MTTTS16 Learning from Multiple Sources

5 ECTS credits

Autumn 2019, University of Tampere
https://coursepages.uta.fi/mttts16/

Lecturer: Jaakko Peltonen

Lecture 0: Introduction, general overview of the topic

On this lecture:

- Course overview, how to pass the course
- List of course topics
- Introduction to the overall topic

Part 1: Course overview

Multi-view and multi-task learning are two aspects of jointly learning from a set of (partially) related learning problems / views / tasks.

This general concept underlies several subfields receiving increasing interest from the machine learning community, which differ in terms of the assumptions made about the dependency structure between learning problems.

We will cover a number of different learning tasks for integrating multiple sources and go through recent advances in the field. Examples of topics covered by the course include **data fusion**, **transfer learning**, **multitask learning**, **multiview learning**, and **learning under covariate shift**.

Practical Information

- Lectures on Tuesdays 14:15-16 each week in Pinni B0016, from September 3 to December 10.

- Some lectures may be rescheduled due to travel, more information later.

- No exercise sessions, instead home exercise packs, see next slide.

- Language: English

- 5 ECTS credits

- **You must sign up for the course** by filling in the online form. If you did not do this yet, do so now as soon as possible by contacting the lecturer.

Tentative Topics (note: some rescheduling may occur)

0	Intro
1	Basic CCA
2	Basic multitask learning with neural network arrangements
3	Transfer learning
4	Probabilistic CCA and kernel CCA
5	Multitask learning with task clustering or gating
6	Multitask learning with kernel methods and nonparametric models
7	Multi-view learning for classification by co-training
8	Co-training, continued
9	Multitask and transfer learning combined with semisupervised learning
10	Multi-view learning for clustering
11	Domain adaptation and covariate shift
12	Special setups 1: Learning sample correspondence
13	Possible 13th lecture: Asymmetric multitask learning

Practical Information

Material:

- course slides, additional-reading articles
- Slides originally in part by Arto Klami and Sohan Seth
- Exercise packs released later during the fall. Will contain some mathematical exercises, some implementation & testing of methods, either from scratch or using pre-existing toolboxes.

Practical Information, cont.

Grading:

- Each exercise graded 0-2 (integer), exercise packs total graded 0-5.
- Exam on final lecture, graded 0-5.

- To pass the course, you must pass the **exam** (grade 1 or more) and pass **exercise packs** (grade 1 or more).
- Passing grades are kept fractional between 1 and 5 (e.g. ”3.437”)
- Final course grade
  = round(0.8 * ExamGrade + 0.2 * ExercisesGrade)
  (e.g. 3.499 rounds to 3, 3.501 rounds to 4)

Contact information

Lecturers:

Prof. **Jaakko Peltonen**, jaakko.peltonen@tuni.fi
Office hours: by appointment only, room B0023 of the Pinni-B-building

When contacting by email, please include “Learning from Multiple Sources” in the subject line.

Part 2: Introduction to the overall topic

A typical machine learning scenario

You have a data set $\{\boldsymbol{x}_i\}_{i=1}^N$ **where** $\boldsymbol{x}_i \in R^d$

You may also have labels $\{y_i\}_{i=1}^N$ **, one for each data point**

Your task is to learn something about this data, for example estimate its density, learn underlying trends, reduce dimensionality, learn a classifier or regressor (predict *y* from *x*)

In easy scenarios N is quite large, d is quite small, and the complexity of the property you are trying to learn (density, trend, or predictor) is small

A typical machine learning scenario

You have a data set $\{\boldsymbol{x}_i\}_{i=1}^N$ **where** $\boldsymbol{x}_i \in R^d$

