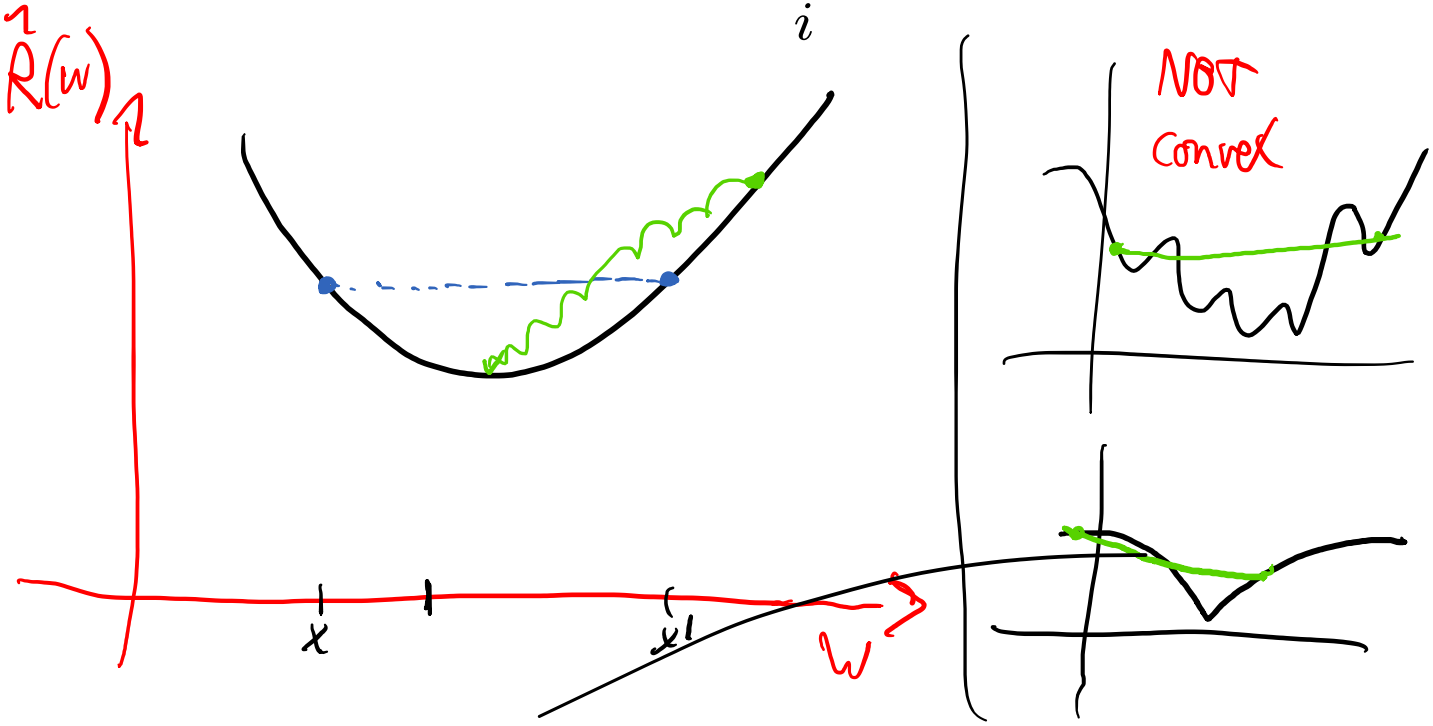
- Hereby:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & & \\ x_{n1} & \cdots & x_{nd} \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Method 2: Optimization

- The objective function $\hat{R}(\mathbf{w}) = \sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2$

is convex!



$f: \mathbb{R}^d \to \mathbb{R}$ convex iff $\forall x, x' \in \mathbb{R}^d$, $\lambda \in [0,1]$ it holds that

