CO3091 - Computational Intelligence and Software Engineering

Lecture 26

Feature Selection and Revision of Optimisation Problems

Leandro L. Minku

Overview

- The need for feature selection
- How to select input features to use with machine learning approaches
- Revision:
    - Lorry problem
    - Requirements optimisation problem
    - Software project scheduling
    - Software energy optimisation
    - Evolutionary software testing
- Feature selection as an optimisation problem

Module Questionnaire

https://leicester.surveys.qmihub.co.uk/

Please complete the module questionnaire for CO3091.

Selecting Input Features for Machine Learning Approaches

- Previous lectures:
  - Practitioners may have some idea about what input features are likely to be related to the output being predicted.
  - Software defect prediction:
    - McCabe cyclomatic complexity
    - Halstead complexity measures
    - Lines of code
    - Lines of comments
    - etc.

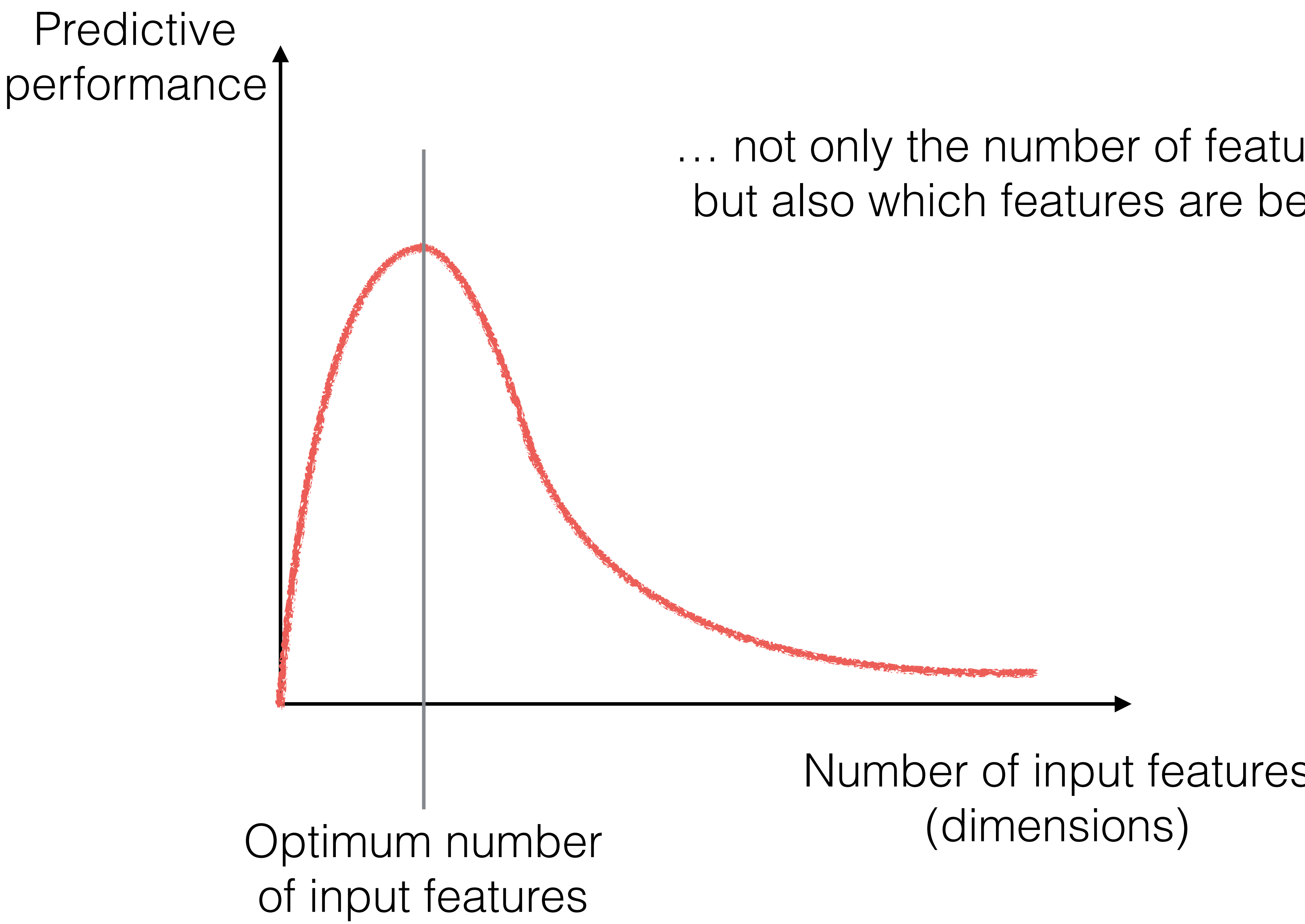
Selecting Input Features for Machine Learning Approaches

- If we miss including some important input feature, our predictive models may perform poorly.
- Result: to avoid missing some important feature, practitioners may suggest many input features, which are not all useful.
    - Is lines of comment really related to defects?
    - Is Halstead Volume needed if we already use LOC?