You may also have labels $\{y_i\}_{i=1}^N$ **, one for each data point**

Learning: typically either

- fit a single model to the data: optimize parameters, e.g. to maximize likelihood or minimize some cost function

$$\theta_{best} = arg\,max\; p(\{\boldsymbol{x}_i\}_{i=1}^N | \theta, M)$$

- Bayesian approach: learn a posterior distribution over models (by sampling, or fitting an approximation to the posterior)

$$p_{posterior}(\theta) \propto p(\{\boldsymbol{x}_i\}_{i=1}^N | \theta, M)\, p_{prior}(\theta | M)$$

A more difficult machine learning scenario

You have a data set $\{\boldsymbol{x}_i\}_{i=1}^N$ **where** $\boldsymbol{x}_i \in R^d$

You may also have labels $\{y_i\}_{i=1}^N$ **, one for each data point**

Your task is to learn something about this data, for example estimate its density, learn underlying trends, reduce dimensionality, learn a classifier or regressor (predict *y* from *x*)

Now N is quite small, d is quite large, and the property you want to learn can be complicated

Overlearning

You have a data set $\{\boldsymbol{x}_i\}_{i=1}^{N}$ **where** $\boldsymbol{x}_i \in R^d$

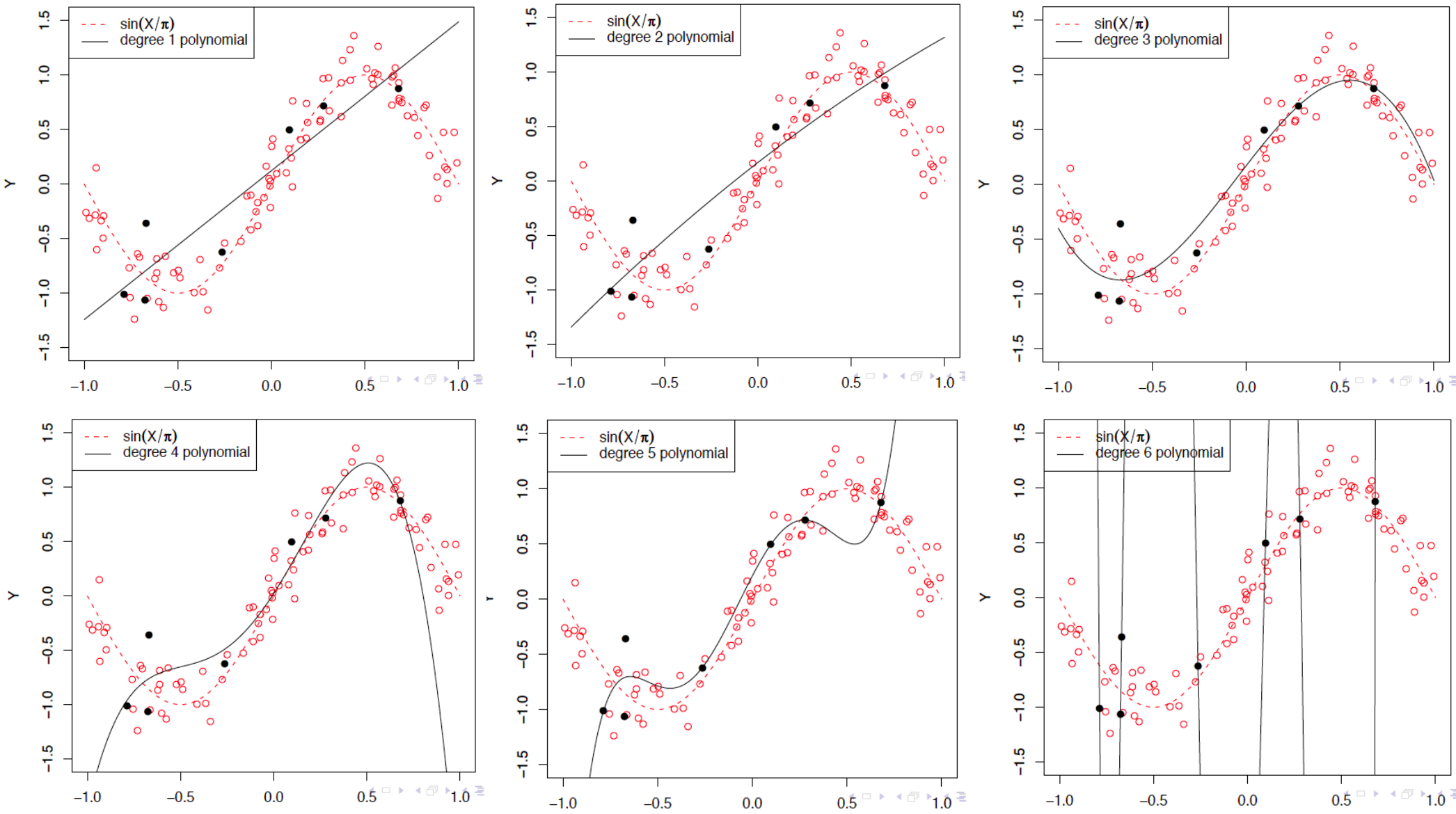
You may also have labels $\{y_i\}_{i=1}^{N}$ **, one for each data point**

