

Lecture 10 – Summary and reflections



UPPSALA
UNIVERSITET

Niklas Wahlström

Division of Systems and Control
Department of Information Technology
Uppsala University.

Email: niklas.wahlstrom@it.uu.se

Contents – Lecture 10

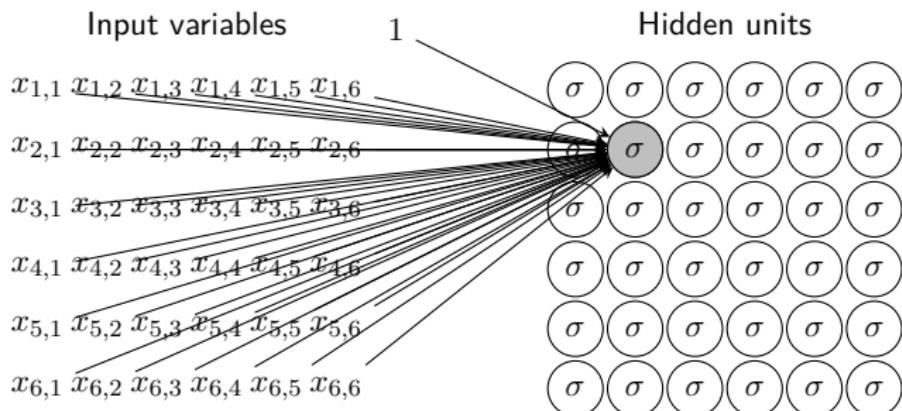
1. Summary of Lecture 9
2. Summary of the laboratory work
3. Summary of the whole course
4. Outlook: *a few words about things that we have not covered*
5. New course!

Summary of Lecture 9 (I/IV)

Convolutional layer

Consider a hidden layer with 6×6 hidden units.

- **Dense layer:** Each hidden unit is connected with **all pixels**.
Each pixel-hidden-unit-pair has its own **unique parameter**.

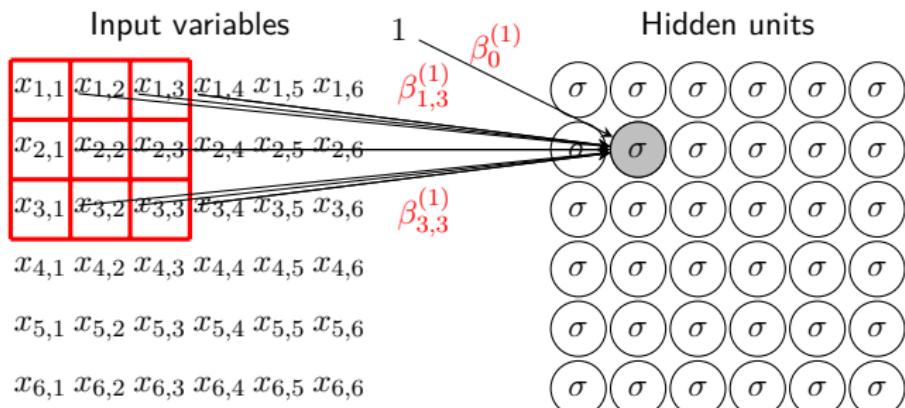


Summary of Lecture 9 (I/IV)

Convolutional layer

Consider a hidden layer with 6×6 hidden units.

- **Dense layer:** Each hidden unit is connected with **all pixels**.
Each pixel-hidden-unit-pair has its own **unique parameter**.
 - **Convolutional layer:** Each hidden unit is connected with a **region of pixels** via a set of parameters, so-called **kernel**.
Different hidden units have the **same set of parameters**.
-

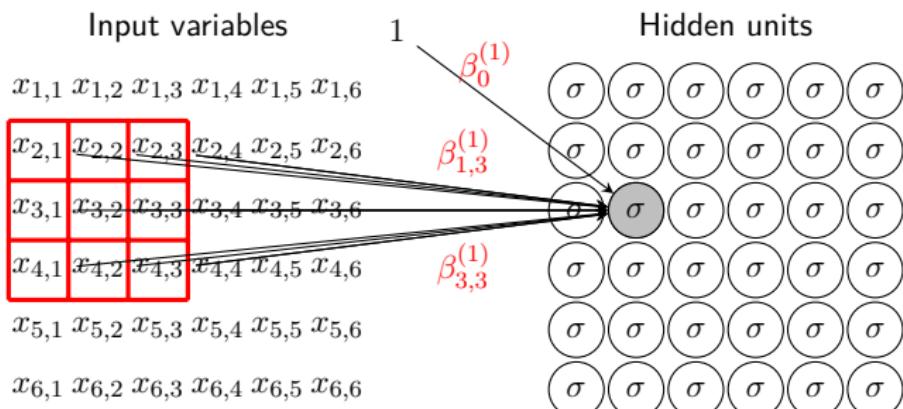


Summary of Lecture 9 (I/IV)

Convolutional layer

Consider a hidden layer with 6×6 hidden units.

