ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

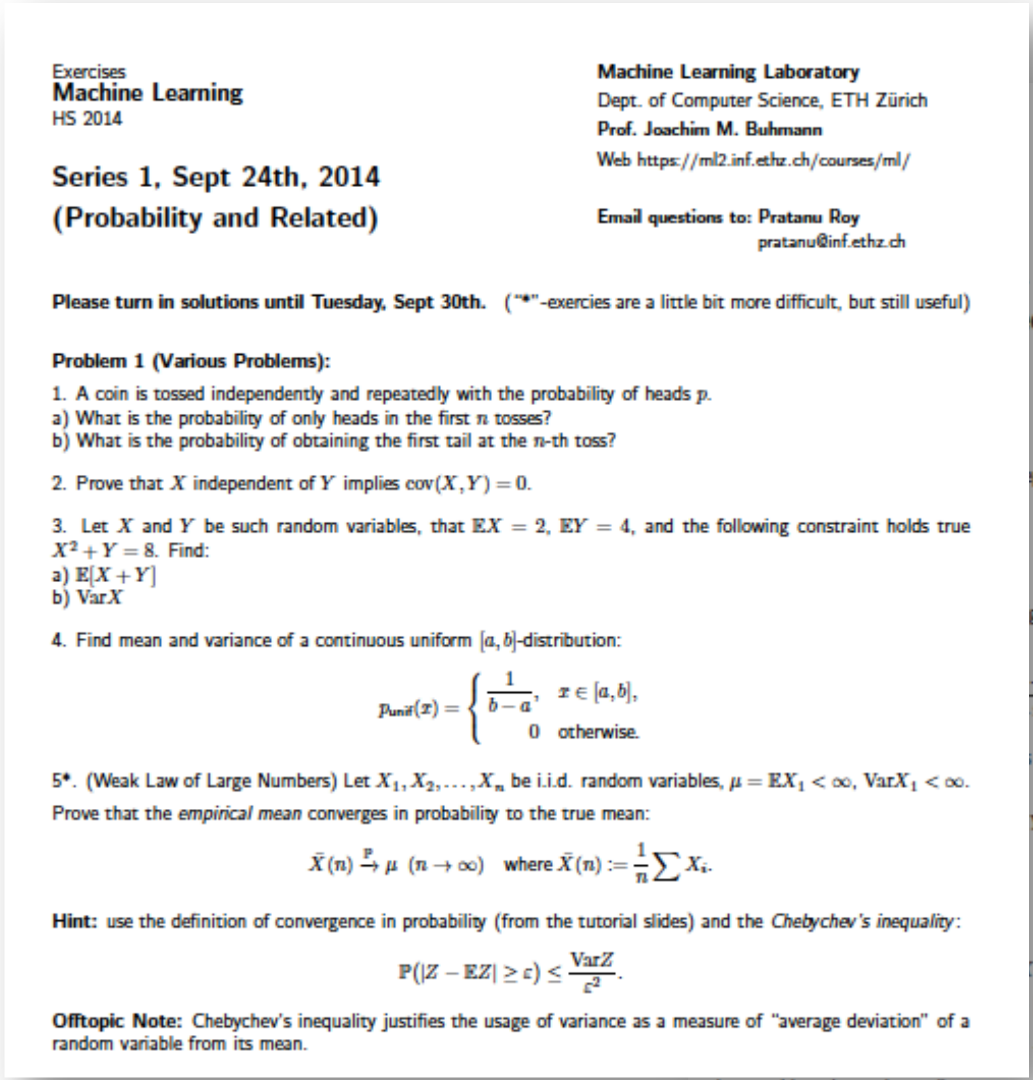
inf | Informatik Computer Science

Machine Learning

Linear regression

Ekaterina (Kate) Lomakina
Darko Makreshanski
October 8-10

Series 1



Exercises
Machine Learning
HS 2014

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Joachim M. Buhmann
Web https://ml2.inf.ethz.ch/courses/ml/

Series 1, Sept 24th, 2014
(Probability and Related)

Email questions to: Pratanu Roy
pratanu@inf.ethz.ch

Please turn in solutions until Tuesday, Sept 30th. ("*"-exercies are a little bit more difficult, but still useful)

Problem 1 (Various Problems):

1. A coin is tossed independently and repeatedly with the probability of heads p.
a) What is the probability of only heads in the first n tosses?
b) What is the probability of obtaining the first tail at the n-th toss?