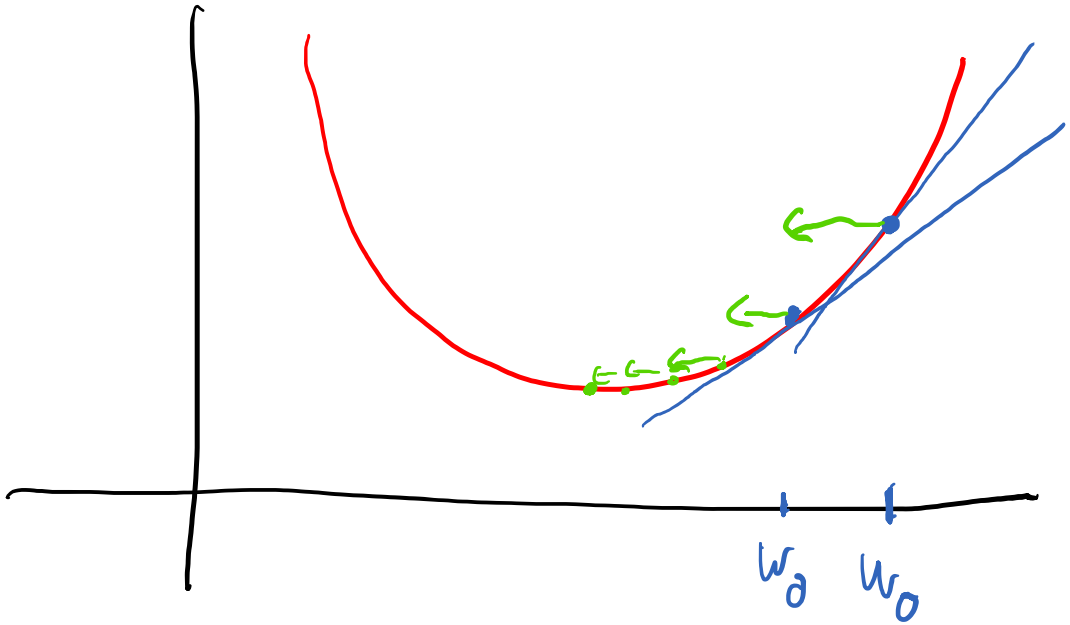
$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$

Gradient Descent

$\frac{1}{2}$ for Least squares

- Start at an arbitrary $\mathbf{w}_0 \in \mathbb{R}^d$
- For t=1,2,... do $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \hat{R}(\mathbf{w}_t)$

- Hereby, η_t is called learning rate



Convergence of gradient descent

- Under mild assumptions, if step size sufficiently small, gradient descent converges to a stationary point (gradient = 0)
- For convex objectives, it therefore finds the optimal solution!

- In the case of the squared loss, constant stepsize ½ converges linearly

Can find x_t s.t. $f(x_t) - \min_x f(x) \leq \varepsilon$ in $O(\log \frac{1}{\varepsilon})$ iterations

Computing the gradient

$$\nabla \hat{R}(w) = \left[\frac{\partial}{\partial w_1} \hat{R}(w) \quad \cdots \quad \frac{\partial}{\partial w_d} \hat{R}(w) \right]$$

In 1-d: $\nabla \hat{R}(w) = \frac{d}{dw} \hat{R}(w) = \frac{d}{dw} \sum_{i=1}^{n} (y_i - w \cdot x_i)^2$

$$= \sum_{i=1}^{n} \frac{d}{dw} (y_i - w \cdot x_i)^2$$

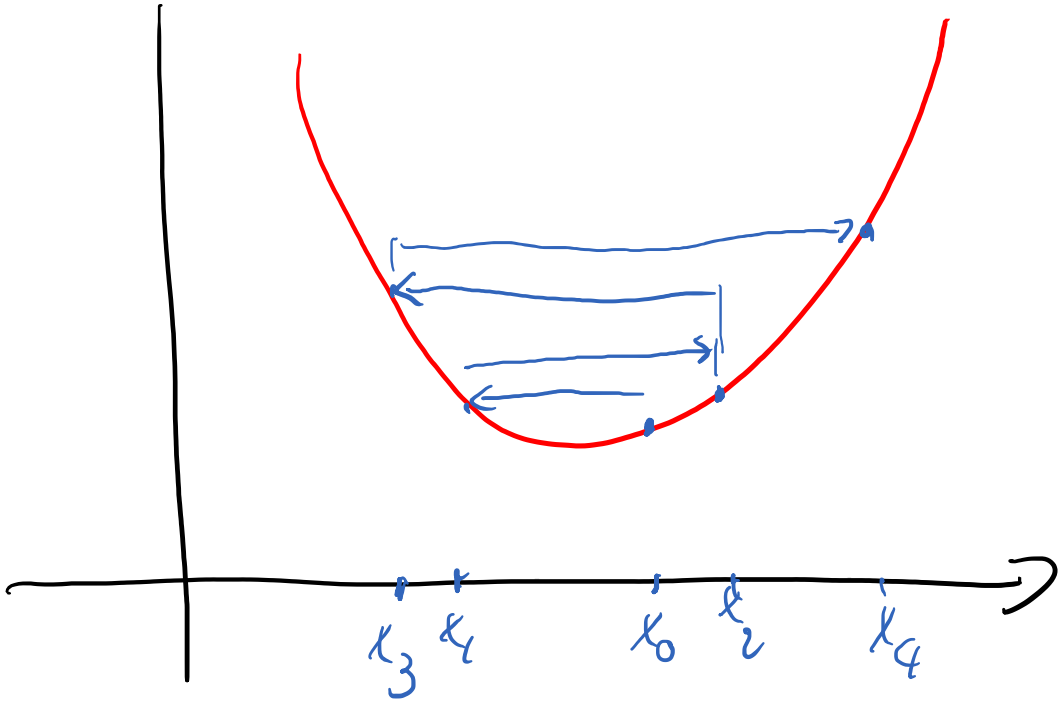
$$= \sum_{i=1}^{n} \underbrace{2(y_i - w \cdot x_i)}_{r_i} (-x_i) = -2 \sum_i r_i \cdot x_i$$