The Curse of Dimensionality

Image from: http://vignette1.wikia.nocookie.net/pirates/images/3/38/Fight_on_Isla_de_Muerta_16.png/revision/latest?cb=20110702154006

The Curse of Dimensionality



… not only the number of features matter, but also which features are being used.

Why Does This Happen?

- As we increase the number of input features (dimensions), we have less and less training data representing certain combinations of input feature values.
  - Certain areas of the input space will be uncovered.
- To cover those input feature values, we would need more training data, which is frequently unavailable.
  - The amount of training data needed often grows exponentially with the number of input features.

Therefore, having too many input features has a similar effect to having too few data, hindering the predictive performance of machine learning approaches.

Why Does This Happen? (cont.)

- Instances that are similar to each other on some input features may be very different on others.

1	1	0	0	0	0	0	0

1	1	0	0	1	1	1	1

- In the example, if the last 4 input features are irrelevant to the problem, they will deceive the algorithm into thinking that the two instances above are dissimilar when they are actually very similar.

- This can affect some machine learning approaches.

Select only the best features

"Make things as simple as possible,
but no simpler"

— Albert Einstein

Don't miss any important feature

How to Select Features Among a Set of Potential Features?

- Filter feature selection methods:
  - Light, can be applied before building a predictive model.
  - E.g.: correlation-based feature selection method (CFS).
- Wrapper feature selection methods.
  - Formulate feature selection as an optimisation problem.
  - We want to select the features that lead to better predictive performance.
  - Heavier, but possibly leads to better predictive performance than filter methods.

We can use optimisation algorithms!

Breakpoint!

Formulation of Optimisation Problems

- Design variables represent a solution.
- Design variables define the search space of candidate solutions.
- [Optional] Solutions must satisfy certain constraints.
- Objective function defines our goal.
  - Can be used to evaluate the quality of solutions.
  - Function to be optimised (maximised or minimised).

Lorry Problem

- Consider the following problem:
  - You need to load a lorry with products. The maximum total weight of products that the lorry can stand is W.
  - You have N products that can be loaded, and each product i has a weight w_i, and a profit p_i.

Problem: decide which products to load so as to maximise the total profit of loaded products.

Image from: http://www.wingstransport.com/cms-files/hpSlide-1357902701.png

Lorry Problem Formulation

- Design variable:
  - $v \in \{0,1\}^N$, where $v_i = 0$ represents that product i is not loaded, $v_i = 1$ represents that product i is loaded, and N is the number of products.
- Objective function:

$$f(v) = \sum_{i=1}^{N} v_i \, p_i \quad \text{(to be maximised).}$$

0 if product i is not loaded
1 otherwise

Total profit of loaded products

- Constraint:

$$\sum_{i=1}^{N} v_i \, w_i \leq W$$

Total weight of loaded products

Image from: https://s.yimg.com/pw/images/spaceout.gif

Requirements Selection

- As requirements have a cost, we may need to select a subset of all possible requirements to implement, so that the project will:
  - be within budget or
  - have lower cost.

- We need to decide which possible requirements to implement, considering (potentially among others):
  - their cost,
  - their value from different stakeholders perspectives,
  - the importance of the stakeholder who wants the requirement.

Requirements Selection Problem Formulation

Decision variable: $\mathbf{x} = \{0,1\}^n$

$x_j = 0$ if requirement r_j is not included

$x_j = 1$ if requirement r_j is included

n is the number of requirements

Objective: Maximise $\sum_{j=1}^{n} \text{score}(r_j)\, x_j$

Score reflects value of requirement from each stakeholder point of view and the importance of the stakeholder.

Constraint: $\sum_{j=1}^{n} \text{cost}(r_j)\, x_j \leq C$

total cost of selected requirements

Software Project Scheduling

Setting: assume we are given

- n employees $e_1, \ldots, e_n$ with salaries sal_i and sets of skills $skill_i$;
- m tasks $t_1, \ldots, t_m$ with required efforts $reqEff_j$ and sets of required skills $reqSk_j$;
- a task precedence graph (TPG).

Problem: allocate employees to tasks so as to:

- minimise cost (total salaries paid) and
- minimise duration (completion time).

Constraints:

- team must have required skills and
- no overwork.

Formulation — Design Variable

- Design variable:
  - **x'** = Matrix of employees by tasks.
  - Entries $x'_{i,j}$ represent the dedication of employee i to task j.
  - Dedication = percentage of the employee's time dedicated to the task.
  - Values that each entry $x'_{i,j}$ could assume depend on the granularity *k* of the problem.

