Deep Learning for Natural Language Processing

A. Modupe[1] V. Marivate[2]

[1]School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg.

[2]Data Science Unit, MDS
Council for Scientific and Industrial Research,
Pretoria. South Africa

BootCamp, Lagos. Nigeria. 2018

About this tutorial

- DL has achieve state-of-the-art results on a suite of natural language processing problems

- Outline:
  - Section One:
    - Introduction: *motivation and why?*
    - DNNs as a friends: *basic machine learning concepts*
    - Word Level Semantics: we discuss the concept of word embedding to NLP
    - Overview of the Practicals
  - Section Two:
    - Language Modelling and RNNs Part 1
    - Language Modelling and RNNs Part 2
    - Text Classification

Why take NLP?

language is the most compelling manifestation of intelligence conducted between the mind and body communication

- **Prerequisites:**
  - Linear Algebra,
  - Calculus,
  - Probability.
- **Machine Learning:**
  - Evaluating ML models (train/validation/test split, cross validation etc.),
  - overfitting, generalisation, and regularisation,
  - optimisation (objective functions, stochastic gradient descent),
  - linear regression and classification, neural networks (common non-linearities, backpropagations etc.).
- **Programming:**
  - Knowledge of, or ability to learn quickly, a NN toolkit (e.g. Keras, TensorFlow, Theano, DyNet etc.)

What this course is, and is not, about

- In this lecture, we survey the use of Deep Learning techniques for a range of Natural Language Processing applications.
- Not a general course on NLP. There is a lot more to language and computational linguistics, and many interesting paradigms beyond deep learning, than we will cover.

Language Understanding

- CNN article:

  Document:

  > The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." . . .

  Query:

  > Who does the article say will not press charges against Jeremy Clarkson?

  Answer:

  > Oisin Tymon

Speech Processing and Machine Translation

Speech Recognition (ASR)

Les chiens aiment les os

Machine Translation (MT)

Dogs love bones

Text to Speech (TTS)

Image Understanding

Figure 1: Agrawal et al., VQA: Visual Question Answering, ICCV 2015.

What is the man handing?

Does it appear to be raining?

Does this man have 20/20 vision?

Linguistic Structure

Sense

I saw her duck

Idioms

He kicked a goal
He kicked the ball
He caught the ball
He kicked the bucket

Reference

The ball did not fit in the box because **it** was too [big/small].

etc.

A typical ML task: classification

- Learn decision boundaries between the different classes in a dataset

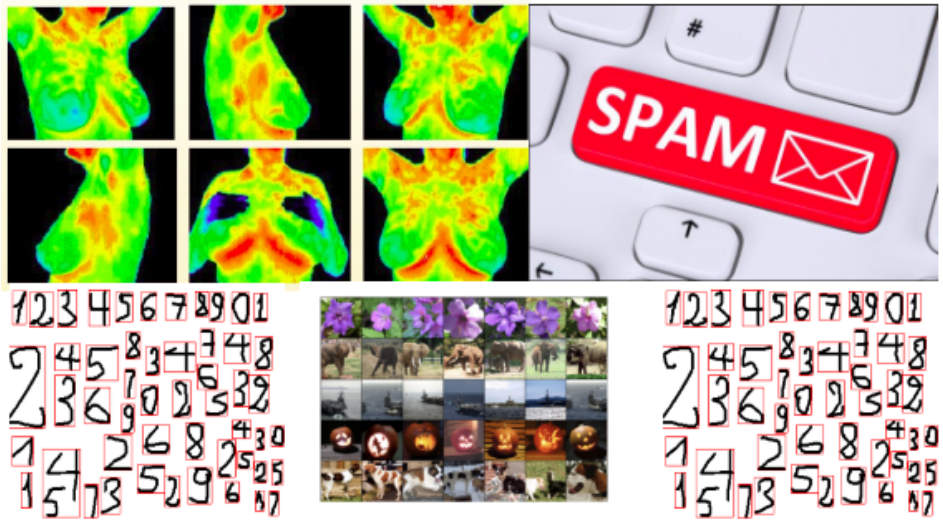


Figure 2: classification model

- Example applications:
  - handwritten character recognition
  - e-mail spam filter
  - cancer diagnosis