In d-d: $\nabla \hat{R}(w) =$ $= -2 \sum_i r_i \cdot x_i^T$

Demo: Gradient descent

Choosing a stepsize

- What happens if we choose a poor stepsize?



Adaptive step size

- Can update the step size adaptively. Examples:
- 1) Via line search (optimizing step size every step)

Sps. $g_t = \nabla \hat{R}(w_t)$

pick $\eta_t \in \operatorname{argmin}_{\eta \in (0,\infty)} \hat{R}(w_t - \eta \cdot g_t)$

- 2) „Bold driver“ heuristic
  - If function decreases, increase step size:
  - If function increases, decrease step size:

e.g. $c_{acc} = 1.1$, $c_{dec} = 0.5$

If $\hat{R}(w_t - \eta_t g_t) < \hat{R}(w_t) \Rightarrow \eta_{t+1} = \eta_t \cdot c_{acc}$

If $\hat{R}(\underbrace{w_t - \eta_t g_t}_{w_{t+1}}) > \hat{R}(w_t) \Rightarrow \eta_{t+1} = \eta_t \cdot c_{dec}$

Demo: Gradient Descent for Linear Regression

Gradient descent vs closed form

- Why would one ever consider performing gradient descent, when it is possible to find closed form solution?

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

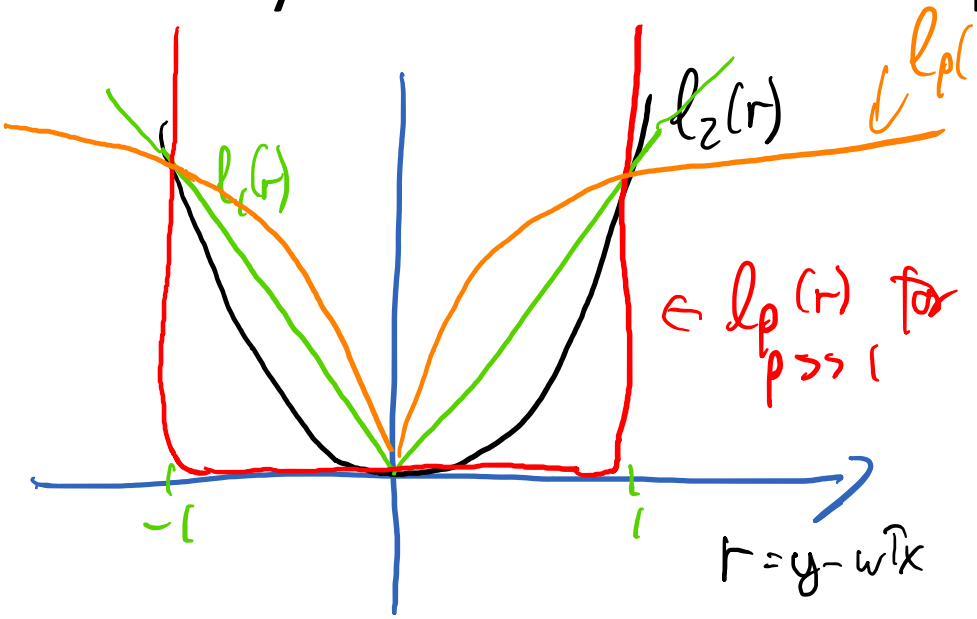
$O(nd^2)$, solve lin. sys. $\rightarrow O(d^3)$

GD: Calc. $\nabla \hat{R}(w) = \sum_i (y_i - w^T x_i) x_i \Rightarrow O(n \cdot d)$, need $O(\log \frac{1}{\varepsilon})$ this

- Computational complexity
- May not need an optimal solution
- Many problems don‘t admit closed form solution

Other loss functions

- So far: Measure goodness of fit via squared error
- Many other loss functions possible (and sensible!)