2. Prove that X independent of Y implies $\operatorname{cov}(X,Y) = 0$.

3. Let X and Y be such random variables, that $\mathbb{E}X = 2$, $\mathbb{E}Y = 4$, and the following constraint holds true $X^2 + Y = 8$. Find:
a) $\mathbb{E}[X+Y]$
b) $\operatorname{Var}X$

4. Find mean and variance of a continuous uniform $[a,b]$-distribution:

$$p_{unif}(x) = \begin{cases} \dfrac{1}{b-a}, & x \in [a,b], \\ 0 & \text{otherwise.} \end{cases}$$

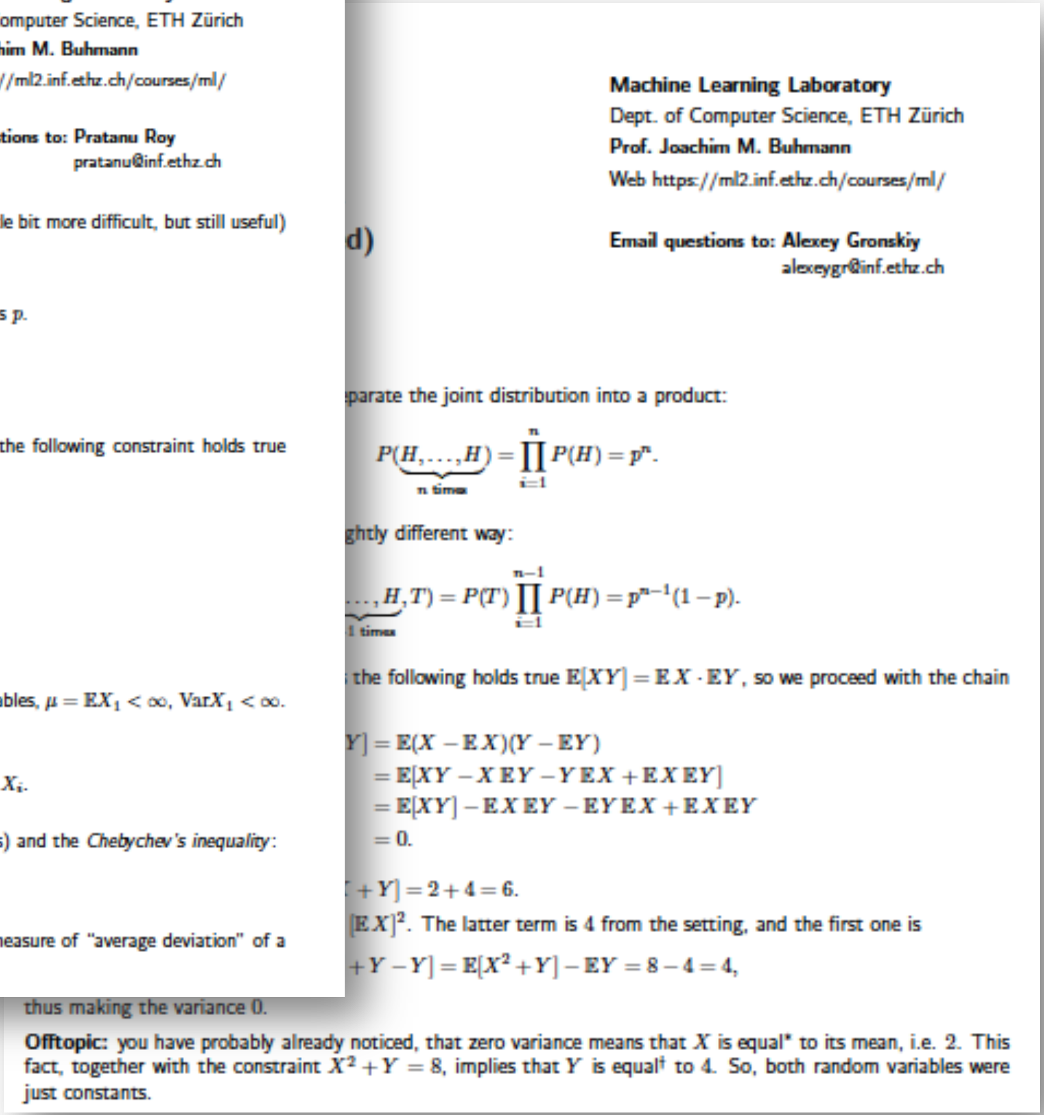
5*. (Weak Law of Large Numbers) Let $X_1, X_2, \dots, X_n$ be i.i.d. random variables, $\mu = \mathbb{E}X_1 < \infty$, $\operatorname{Var}X_1 < \infty$. Prove that the *empirical mean* converges in probability to the true mean:

$$\bar{X}(n) \xrightarrow{P} \mu \ (n \to \infty) \quad \text{where } \bar{X}(n) := \frac{1}{n}\sum X_i.$$

Hint: use the definition of convergence in probability (from the tutorial slides) and the *Chebychev's inequality*:

$$\mathbb{P}(|Z - \mathbb{E}Z| \geq \varepsilon) \leq \frac{\operatorname{Var}Z}{\varepsilon^2}.$$

Offtopic Note: Chebychev's inequality justifies the usage of variance as a measure of "average deviation" of a random variable from its mean.



Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Joachim M. Buhmann
Web https://ml2.inf.ethz.ch/courses/ml/

d)

Email questions to: Alexey Gronskiy
alexeygr@inf.ethz.ch

parate the joint distribution into a product:

$$P(\underbrace{H,\dots,H}_{n \text{ times}}) = \prod_{i=1}^{n} P(H) = p^n.$$

ghtly different way:

$$\underbrace{\dots, H}_{1 \text{ times}}, T) = P(T) \prod_{i=1}^{n-1} P(H) = p^{n-1}(1-p).$$

the following holds true $\mathbb{E}[XY] = \mathbb{E}X \cdot \mathbb{E}Y$, so we proceed with the chain

$$\begin{aligned} Y] &= \mathbb{E}(X - \mathbb{E}X)(Y - \mathbb{E}Y) \\ &= \mathbb{E}[XY - X\,\mathbb{E}Y - Y\,\mathbb{E}X + \mathbb{E}X\,\mathbb{E}Y] \\ &= \mathbb{E}[XY] - \mathbb{E}X\,\mathbb{E}Y - \mathbb{E}Y\,\mathbb{E}X + \mathbb{E}X\,\mathbb{E}Y \\ &= 0. \end{aligned}$$

$+ Y] = 2 + 4 = 6.$

$\mathbb{E}X]^2$. The latter term is 4 from the setting, and the first one is

$+ Y - Y] = \mathbb{E}[X^2 + Y] - \mathbb{E}Y = 8 - 4 = 4,$

thus making the variance 0.

Offtopic: you have probably already noticed, that zero variance means that X is equal* to its mean, i.e. 2. This fact, together with the constraint $X^2 + Y = 8$, implies that Y is equal† to 4. So, both random variables were just constants.

