Introduction to Machine Learning

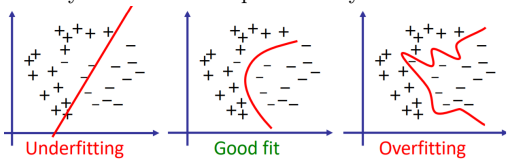
Marco Dober

21st February 2019

1 Introduction

There exists two basic forms of learning:

- **Supervised learning**
  You have data with x- and y-labels → try to fit model → predict with found model new cases.
  Examples of supervised learning:
  – Classification
  – Regression
  – Structured prediction, ...
  Model selection is very important and a trade off between complexity vs. goodness of fit. Ideal models are statistically and computationally efficient.



- **Unsupervised learning**
  "Learning without labels" (no y-label on data).
  Examples:
  – Clustering (e.g., unsupervised classification)
  – Dimension reduction (e.g., unsupervised regression)
  – Generative modeling
  Common goals:
  – Compact representation / compression of data sets
  – Identification of latent variables

Example of unsupervised learning, Clustering: Input data with no labels and assign data to clusters

Other models of learning: semi-supervised, transfer, active, online, reinforcement, ...
The key challenge in ML is:
- Trading goodness of fit and model complexity
- Also the **representation of data** is of key importance.

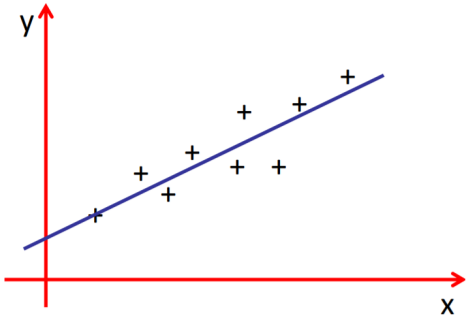
2 Regression

Is an instance of supervised learning.
Goal: Predict real valued labels.
Important choices in regression:

- What **types of functions** should we consider?
- How should we measure **goodness of fit**?

2.1 Linear Regression

$y \approx f(x)$, f is linear (affine) in x



- 1-d: $f(x) = ax + b$, (line)
- 2-d: $f(x) = ax_1 + bx_2 + c$, (area)
- d-d: $f(x) = w_1x_1 + \cdots + w_dx_d + w_0 = \sum_{i=1}^{d} w_ix_i + w_0 = w^Tx + w_0$
  $w = [w_1, \ldots, w_d]^T$, $x = [x_1, \ldots, x_d]^T$

Homogeneous representation (no offset):
$w^Tx + w_0 = \tilde{w}^T\tilde{x}$
$\tilde{w} = [w_1, \ldots, w_d, w_0]^T$, $\tilde{x} = [x_1, \ldots, x_d, 1]^T$

2.1.1 Quantifying goodness of fit

Calculate **residuals**, difference from fitet-point and data-point. Different ways to compute loss function (abs-value, square, ...), see below. Most famous is the **least-square optimization**:
$r_i = y_i - f(x_i) = y_i - w^Tx_i$
Cost $\hat{R}(w) = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n}(y_i - w^Tx_i)^2$
How do we find the optimal weight vector w^*?

$$w^* = \arg\min_w \sum_{i=1}^{n}(y_i - w^Tx_i)^2$$

- **Closed form**
The problem from above can be solved in closed form!

$$w^* = (X^TX)^{-1}X^Ty$$

$$X = \begin{pmatrix} x_{1,1} & \ldots & x_1,d \\ \ldots & \ldots & \ldots \\ x_{n,1} & \ldots & x_{n,d} \end{pmatrix}, y = \begin{pmatrix} y_1 \\ \ldots \\ y_n \end{pmatrix}$$

- **Optimization**
The objective function is **convex**. In convex functions local minimum = global minimum!