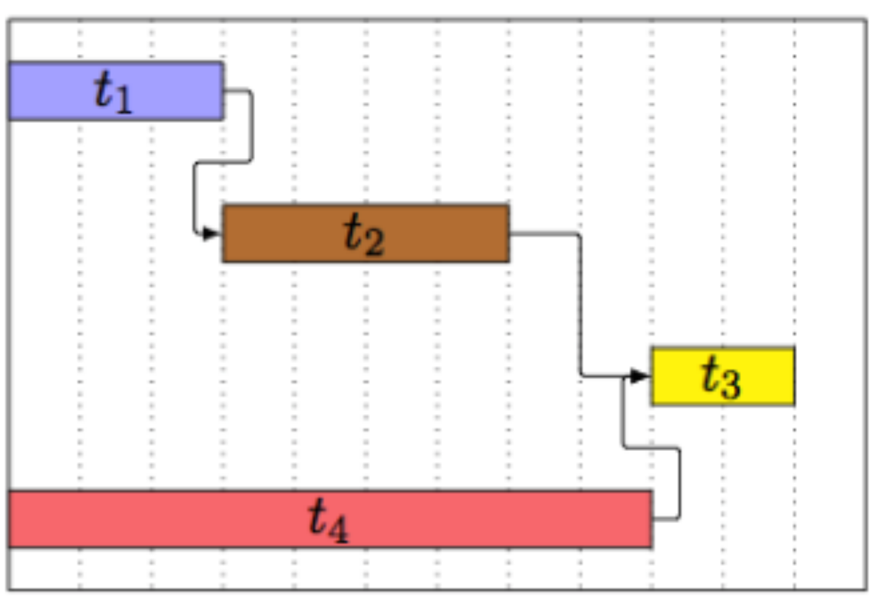
x'	**t_1**	**t_2**	**...**	**t_m**
e_1	$x'_{1,1}$	$x'_{1,2}$	...	$x'_{1,m}$
e_2	$x'_{2,1}$	$x'_{2,2}$	...	$x'_{2,m}$
...	...	...	...	...
e_n	$x'_{n,1}$	$x'_{n,2}$	...	$x'_{n,m}$

$$x'_{i,j} \in \left\{ \frac{0}{k}, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k}{k} \right\}$$

Calculating Cost and Duration

x'	t_1	t_2	t_3	t_4
e_1	$x'_{1,1}$	$x'_{1,2}$	$x'_{1,3}$	$x'_{1,4}$
e_2	$x'_{2,1}$	$x'_{2,2}$	$x'_{2,3}$	$x'_{2,4}$
e_3	$x'_{3,1}$	$x'_{3,2}$	$x'_{3,3}$	$x'_{3,4}$

+ TPG, tasks required efforts



+ salaries

Cost and duration

Software Energy Optimisation

- Mobile apps have widespread use.
- Mobile apps consume energy (battery).
- Studies show that battery consumption is one of the key factors considered by users when choosing a mobile app.
- In order to reduce energy consumption of OLED screens, one can optimise the colours used by the GUI.

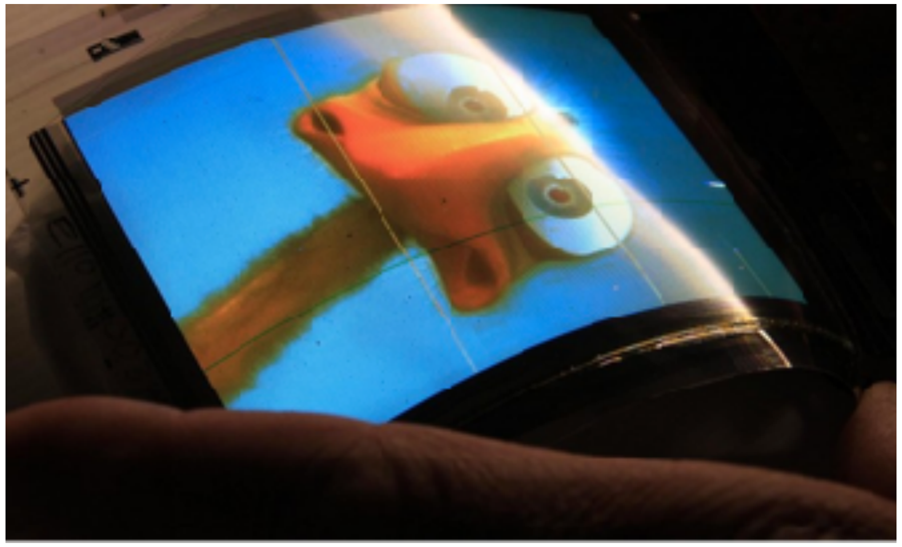


Image from: http://cdn.slashgear.com/wp-content/uploads/2013/04/flexible_oled-820x420.jpg