A typical ML task: clustering

- Group similar datapoints into meaningful clusters

Figure 3: Clustering model

- Example applications:
  - identify clients with similar insurance risk profiles
  - discover links between symptoms and diseases
  - build visual dictionaries

A typical ML task: reinforcement learning

- Learn a winning strategy through constant feedback.

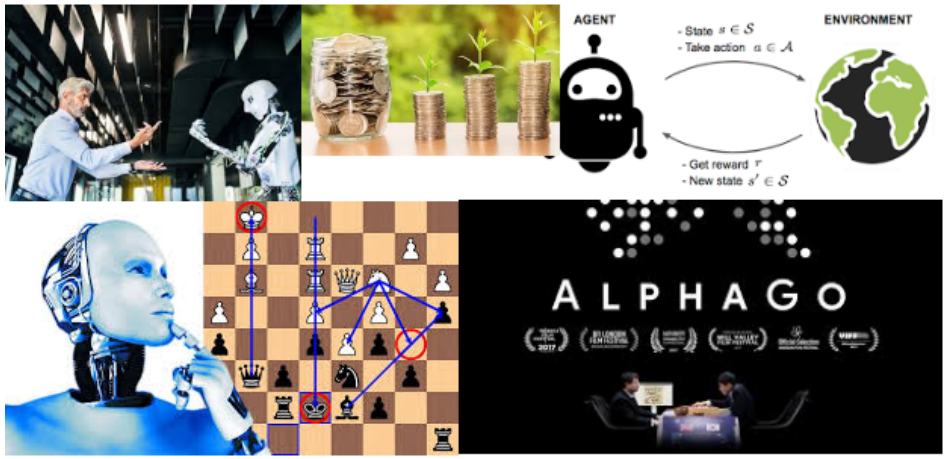


Figure 4: Application of Reinforcement learning

- Example applications:
  - autonomous navigation
  - evaluate trading strategies
  - learning to fly a helicopter

A typical ML task: make a recommendation

- Generate personalised recommendation, through collaborative filtering of sparse user ratings.

Table 1: Books review ratings

User	Book1	Book2	Book3	book4	Book5	Book6
A	3	2		5		
B	4	4	3	0		5
C	1			3	2	
D		3	2		4	

- Example applications:
  - which movies or books might a particular user enjoy
  - which news articles are relevant for a particular user
  - which song should be played next

Characteristics of an ML model

- Consider the ML task of polynomial regression: given training data $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$, fit an n^{th} degree polynomial of the form

$$y = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \tag{1}$$

- The degree of the polynomial is a hyperparameter of the model. Its value should be chosen prior to training (akin to model selection).

- The coefficients $a_n, a_{n-1}, \cdots, a_1, a_0$ are parameters for which optimal values are learned during training (model fitting).

Learning as an optimisation problem

- Consider a model with parameter set θ. Training involves finding values for θ that lets the model fit (or optimised) the data.
- The idea is to minimise an error, or difference between model predictions and true output in the training set.
  - We define an error/loss/cost function $\mathcal{E}$ of the model parameters
  - A popular minimisation technique is gradient descent:

$$\theta^{t+1} = \theta^t - \eta \frac{\partial \mathcal{E}}{\partial \theta} \tag{2}$$

  - The learning rate η can be updated (by some policy) to aid convergence.
  - Common issues to look out for!
    - Convergence to a local, non-global minimum.
    - Convergence to a non-optimium (if the learning rate becomes too small too quickly).