- **Dense layer:** Each hidden unit is connected with **all pixels**.
Each pixel-hidden-unit-pair has its own **unique parameter**.
- **Convolutional layer:** Each hidden unit is connected with a **region of pixels** via a set of parameters, so-called **kernel**.
Different hidden units have the **same set of parameters**.

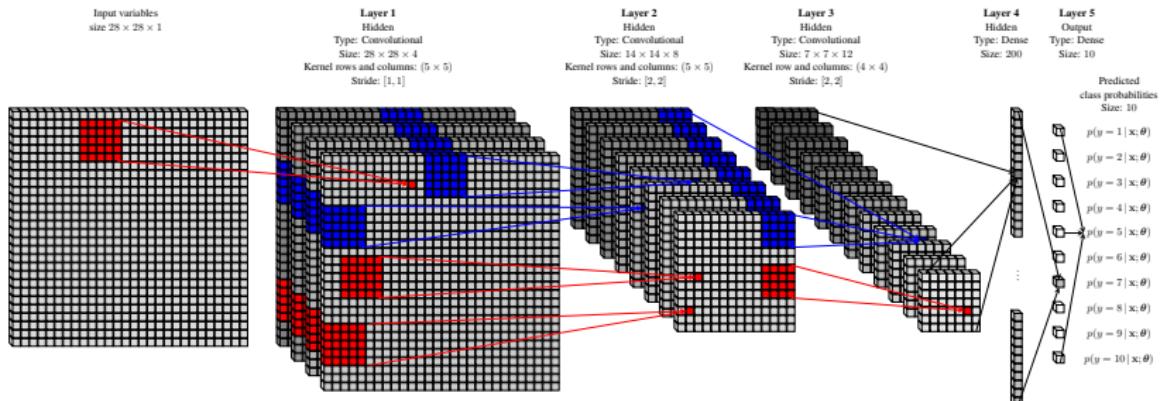


Summary of Lecture 9 (II/IV)

Convolutional neural network (CNN)

A full CNN usually consists of

- multiple convolutional layers (here three),...
- ...and a few final dense layers (here two).



Summary of Lecture 9 (III/IV)

Training a neural network

We train a network by considering the optimization problem

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \quad J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n L(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta})$$

- $\boldsymbol{\theta}$ – all parameters of the network
- $\hat{\boldsymbol{\theta}}$ – the estimated parameters
- $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$ – the training data
- $L(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta})$ – the loss function (for example cross-entropy)
- $J(\boldsymbol{\theta})$ – the cost function

Summary of Lecture 9 (IV/IV)

Stochastic gradient descent

At each optimization step we need to compute the gradient

$$\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) = \frac{1}{n} \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} L(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}_t).$$

Challenge - n is big - expensive to compute gradient.

Solution: For each iteration, we only use a small random batch of the data set – a **mini-batch** – to compute the gradient g_t . This procedure is called the **stochastic gradient descent**.

Training data (reshuffled)

\mathbf{x}_{19}	\mathbf{x}_{16}	\mathbf{x}_{18}	\mathbf{x}_6	\mathbf{x}_9	\mathbf{x}_{13}	\mathbf{x}_1	\mathbf{x}_{14}	\mathbf{x}_{20}	\mathbf{x}_{11}	\mathbf{x}_3	\mathbf{x}_8	\mathbf{x}_7	\mathbf{x}_{12}	\mathbf{x}_4	\mathbf{x}_{17}	\mathbf{x}_5	\mathbf{x}_{10}	\mathbf{x}_2	\mathbf{x}_{15}
\mathbf{y}_{19}	\mathbf{y}_{16}	\mathbf{y}_{18}	\mathbf{y}_6	\mathbf{y}_9	\mathbf{y}_{13}	\mathbf{y}_1	\mathbf{y}_{14}	\mathbf{y}_{20}	\mathbf{y}_{11}	\mathbf{y}_3	\mathbf{y}_8	\mathbf{y}_7	\mathbf{y}_{12}	\mathbf{y}_4	\mathbf{y}_{17}	\mathbf{y}_5	\mathbf{y}_{10}	\mathbf{y}_2	\mathbf{y}_{15}