When the amount of data is small, the number of features is large, and you are trying to learn a complicated property with a flexible model, your model may easily **mistake a peculiarity of the particular samples that you have for a real trend.**

For example, if you throw a coin twice and get **"tails, heads"**, that doesn't mean the coin gives "tails" on every odd-numbered throw... **but a model that learns from the data might think that!**

-----> Poor performance on new data

Polynomial regression example - fitting to a limited set of samples can lead to overlearning where the available samples appear to be fitted very well but the model predicts very poorly for new samples.



Overlearning

You have a data set $\{\boldsymbol{x}_i\}_{i=1}^N$ **where** $\boldsymbol{x}_i \in R^d$

You may also have labels $\{y_i\}_{i=1}^N$ **, one for each data point**

Typical solutions against overlearning:

- Add a regularization term (or prior)
- Restrict complexity of the model
- Preprocess the data to reduce dimensionality

These work but limit your ability to learn a complicated model

Overlearning

You have a data set $\{\boldsymbol{x}_i\}_{i=1}^N$ **where** $\boldsymbol{x}_i \in R^d$

You may also have labels $\{y_i\}_{i=1}^N$ **, one for each data point**

What about the Bayesian approach?

In principle, if you infer the correct posterior without approximation, and your assumed model family is correct, then no overlearning happens.

But if you use a wrong model family **(almost always true!)**, or if you do approximations, you can still get bad performance.

But that's pretty much the best we can do with a data set!
Is the situation hopeless?

Can we get more data?

You have a several data sets, $D_k = \{ \boldsymbol{x}_{ki}, y_{ki} \}_{i=1}^{N_k}$, $\boldsymbol{x}_{ki} \in R^{d_k}$

in general, they have different numbers of data, different dimensionalities

You assume they are all about the same phenomenon (or at least similar ones). But they are not identical: underlying distributions of data are not the same, underlying conditional distributions of labels are not the same.

Can we get more data?

You have a several data sets, $D_k = \{ \boldsymbol{x}_{ki}, y_{ki} \}_{i=1}^{N_k}$, $\boldsymbol{x}_{ki} \in R^{d_k}$

in general, they have different numbers of data, different dimensionalities

Simple answer: **single-task learning.** Forget the other data sets, just use one of them. Will not help against overlearning. On the other hand, ensures that you will not accidentally learn from undesirable effects present in the other data sets.

Can we get more data?

You have a several data sets, $D_k = \{ \boldsymbol{x}_{ki}, y_{ki} \}_{i=1}^{N_k}$, $\boldsymbol{x}_{ki} \in R^{d_k}$

in general, they have different numbers of data, different dimensionalities

Simple answer: **pool all data and learn from the pooled set.** Requires all data sets to have the same dimensionality. Assumes that the underlying data distributions are the same in all data sets. Helps against overlearning (lots of more data to learn from), but if the data sets are not from the same distribution, adds undesirable distortions.