Probability

- Thinking about ML in terms of probability distributions can be extremely useful and powerful, esp. for reasoning under uncertainty.
- Your training data are samples from some underlying distribution, and a crucial underlying assumption is that that data in the test set are sampled from the same distribution.
- Discriminative modelling:
  Predict a most likely label y given a sample x, i.e. solve $\arg\max_y p(y|x)$.
  Find the boundary between classes, and use it to classify new data.
- Generative modelling:
  Find y by modelling the joint $p(x, y)$, i.e. solve $\arg\max_y p(x|y)p(y)$.
  Model the distribution of each class, and match new data to these distributions.

Practical considerations

- Let's take a closer look at a few practical aspects of $\cdots$
  - data collection and features
  - models and training
  - improving your model

Practical 1: data collection and features

- Implementing linear regression with stochastic gradient descent from scratch in python:

The core of many ML algorithms is optimisation.

Wine Quality Dataset

  - To find a good set of model parameters based on the given data.

Using Python for machine learning: core packages

The version numbers of the major Python packages that were used for writing this book are listed below. Please make sure that the version numbers of your installed packages are equal to, or greater than, those version numbers to ensure the code examples run correctly:

- NumPy 1.9.1
- SciPy 0.14.0
- scikit-learn 0.15.2
- matplotlib 1.4.0
- pandas 0.15.2
- Jupyter Notebook

Data exploration and cleanup

- Explore your data: print out a few frames, plot stuff, eyeball potential correlations between inputs and outputs or between input dimensions, $\cdots$
- I Deal with dirty data, missing data, outliers, etc. (domain knowledge crucial)
- Preprocess or filter the data (e.g. de-noise)?
  Maybe not! We don't want to throw out useful information!
- Most learning algorithms accept only numerical data, so some encoding may be necessary. E.g. one-hot encoding on categorical data:

ID	Gender
0	female
1	male
2	not specified
3	not specified
4	female

ID	male	female	not specific
0	0	1	0
1	1	0	0
2	0	0	1
3	0	0	1
4	0	1	0

Feature engineering

- Until recently, this has been viewed as a very important step. Now, since the surge of DL, it's becoming almost taboo in certain circles!
  With enough data, the machine should learn which features are relevant
- But in the absence of a lot of data, careful feature engineering can make all the difference
  (good features $>$ good algorithms).
- Features should be informative and discriminative.
- Domain knowledge can be crucially useful here.
  Which features might cause the required output?
- A few feature selection strategies:
  - remove features one-by-one and measure performance
  - remove features with low variance (not discriminative)
  - remove features that are highly correlated to others (not informative)

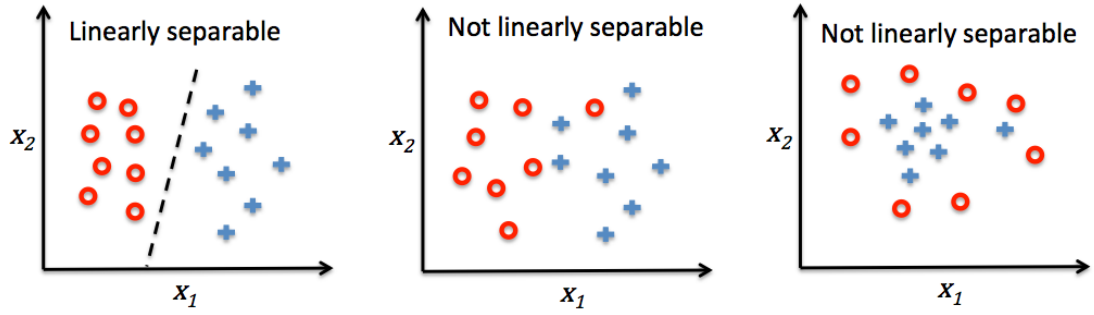
Feature transformations

- Feature normalisation can help remove bias in situations where different features are scaled differently

House	Size (m^2)	Bedroom
0	208	3
1	460	5
2	240	2
3	373	4

House	Size	Bedroom
0	-0.956	-0.387
1	1.190	0 1.162
2	-0.684	-1.162
3	-0.449	0.387

- Dimensionality reduction (e.g. PCA).