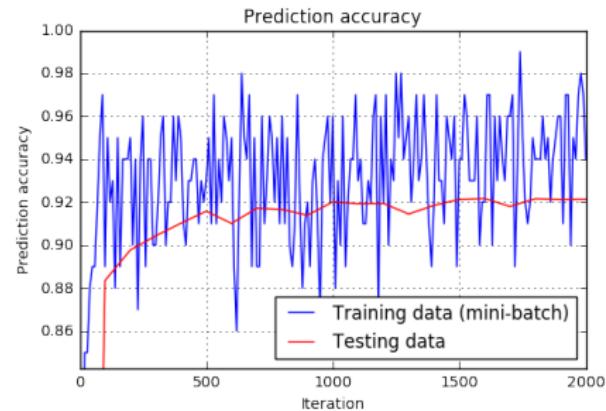
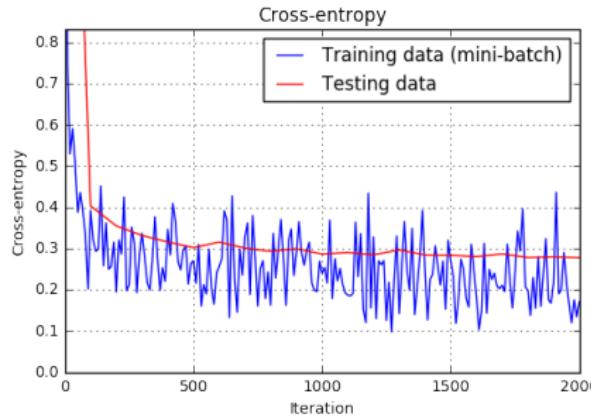
Iteration: 6

Epoch: 2

Summary of laboratory work

One layer neural network (logistic regression)

Trained for 10'000 iterations. SGD with learning rate: $\gamma = 0.5$

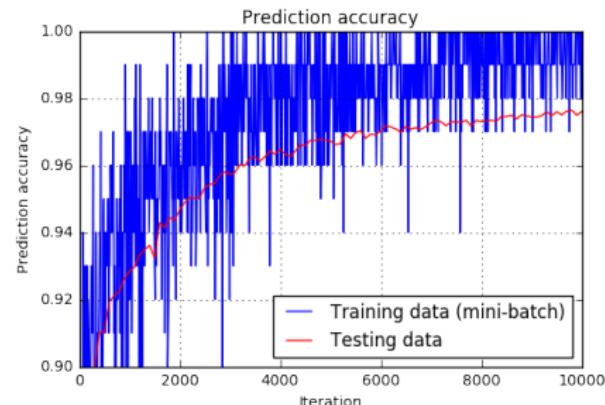
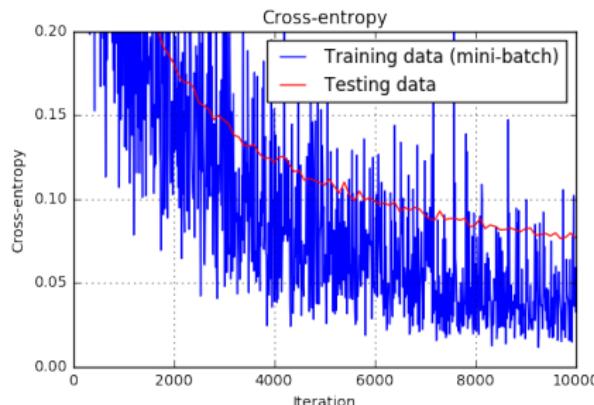


Summary of laboratory work

Two layer neural network with sigmoid activation function.

Trained for 10'000 iterations.

SGD with learning rate: $\gamma = 0.5$



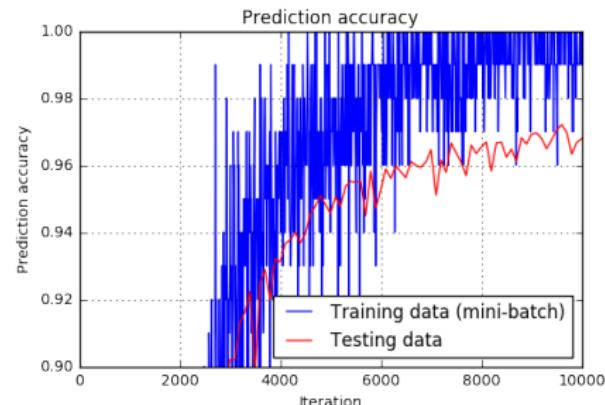
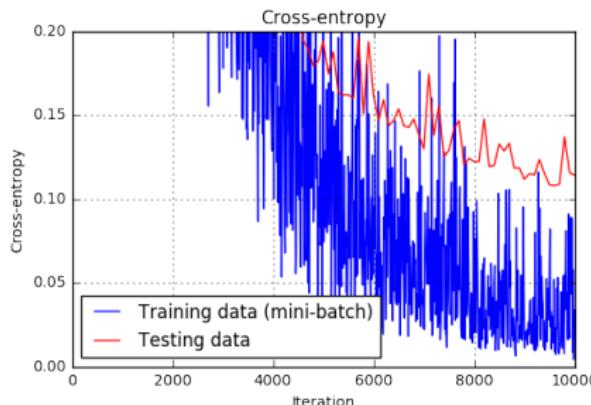
Significantly better performance.

Summary of laboratory work

Five layer neural network with sigmoid activation function.