Supervised learning

Model: Loss-function + Regularization

Squared loss, 0/1 loss, Perceptron loss, Hinge loss, Regret, Bayesian expected loss, …

L^2 norm, L^1 norm, Smoothness…

Method: Exact solution, Gradient Descent, SGD, Sampling, Dynamic programming,…

Model selection: Cross-Validation, Bayes factor, Minimum description length …

Regression

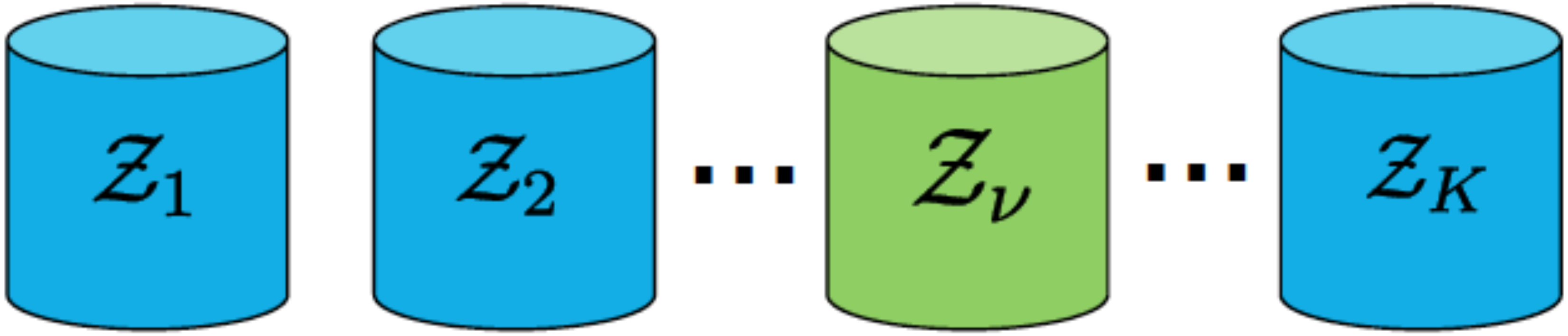
	Model	Method
Linear regression	$\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$	Exact solution
Ridge regression	$\min_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \|\|\mathbf{w}\|\|_2^2$	Gradient descent
Lasso	$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 + \lambda \|\|\boldsymbol{w}\|\|_1$	Quadratic programming

Exact inference is computationally demanding, however approximate inference leads to the optimal solution. Why?

Do we need model selection for linear regression? And for ridge regression? What for?

K-fold cross-validation

Split data in K approximately equally sized subsets, i.e., $\mathcal{Z} = \mathcal{Z}_1 \bigcup \mathcal{Z}_2 \bigcup \cdots \bigcup \mathcal{Z}_\nu \bigcup \cdots \bigcup \mathcal{Z}_K$;