- Potential downside: dimensions lose their intuitive meaning

Practical considerations

- Let's take a closer look at a few practical aspects of...
  - data collection and features
  - models and training
  - improving your model

Choosing a model

- The choice depends on the data and what we want to do with it...
  - Is the training data labelled or unlabelled?
  - I Do we want categorical/ordinal/continuous output?
  - How complex do we think is the relationship between inputs and outputs?
- Many models to choose from!
  - Regression: linear, nonlinear, ridge, lasso,...
  - Classification: naive Bayes, logistic regression, SVM, neural nets,...
  - Clustering: k-means, mean-shift, EM,...
- No one model is inherently better than any other. *(the "no free lunch" theorem)*
  It really depends on the situation.
- Choosing the appropriate model and tuning its hyperparameters can be tricky, and may have to become part of the training phase.

Measuring performance

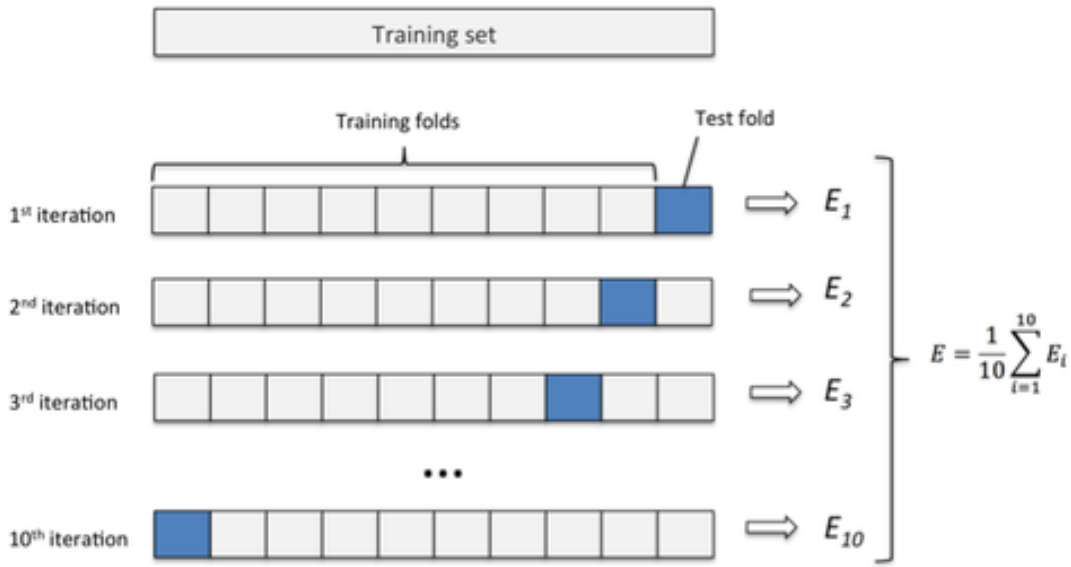
- The value of the loss function after training (or how well training labels are reproduced) may give insight into how well the model assumptions fit the data, but it is not an indication of the model's ability to generalise!

- Withhold part of our data from the training phase: the test set.
- Can we use our test set to select the best model, or to find good values for the hyperparameters? No! That'd be like training on the test set!
  The performance measure on the test set should be unbiased.
- Instead, extract a subset of the training set: the validation set.
- Train on the training set, and use the validation set for an indication of generalisability and to optimise over model structure and hyperparameters.

K-fold cross validation

- The training/validation split reduces the data available for training.

- Idea: partition the training set into $\mathcal{K}$ folds, and give every fold a chance to act as validation set. Average the results, for a more unbiased indication of how our model might generalise



Performance metrics

- How exactly do we measure the performance of a trained model (when validating or testing)?