Can we get more data?

You have a several data sets, $D_k = \{ \boldsymbol{x}_{ki}, y_{ki} \}_{i=1}^{N_k}$, $\boldsymbol{x}_{ki} \in R^{d_k}$

in general, they have different numbers of data, different dimensionalities

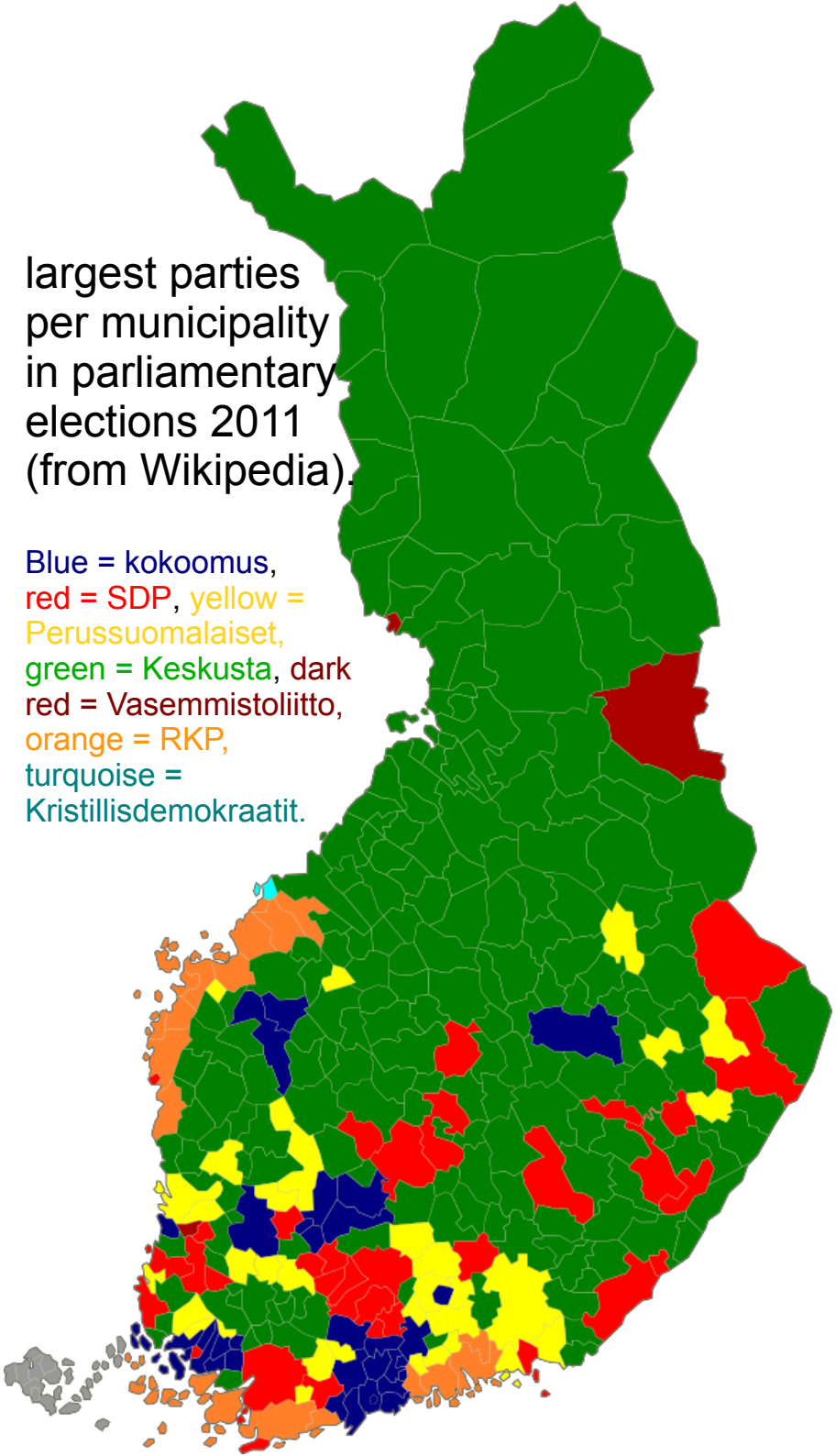
Complicated answer: **build a (hierarchical) model for all the data sets, where the models of different data sets share parameters or hyperparameters.** Then learn from all data sets using the model. Precise definition of the model depends on the learning problem, known connections between data sets, and assumed connections between the underlying distributions.