Trained for 10'000 iterations.

SGD with learning rate: $\gamma = 0.5$



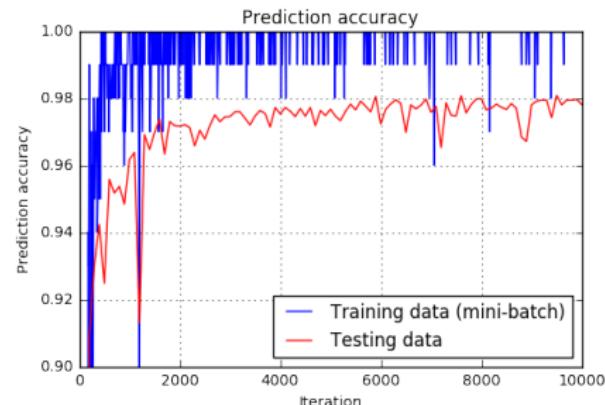
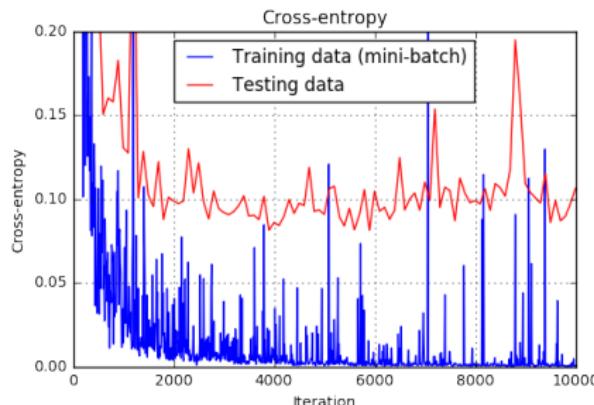
Convergence slow, not yet converged.

Summary of laboratory work

Five layer neural network with ReLU activation function.

Trained for 10'000 iterations.

SGD with learning rate: $\gamma = 0.5$



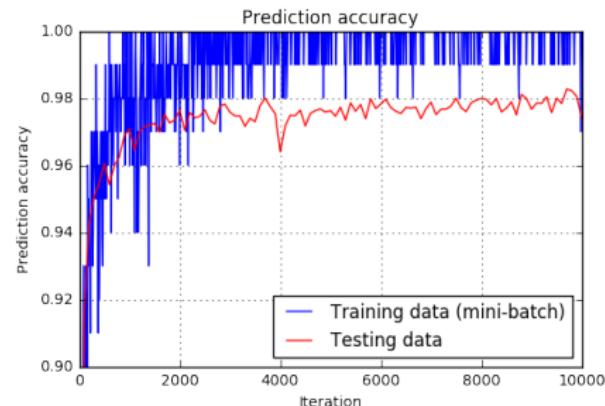
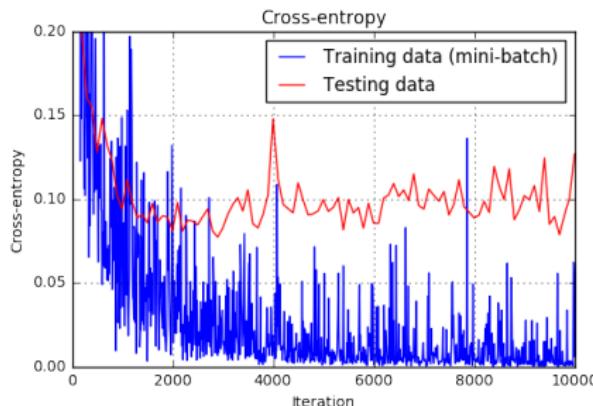
It trains much faster!

Summary of laboratory work

Five layer neural network with ReLU activation function.

Trained for 10'000 iterations.

Adam with learning rate: $\gamma = 0.002$



Not a big difference with Adam optimizer (but it is important in the CNN part!)

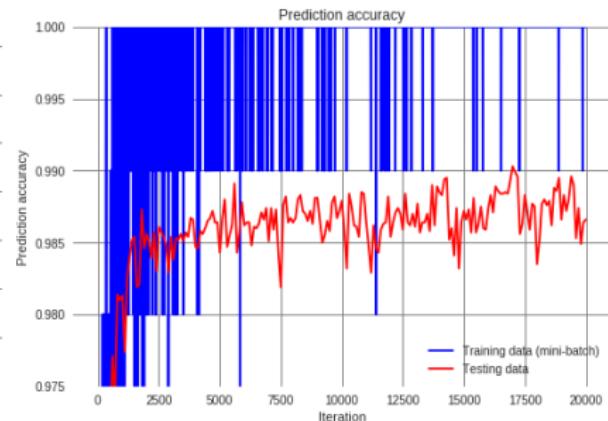
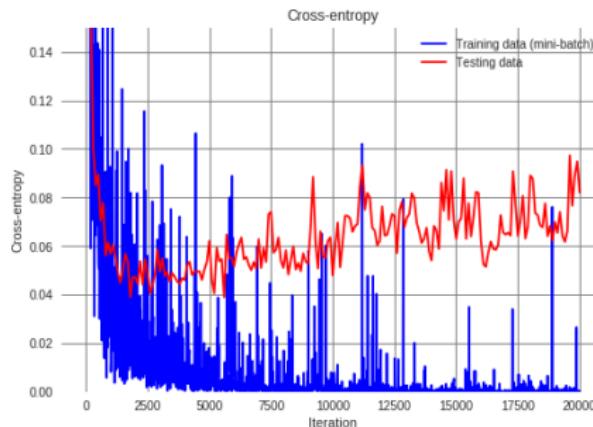
Summary of laboratory work