Gradient descent:
- Start at arbitrary $w_0 \in \mathbb{R}^d$
- For $t = 1, 2, \ldots$ do

$$w_{t+1} = w_t - \eta_t \nabla\hat{R}(w_t)$$

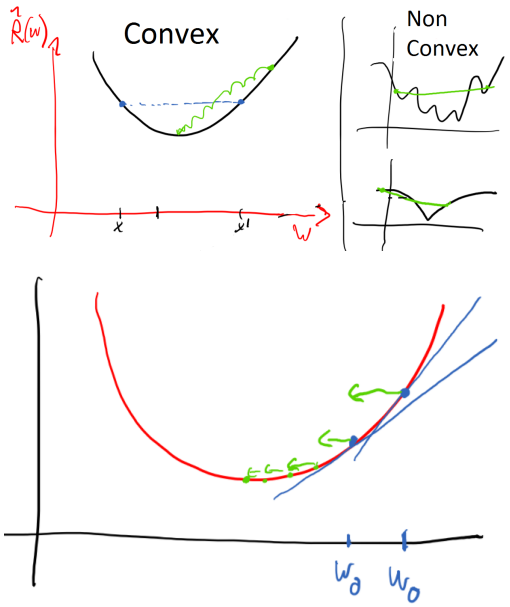
If I descend along the gradient I will for sure find a minimum (step size / learning rate η_t sufficiently small). For convex objectives it finds an optimum. For the squared loss, **constant step size 0.5 converges linearly**!
$\nabla\hat{R}(w) = \cdots = -2\sum_i^n r_ix_i^T$, (in all dimensions)

Adaptive step size
- step size too big: oscillation, divergence
- step size too small: long computation
Different ways to update step size adaptively:

1. Line search (optimizing step size every step)
   $g_t = \nabla\hat{R}(w_t)$
   $\eta_t = \arg\min_{\eta\in(0,\text{inf})} \hat{R}(w_t - \eta g_t)$
2. Bold driver heuristic
   - If function decreases, increase step size:
     $\hat{R}(w_t - \eta_t g_t) < \hat{R}(w_t) \Rightarrow \eta_{t+1} = \eta_t \cdot c_{acc.}$
   - If function increases, decrease step size:
     $\hat{R}(w_t - \eta_t g_t) > \hat{R}(w_t) \Rightarrow \eta_{t+1} = \eta_t \cdot c_{dec.}$
     $c_{acc.} > 1$ & $c_{dec.} < 1$

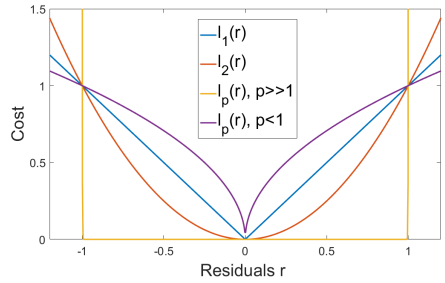
Gradient descent vs. closed form (many times gradient descent (optimization) is better than closed form):

- computational complexity:
  closed form can get very ugly (shit-ton of calculations)
- May not need an optimal solution:
  Uncertainty in data, so it makes no sense to find the optimal solution, I can stop earlier with optimization.
- Many problems don't admit closed form solution



2.1.2 Other loss functions

so far we used squared error to measure goodness of fit, but there are many other possibilities.



$l_p(r) = |r|^p = |y_i - w^Tx_i|^p$, with the parameter p you can vary your optimization.

- $p = 1$: weight all r the same, less sensible to outliers
- $p = 2$: closed form sol., most common
- $p >> 1$: make tolerance band of allowed errors
- $p << 1$: almost no weight to outliers

$\hat{R}(w) = \sum_{i=1}^{n} l_p(y_i - w^Tx_i)$

2.2 Fitting nonlinear functions

We can fit nonlinear functions via linear regression using nonlinear features of our data (basis functions). With a nonlinear transformation we transform data to be able to fit linearly.
....