

Disclaimer

This summary is part of the lecture “Introduction to Machine Learning” by Prof. Krause (FS20). It is based on the lecture slides.

Please report errors to doberm@student.ethz.ch such that others can benefit as well.

The upstream repository can be found at <https://github.com/mrrebod/Summaries>

Introduction to Machine Learning

Marco Dober

1st March 2020

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.

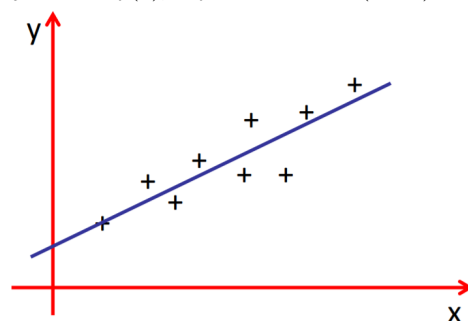
Goal: Predict real valued labels.

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: $f(x) = w_1x_1 + \dots + w_dx_d + w_0 = \sum_{i=1}^d w_i x_i + w_0 = \mathbf{w}^T \mathbf{x} + w_0$
 $\mathbf{w} = [w_1, \dots, w_d]^T$, $\mathbf{x} = [x_1, \dots, x_d]^T$

Homogeneous representation (no offset):

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

$$\tilde{\mathbf{w}} = [w_1, \dots, w_d, w_0]^T, \tilde{\mathbf{x}} = [x_1, \dots, x_d, 1]^T$$

2.1.1 Quantifying goodness of fit

Calculate **residuals**, difference from fit-point and data-point. Different ways to compute loss function (abs-value, square, ...), see below. Most famous is the **least-square optimization**:

$$r_i = y_i - f(\mathbf{x}_i) = y_i - \mathbf{w}^T \mathbf{x}_i$$

$$\text{Cost } \hat{R}(\mathbf{w}) = \sum_{i=1}^n r_i^2 = \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

How do we find the optimal weight vector \mathbf{w}^* ?

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

• Closed form

The problem from above can be solved in closed form!

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

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

• Optimization

The objective function $\hat{R}(\mathbf{w}) = \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i)^2$ is **convex**. In convex functions local minimum = global minimum!

Gradient descent:

- Start at arbitrary $\mathbf{w}_0 \in \mathbb{R}^d$
- For $t = 1, 2, \dots$ do

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \hat{R}(\mathbf{w}_t)$$

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!**

$$\nabla \hat{R}(\mathbf{w}) = \dots = -2 \sum_{i=1}^n r_i \mathbf{x}_i^T, \text{ (in all dimensions)}$$

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)

$$g_t = \nabla \hat{R}(\mathbf{w}_t)$$

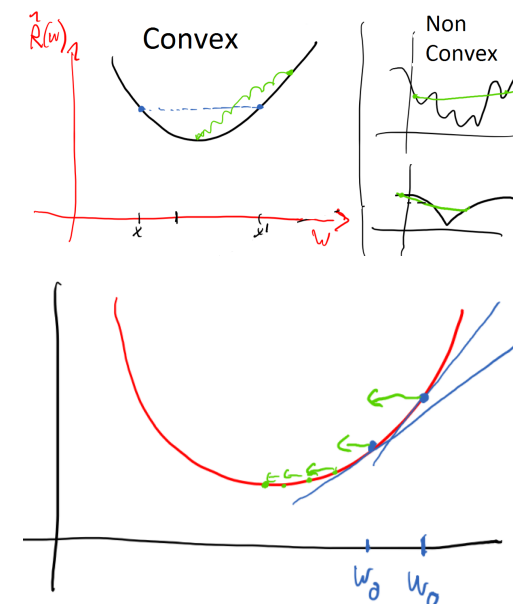
$$\eta_t = \arg \min_{\eta \in (0, \inf)} \hat{R}(\mathbf{w}_t - \eta g_t)$$

2. Bold driver heuristic

- If function decreases, increase step size:
 $\hat{R}(\mathbf{w}_t - \eta_t g_t) < \hat{R}(\mathbf{w}_t) \Rightarrow \eta_{t+1} = \eta_t \cdot c_{acc.}$
- If function increases, decrease step size:
 $\hat{R}(\mathbf{w}_t - \eta_t g_t) > \hat{R}(\mathbf{w}_t) \Rightarrow \eta_{t+1} = \eta_t \cdot c_{dec.}$
 $c_{acc.} > 1$ & $c_{dec.} < 1$

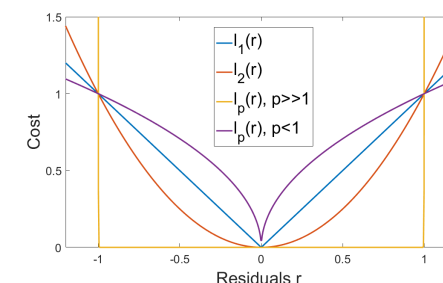
Gradient descent vs. closed form (many times gradient descent (optimization) is better than closed form):

- 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.



$l_p(r) = |r|^p = |y_i - \mathbf{w}^T \mathbf{x}_i|^p$, 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

$$\hat{R}(\mathbf{w}) = \sum_{i=1}^n l_p(y_i - \mathbf{w}^T \mathbf{x}_i)$$

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.