CNN - three conv layers, two dense layers

Channels/units: 4-8-12-200-10,

Kernels 5x5str1 - 5x5str2 - 4x4str2

Adam with learning rate: $\gamma = 0.002$



CNN increases performance!

Cost function oscillates \Rightarrow decrease learning rate

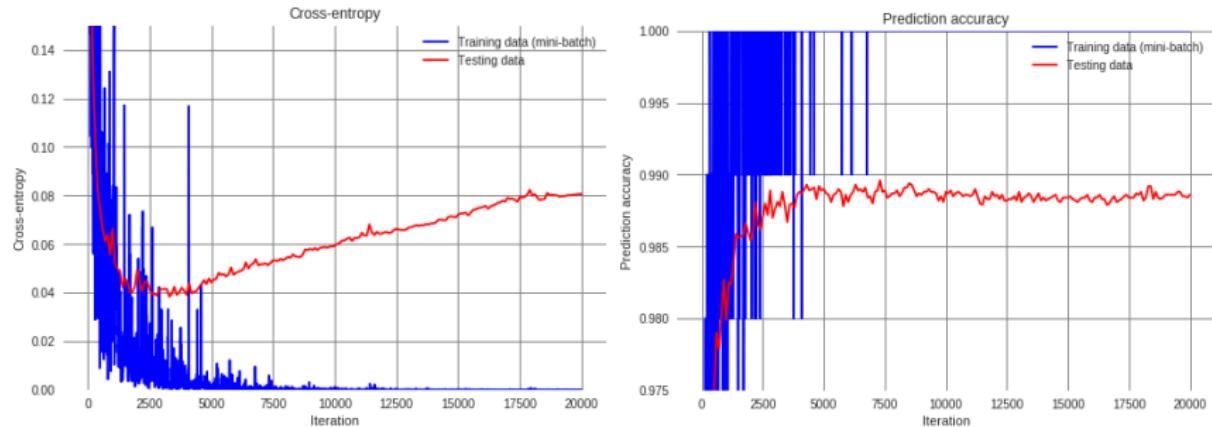
Summary of laboratory work – Extras!

CNN - three conv layers, two dense layers

Channels/units: 4-8-12-200-10,

Kernels 5x5str1 - 5x5str2 - 4x4str2

Adam with decaying learning rate from: $\gamma = 0.003$ to $\gamma = 0.0001$



And now we start to overfit ... Regularize!

Summary of laboratory work – Extras!

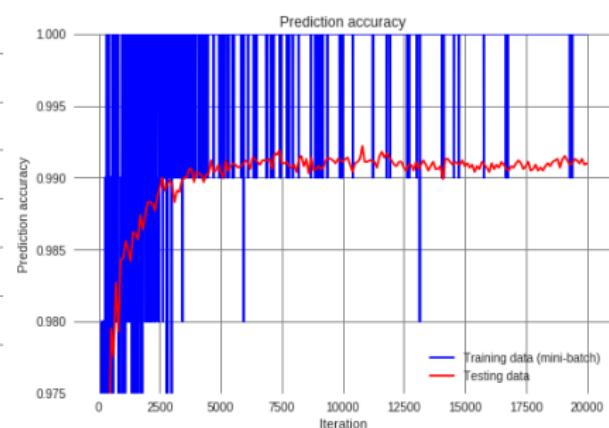
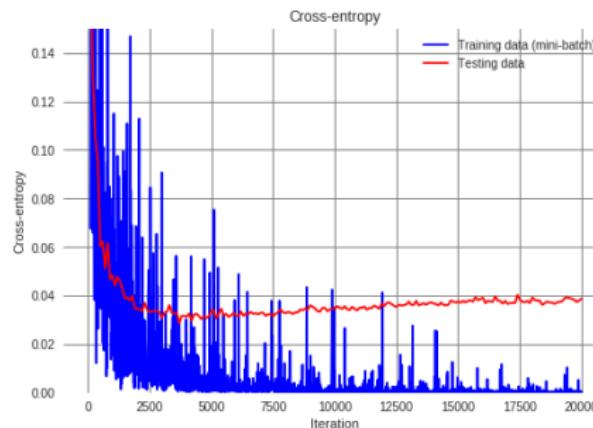
CNN - three conv layers, two dense layers

Channels/units: 4-8-12-200-10,

Kernels 5x5str1 - 5x5str2 - 4x4str2

Adam with decaying learning rate from: $\gamma = 0.003$ to $\gamma = 0.0001$

Dropout with $p = 0.75$ on units in last hidden layer.