Software Energy Optimisation — Formulation

- Consider an initial GUI design with its colours.
- Design variables: RGB colour of each pixel.
- Objectives:
  - Minimise power consumption on OLED displays.
    - Power consumption of a GUI depends on the power consumption of its screens and the amount of time users spend on different screens.
    - Power consumption of a screen depends on the power consumption of its pixels.
    - Power consumption of a pixel is the sum of the power consumptions of its RGB colour components.
  - Maximise contrasts between adjacent GUI elements.
  - Minimise difference with respect to the original GUI design.
- Constraints:
  - Adjacent elements of the GUI should not have the same colour or colours with too low contrast between them.

Evolutionary Software Testing

- Software testing is an essential component for software development success.
- Software testing is one of the most expensive tasks in the software development process.
- Test suite: set of test cases, each consisting of a sequence of inputs and expected outputs from the program.
- Challenging for large and complex software.
- A good test suite should exercise the code well and be fast to run.

Evolutionary Software Testing — Formulation

- Design variable: list with a given number of input sequences.
  - Design variable can be seen as a test suite.
  - Each input sequence is a test case with an expected “output” of “no crash”.
  - Examples of inputs for Android: Touch, Motion, Rotation, Track-ball, PinchZoom, Flip, Nav, MajorNav, AppSwitch, SysOp, enter text, or clicks on widgets.
- Objectives:
  - Maximise coverage.
  - Minimise length of test cases.
  - Maximise number of crashes found.
- Constraints:
  - N/A

Back to Feature Selection

Image from: https://images-na.ssl-images-amazon.com/images/S/sgp-catalog-images/region_US/nbcu-61101081-Full-Image_GalleryBackground-en-US-1484000598187._RI_SX940_.jpg

Wrapper Methods

- Formulate feature selection as an optimisation problem.
- We want to select the features that lead to better predictive performance.

Feature Selection Problem Formulation

- Design variable: ?
- Objectives: ?
- Constraints: ?

Feature Selection Problem Formulation

- Design variable:
    - $v \in \{0,1\}^N$, where $v_i = 0$ represents that input feature i is not selected, $v_i = 1$ represents that input feature i is selected, and N is the number of available input features.

- Objective function:
  - Predictive performance of model created using the selected features.
    - Build a predictive model using a machine learning algorithm and the training set using the selected input features.
    - Evaluate the predictive performance of this predictive model.

Examples of Evaluation Functions Using a Known Data Set

- Classification error:
  - Given a data set D with examples $(\mathbf{x}_i, y_i)$, $1 \leq i \leq m$.
  - The actual output (target) for $\mathbf{x}_i$ is y_i.
  - The prediction given by a classification model to $\mathbf{x}_i$ is y_i'.
  - y_i and y_i' are categorical values.

$$\text{Classification error} = \frac{1}{m}\sum_{i=1}^{m} (y_i \neq y_i')$$

- Classification accuracy:

Classification accuracy = 1 - classification error

Examples of Evaluation Functions Using a Known Data Set

- Mean Squared Error (MSE):
  - Given a data set D with examples $(\mathbf{x}_i, y_i)$, $1 \leq i \leq m$.
  - The actual output (target) for $\mathbf{x}_i$ is y_i.
  - The prediction given by a regression model to $\mathbf{x}_i$ is y_i'.
  - y_i and y_i' are numerical values.

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m} (y_i - y_i')^2$$

- Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\text{MSE}}$$

Difference between optimisation and machine learning from problem perspective:

in machine learning we wish to create models with good generalisation ability.

Which Data To Use For Computing the Predictive Performance?

- We had three different types of data sets:
  - Training set:
    - Used by the machine learning approach to learn a model.
  - Validation set:
    - Used to choose between different machine learning approaches (or parameters for the approaches). It estimates the error on data unseen at the time of building the model.
  - Test set:
    - Separate data set used neither for training nor for validation. It can be used to give an idea of how well the model will perform / is performing in practice, i.e., how good the generalisation to future unseen data is likely to be.

Feature Selection Problem Formulation

- Design variable:
  - $v \in \{0,1\}^N$, where $v_i = 0$ represents that input feature i is not selected, $v_i = 1$ represents that input feature i is selected, and N is the number of available input features.
- Objective function:
  - Validation error of the model built based on the training set using only the selected features.
- Constraints:
  - None.

Now that the problem is formulated, it is possible to design an optimisation algorithm to solve it!

Overview

Optimisation problems:

- Feature selection
- Lorry problem
- Requirements optimisation problem
- Software project scheduling
- Software energy optimisation
- Software test case generation

Next lecture:

- Optimisation algorithms