Least squares: $\ell_2(r) = r^2$

Alternatives:

$\ell_1(r) = |r|$

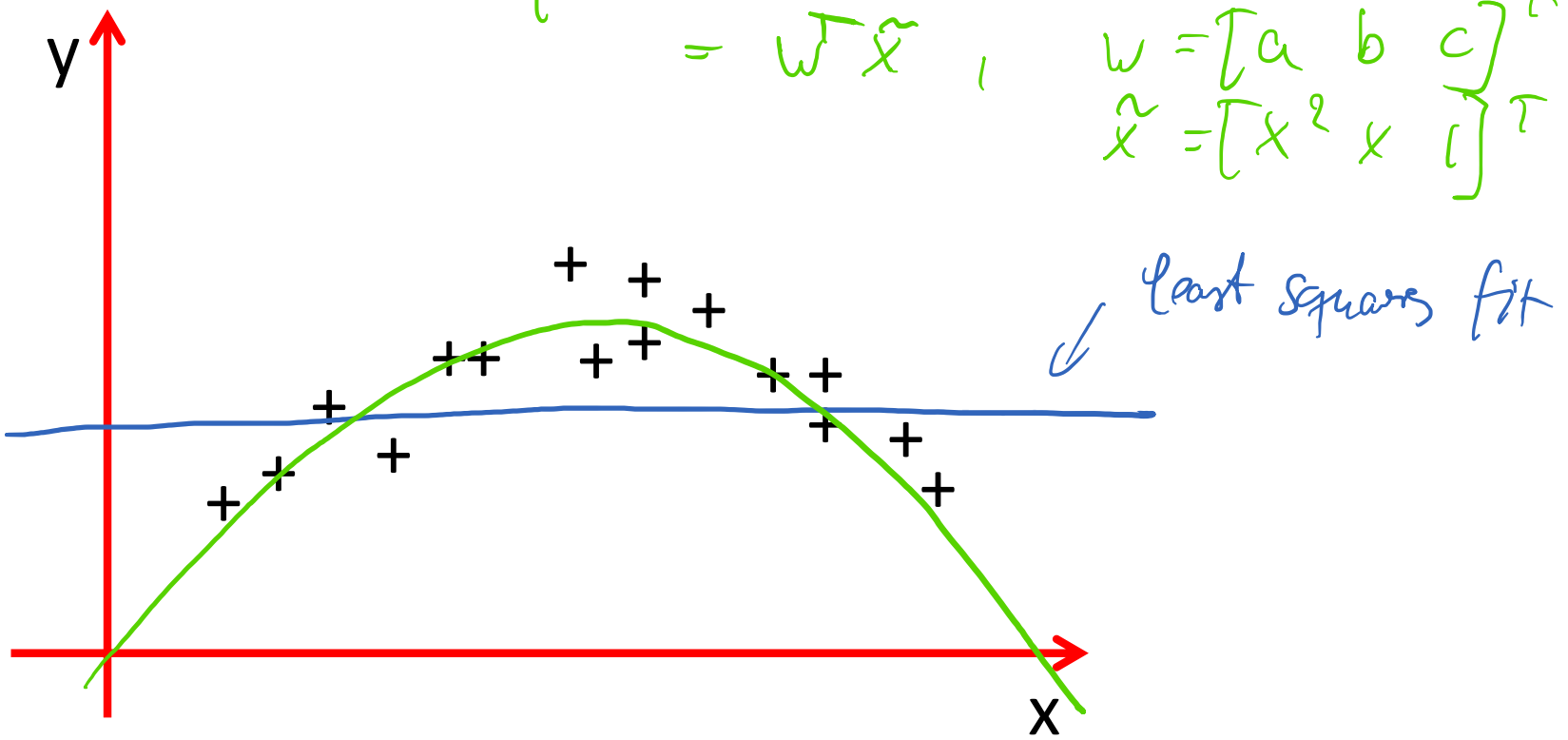
$\ell_p(r) = |r|^p$

$$\hat{R}(w) = \sum_{i=1}^{n} \ell_p(y_i - w^T x_i)$$

Fitting nonlinear functions

- How about functions like this:

$$f(x) = ax^2 + bx + c$$
$$= w^T \tilde{x}, \quad w = [a \; b \; c]^T$$
$$\tilde{x} = [x^2 \; x \; 1]^T$$



Linear regression for polynomials

We can fit non-linear functions via linear regression, using nonlinear features of our data (basis functions)

$$f(\mathbf{x}) = \sum_{i=1}^{D} w_i \phi_i(\mathbf{x})$$

$x \mapsto \tilde{x} = \phi(x)$

$D = |\phi(x)|$

In 1-d: $\phi(x) = [1, x, x^2, \ldots x^k]$

In 2-d: $\phi(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1 \cdot x_2, \ldots]$

In d-d: $\phi(x)$ = vector of all monomials of degree up to k in $x_1 \ldots x_d$