Better cross-entropy, and now also an improvement in accuracy!!

Summary of laboratory work – Extras!

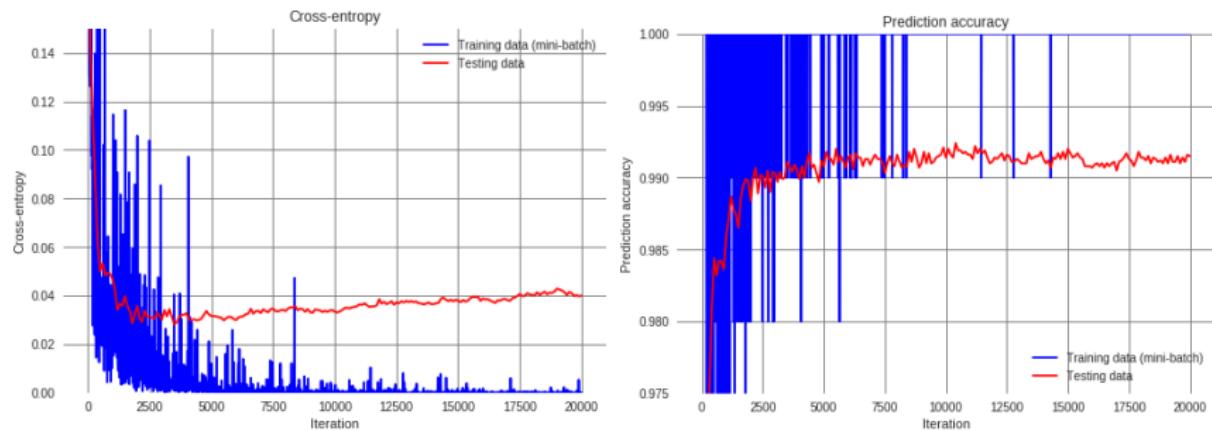
CNN - three conv layers, two dense layers

Channels/units: 6-12-24-200-10,

Kernels 6x6str1 - 5x5str2 - 4x4str2

Adam with decaying learning rate from: $\gamma = 0.003$ to $\gamma = 0.0001$

Dropout with $p = 0.75$ on units in last hidden layer.



This was the best I could get. Did you get any better?



This course

Machine learning gives computers the ability to
solve problems without being explicitly programmed
for the task at hand.

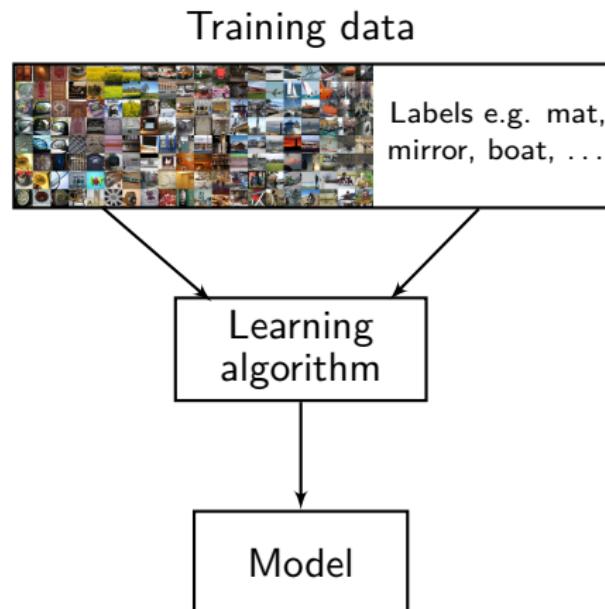
This is done by **learning from examples**, i.e. from training data.

Data on its own is typically useless, it is only when we can extract knowledge from the data that it becomes useful.

Specifically, we have studied **supervised learning** methods, in which we build a **model** of the relationship between an input variable x and an output variable y .

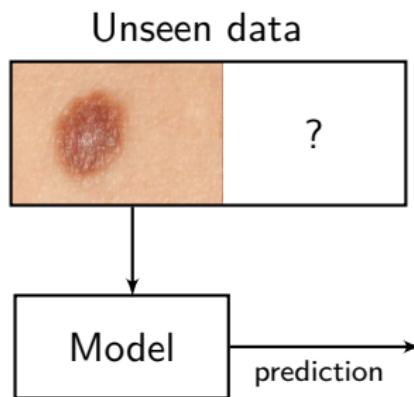
Supervised Machine Learning

Learning a model from labeled data.