Can we get more data?

You have a several data sets, $D_k = \{ \boldsymbol{x}_{ki}, y_{ki} \}_{i=1}^{N_k}$, $\boldsymbol{x}_{ki} \in R^{d_k}$

in general, they have different numbers of data, different dimensionalities

Complicated answer, continued:
"Sharing statistical strength": if the models of all data sets share some parameters, then learning of those parameters is affected by all data from all data sets.
Thus the shared parameters are learned well
---> only the non-shared parameters suffer from overlearning
----> overall, models of all data sets are learned better than with single-task learning.



Example

Polls ask only a small number of people - how to get reliable measures?

Even results of parliamentary elections are only a sample based measure of party popularity - not everyone votes.

Popularity in each municipality: only a few samples (voters / polled people) available. Can we use voters/polled people from other municipalities (even the whole country) to build better estimates per municipality?

Different connections between data sets

1. Same data, different outputs: essentially you only have one data set, but you have several different output types for each data point. *Example: data point=gene expression measurement from a person, outputs=disease type, gender, age*

Different connections between data sets

1. Same data, different outputs: essentially you only have one data set, but you have several different output types for each data point. *Example: data point=gene expression measurement from a person, outputs=disease type, gender, age*

2. Paired data: for each data point in one set, there is a counterpart in each of the other sets. This can also be seen as single data set where the features are divided into several groups. *Example: video recording, video and audio.*

Different connections between data sets

3. Data sets with unknown pairings: each data point has a pair in the other sets, but you don't know which points are pairs.

Example: voting vs. exit polls. Each person votes, and is then interviewed when leaving. The set of votes is anonymous, but each interviewed person gave one of the votes.

Different connections between data sets

3. Data sets with unknown pairings: each data point has a pair in the other sets, but you don't know which points are pairs.

Example: voting vs. exit polls. Each person votes, and is then interviewed when leaving. The set of votes is anonymous, but each interviewed person gave one of the votes.

4. No connection between data samples: you just assume the underlying phenomena (generating distributions) are related.

Example: matriculation examination performances of students in different high schools. Each student is a data sample (features = scores in different subjects, biographical information), each student only goes to one high school.

Not just avoiding overfitting: many other reasons for using more than one data set

•Sometimes you know that future data (test data) will not be exactly like your current training data: if you have some samples of test data you can adapt your learning from the training data

•You want to adapt a previously learned model to a new setting which is different but similar to the one you learned the model for.

•The relationship between data sets, or the shared property (shared trend or effect) is itself the interesting thing. *Example: what is common between effects of different cancer types on cell activity.*

Not just avoiding overfitting: many other reasons for using more than one data set

•The non-shared properties in each data may sometimes be distortions, e.g. caused by the specific way they were measured. You don't want to fit your model to the distortion in any of the sets.

•Sometimes each type of data tells a “part of the picture” and they complement one another. *Example: one data set has side profiles of heads, another has front profiles.*

•Can learn to complete missing parts from one data type based on others, or can learn to search across data types. *E.g. finding images based on a sound sample.*

• *In principle:* more efficient storage.

Concepts

Multitask learning

Transfer learning

Multi-view learning

Covariate shift

Semisupervised learning