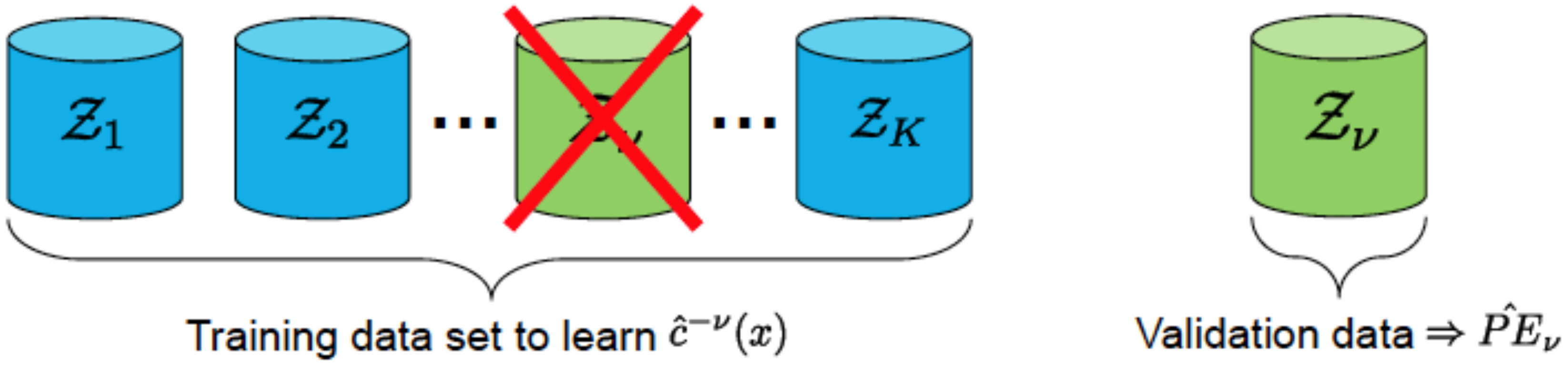


For every partition of a data set in K subsets, we can define K training data sets with approximately $n\frac{K-1}{K}$ data samples.

K-fold cross-validation

2) ν-th step

Adapt a model to the $K-1$ data subsets (learning step); validate the resulting model with the not yet used subset $\mathcal{Z}_\nu$



$\mathcal{Z}_1$
$\mathcal{Z}_2$
$\mathcal{Z}_\nu$
$\mathcal{Z}_K$
$\mathcal{Z}_\nu$
Training data set to learn $\hat{c}^{-\nu}(x)$
Validation data $\Rightarrow \hat{PE}_\nu$