Supervised Machine Learning

Using the learned model on new previously unseen data.



The model must **generalize** to new unseen data.

Inputs and outputs

The **input** \mathbf{x} is composed of all the **available** variables which are believed to be relevant for predicting the value of the **output** y .

We have considered the case where we have p input variables, $\mathbf{x} = (x_j)_{j=1}^p$, and one output variable y .

Both the inputs x_j and the output y can be either

- **quantitative** (can be ordered), or
- **qualitative** (takes values in an unordered set).

Regression and classification

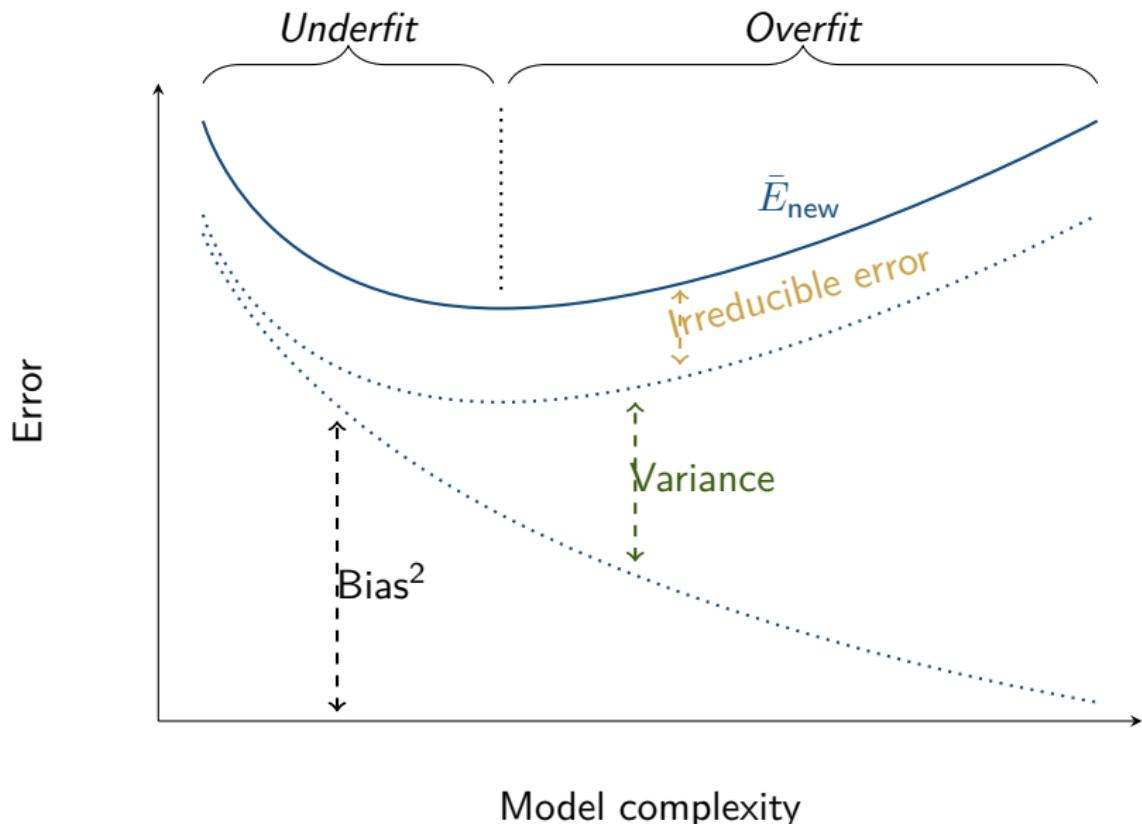
	Regression	Classification
Output, y	quantitative	qualitative
Inputs, x_j	quantitative or qualitative	quantitative or qualitative
Model ("conceptual")	$y = f(\mathbf{x}) + \varepsilon$	$p(y = k \mid \mathbf{x}),$ $k = 1, \dots, K$

Bias-variance

- E_{new} : How well a method will perform for unseen data.
- Bias: The inability of a model to describe the training data.
- Variance: How sensitive a model is to the training data.

$$E_{\text{new}} = \text{bias}^2 + \text{variance} + \text{irreducible error}$$

Bias-variance



Cross validation

To estimate E_{new} , we can use cross-validation.



When using cross validation to select, e.g., inputs and hyperparameters, there is a risk of overfitting! (But it can still be the best available option . . .)



Regularization

Regularization offers a way to decrease the model complexity (and hence risk of overfitting)

- **Ridge Regression:** add a penalty term $\lambda \|\beta\|_2^2$
- **LASSO:** add a penalty term $\lambda \|\beta\|_1$ – can result in *sparse* solutions

Select λ , e.g. by cross validation!