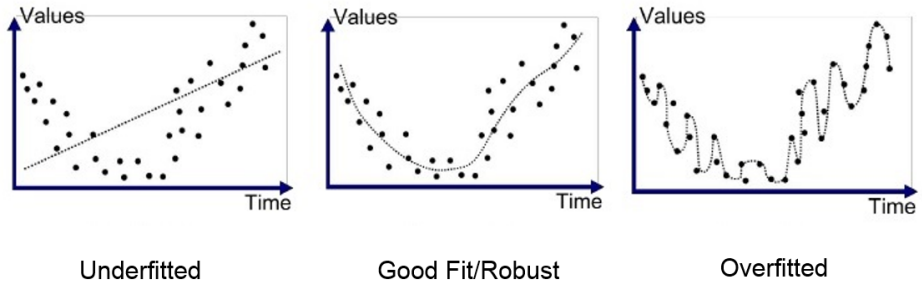
- Many options! Remember: it is very important to be clear about which metric you use. *A number by itself doesn't mean much.*
  - Regression: RMSE, correlation coefficients,...
  - Classification: confusion matrix, precision and recall,...
  - Clustering: validity indices, inter-cluster density,...
  - Recommender systems: rank accuracy, click-through rates,...
- Note: most of these compare model output to a "ground truth"
- Based on the application, which kinds of errors are more tolerable?
- It might also be insightful to ponder the statistical significance of your performance measurements.

Practical considerations

- Let's take a closer look at a few practical aspects of $\cdots$
  - data collection and features
  - models and training
  - improving your model

Underfitting and overfitting

- We want models that generalise well to new (previously unseen) inputs.
- Reasons for under-performance include underfitting and overfitting.
  Example: polynomial regression



- Underfitting (high model bias): the model is too simple for the underlying structure in the data.
- Overfitting (high model variance): the model is too complex, and learns the noise in the data instead of ignoring it.

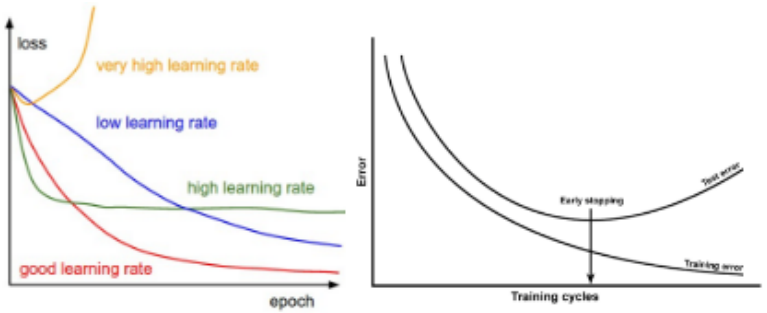
The bias-variance tradeof

- How can we find the sweet spot between underfitting and overfitting? *Use the validation set!*

Table 2: Measuring model performances

	Under fit	Good fit	Over fit
performance on training set	bad	good	very good
performance on validation set	bad	good	bad

- We want to keep training (keep decreasing the training error) only while the validation error is also decreasing.



Potential remedies

- To avoid overfitting, try,$\cdots$
  - a simpler model structure
  - a larger training set (makes the model less sensitive to noise)
  - regularization (`penalize model complexity during optimization`)
  - bagging (train different models on random subsets of the data, and aggregate)
- To avoid underfitting, try, $\cdots$
  - a more complex model structure
  - more features (more information about the underlying structure)
  - boosting (sequentially train models on random subsets of the data, focussing attention on areas where the previous model struggled)
- Note: it seems as if, no matter what, more data is better!

Summary

- We touched on what ML is and why it exists, and listed a few examples of typical ML tasks.
- We viewed the learning process as an optimization problem, and mentioned the importance of considering ML in terms of probability distributions.
- We ran through a few practical considerations concerning data cleanup, feature selection, model selection and training, measuring performance, and improving performance.