3) Estimation of the prediction error

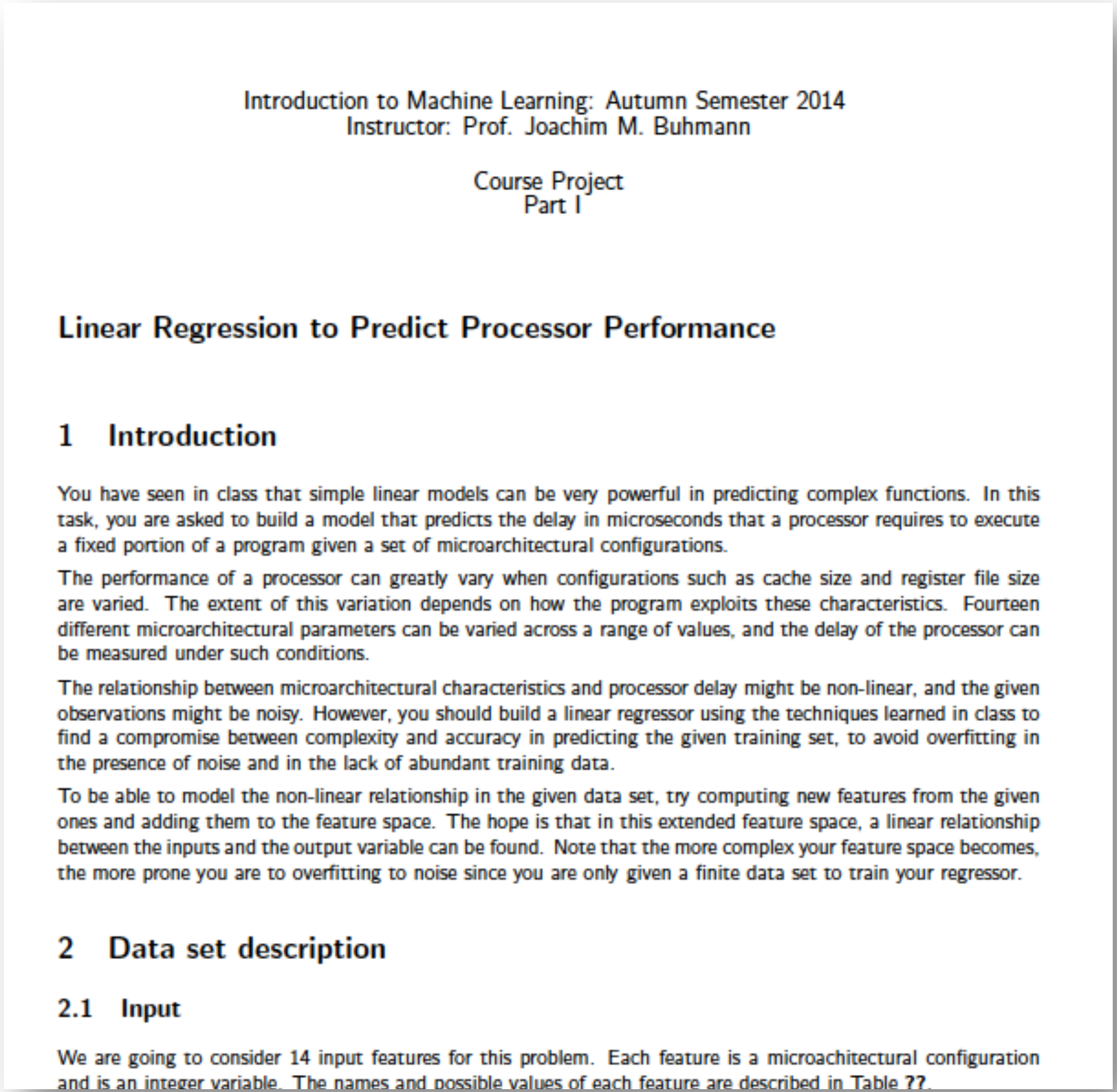
Prediction error

- Prediction error is not the same as loss function!
- Prediction error is defined by the nature of the problem you tackle not by the model or method you use.
- Regression:
  - Mean squared sum of residuals (MSE – mean squared error)
  - Root mean squared sum of residuals (RMSE)
  - …
- Classification:
  - Accuracy
  - Precision and recall
  - Mutual information
  - …

Cross-validation for model selection

- Run cross-validation for every value of the hyperparameter (λ, C,…).
- Choose the value corresponding to the lowest prediction error.
- However (!) you can not report this error as the generalization error of your algorithm. Why?
- You have used all your data already for choosing your hyperparameter.
- So you need a separate dataset to evaluate the generalization error.

Project



Introduction to Machine Learning: Autumn Semester 2014
Instructor: Prof. Joachim M. Buhmann

Course Project
Part I

Linear Regression to Predict Processor Performance

1 Introduction

You have seen in class that simple linear models can be very powerful in predicting complex functions. In this task, you are asked to build a model that predicts the delay in microseconds that a processor requires to execute a fixed portion of a program given a set of microarchitectural configurations.

The performance of a processor can greatly vary when configurations such as cache size and register file size are varied. The extent of this variation depends on how the program exploits these characteristics. Fourteen different microarchitectural parameters can be varied across a range of values, and the delay of the processor can be measured under such conditions.

The relationship between microarchitectural characteristics and processor delay might be non-linear, and the given observations might be noisy. However, you should build a linear regressor using the techniques learned in class to find a compromise between complexity and accuracy in predicting the given training set, to avoid overfitting in the presence of noise and in the lack of abundant training data.

To be able to model the non-linear relationship in the given data set, try computing new features from the given ones and adding them to the feature space. The hope is that in this extended feature space, a linear relationship between the inputs and the output variable can be found. Note that the more complex your feature space becomes, the more prone you are to overfitting to noise since you are only given a finite data set to train your regressor.

2 Data set description

2.1 Input

We are going to consider 14 input features for this problem. Each feature is a microachitectural configuration and is an integer variable. The names and possible values of each feature are described in Table ??.

Project

- Key steps:
  - Construct features
  - Run cross-validation to choose the best model on the training data
  - Apply to a second dataset to estimate the generalization error via submission system. Be careful! Do not overdo it as then you risk to overfit to the second dataset!

- Models can vary in different aspects:
  - Feature transformations
  - Feature sets
  - Output transformations
  - Values of the hyperparameters
  - And many more…

How to play with the data

- Renormalizing data

$$\tilde{x}_{i,j} = (x_{i,j} - \hat{\mu}_j)/\hat{\sigma}_j$$

- Nonlinear transformations of the features (log, power, sqrt)

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

- Output transformation

$$\tilde{y} = \phi_y(y)$$

- Feature selection (bias-variance trade-off!!)