There are also other ways to change the model complexity:

- Increase k in k -NN
- Bagging

Parametric vs. nonparametric models

Parametric models

- Parameterized by a **finite-dimensional parameter** θ
- Training/learning the model = estimating $\hat{\theta}$
- Once $\hat{\theta}$ is estimated, the predictions depend only on $\hat{\theta}$ (not the training data)
- **ex)** Linear regression, LDA, QDA, Neural Networks

Nonparametric models

- The model **flexibility** is allowed to **grow** with the amount of available data
- **Predictions depend directly on the training data**
- Can be viewed as having an infinite number of parameters
- **ex)** k -NN, CART



Ensemble methods

Ensemble methods are a type of “meta algorithms”:

Construct **one powerful model** from multiple **base models** (=ensemble members), each of which may perform poorly on its own!

We have encountered two such approaches:

1. **Bagging:** Reduce variance of low-bias/high-variance models by **bootstrap aggregation**
2. **Boosting:** Construct **weak base models** sequentially, so that each model tries to **correct the mistakes** of the previous one

A toolbox of methods

	Regression	Classification	Non-parametric	Parametric	Ensemble
Linear regression	✓			✓	
Logistic regression		✓		✓	
LDA		✓		✓	
QDA		✓		✓	
k -NN	✓	✓	✓		
CART	✓	✓	✓		
Random Forests	✓	✓	✓		
AdaBoost		✓			
(Deep) Neural nets	✓	✓	(✓)		✓

Summary for the exam (in one slide)

- Classification and regression problem formulations
- Parametric and non-parametric models
- Inputs and outputs / quantitative and qualitative variables
- Decision boundaries / linear vs. nonlinear classifiers
- Cross-validation (the purpose!) and model testing
- Bias-variance trade-off / model flexibility / over-fitting
- Regularization / ridge regression and LASSO
- The **different methods** discussed throughout the course

Summary for life

What should you remember from statistical machine learning?

- The problem formulations: **regression** and **classification**
- The **existence** of different types of methods
- The **bias-variance trade-off** and **cross validation**
- The **possibilities**: Machine learning can be used for an extremely wide range of applications and data types

The TSTF principle: **Try simple things first!**

Outlook: Unsupervised learning

Regression and classification are **supervised learning** problems –
The models are trained using both inputs x and outputs y .

Unsupervised learning methods tries to find patterns in *unlabeled* data, i.e. we train the models from just the x .

- Dimensionality reduction / manifold learning
- Cluster analysis
- Generative model learning
- Blind source separation

Outlook: Reinforcement learning

A **reinforcement learning** system is asked to take ***actions*** that influence its ***environment*** in order to maximize a ***reward***.

Contrary to supervised learning,

- the correct input/output pair is not revealed
- learning has to be carried out based on the reward feedback
- often a focus on online performance
(\Rightarrow exploration-exploitation trade-off)

New course!!

Advanced probabilistic machine learning

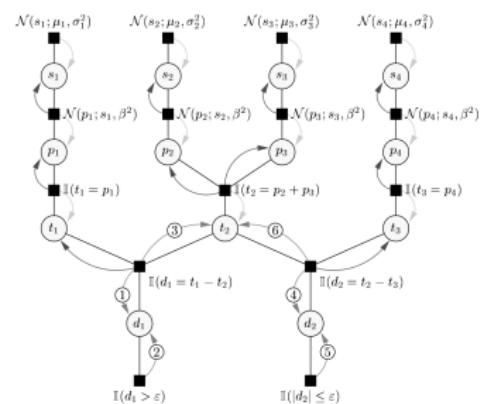
Contents (very brief):

- Probabilistic/Bayesian modeling
- Bayesian linear regression
- Graphical models
- Gaussian processes
- Variational inference
- Monte Carlo methods
- Unsupervised learning
- Variational autoencoders

Examination: Mini-project, lab, oral exam.

When: Period 1, running every year starting this fall.

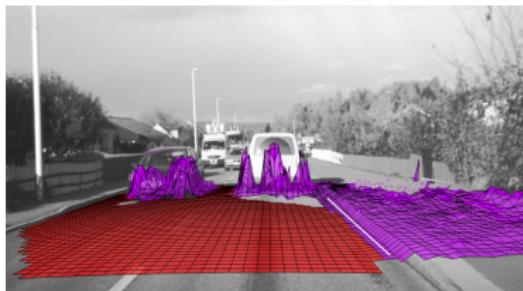
Info: <http://www.it.uu.se/edu/course/homepage/apml/>



Our Machine Learning research – ultrabrief

- Monte Carlo methods (especially sequential Monte Carlo)
- Deep learning
- Gaussian processes
- The use of probabilistic programming
- Applications: Autonomous driving (with Autoliv), digital pathology (with Sectra), etc.

We take a particular interest in nonlinear dynamical systems.



Thank you!

Machine learning gives computers the ability to
solve problems without being explicitly programmed
for the task at hand.

Thank you for your attention and good luck in the future!!!