Machine Learning *ENGG*6500, Fall 2016*

Assignment 2

Due Date: 9:40am Tuesday, November 1, 2016
(Assignments should be submitted as PDFs to CourseLink Dropbox)

This assignment counts for 10% of the final grade.
There are a total of 66 points in this assignment.

Deliverables

Each student should submit the required materials to Dropbox before the start of the lecture on the due date. Late reports will not be accepted.

Reports should not exceed eight (8) pages, including figures and tables but excluding cover page.

Technical Presentation Requirements

Include a cover page indicating your name, title of work, course, instructor's name and date. Assignments will be judged on the basis of visual appearance, the grammatical correctness and quality of writing, and the visual appearance and readability of any graphics, as well as their contents. Please make sure that the text of your report is well-structured, using paragraphs, full sentences, and other features of a well-written presentation. Use itemized lists of points where appropriate. Text font size should be between 10 and 12 point. Solutions for Part 1 may be neatly hand-written.

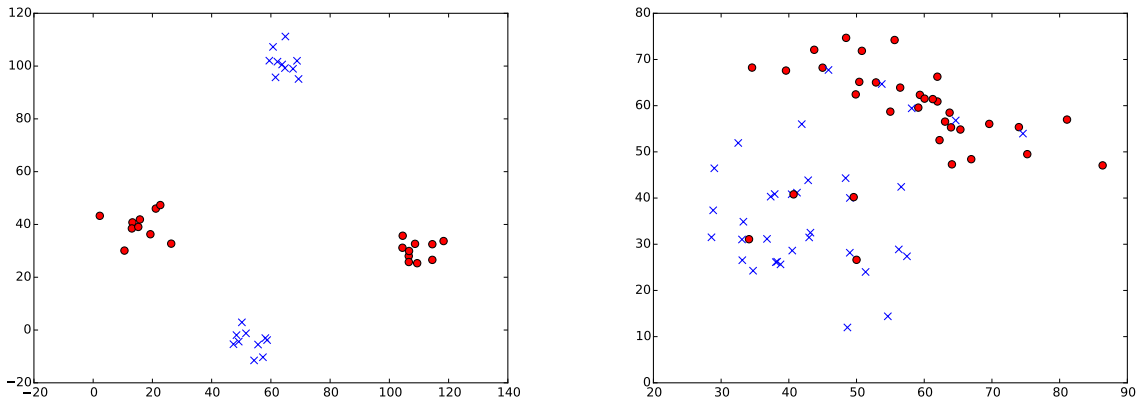
1 Theory

1. This question asks you to show your general understanding of underfitting and overfitting as they relate to model complexity and training set size. Consider a continuous domain and a smooth joint distribution over inputs and outputs, so that no test or training case is ever duplicated exactly. Indicate on your vertical axes where zero error is and draw your graphs with increasing error upwards and increasing complexity/training set size rightwards.

   (a) (5 points) For a fixed training set size, sketch a graph of the typical behaviour of training error rate versus model complexity in a learning system. Add to this graph a curve showing the typical behaviour of test error rate (for an infinite test set drawn independently from the same input distribution as the training set) versus model complexity, on the same axes. Mark a vertical line showing where you think the most complex model your data supports is; chose your horizontal range so that this line is neither on the extreme left nor on the extreme right.

(b) (5 points) For a fixed model complexity, sketch a graph of the typical behaviour of training error rate versus training set size in a learning system. Add to this graph a curve showing the typical behaviour of test error rate (again on an iid infinite test set) versus training set size, on the same axes.

2. (12 points) This question requires you to investigate the K-nearest neighbours algorithm which was not covered in lectures but is a simple and fundamental technique in Machine Learning. Pictured in Fig. 1 are two datasets for classification. For each dataset, explain whether K-nearest neighbours (with $K = 1$ and $K = 5$) and Logistic Regression can be expected to do well on these datasets, assuming that the datasets here are "typical" for the problem. Give a brief explanation (one or two sentences) in each case. It is recommended to make a table with six cells, one for each dataset/algorithm combination.



(a) Dataset 1 (b) Dataset 2

Figure 1: Datasets

3. Consider binary classification in which data are labeled by a random variable y, which takes its values from a discrete set, $y \in \{y_0, y_1\}$. The observed data are in the form of d-dimensional random vectors, $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$. In a simple *generative model* called a *directed belief net*, we must specify marginal probabilities $p(y)$ and conditional probabilities $p(\mathbf{x}|y)$. Given a particular vector $\mathbf{x}$, we wish to assign it to one of the two classes. To this end, we use Bayes' rule and calculate the relevant posterior probability $p(y_0|\mathbf{x})$.

   (a) (1 point) Apply Bayes' rule and write out the posterior.

   (b) (3 points) Show that the posterior probability can be written in the form of the logistic function. Hint: expand the denominator, then introduce the exponential.

   (c) (2 points) In the belief network, we must choose a particular form for the conditional probabilities $p(\mathbf{x}|y)$ (the *class-conditional densities*). Let us assume that they are multivariate Gaussians with identical covariance matrices Σ and per-class means $\boldsymbol{\mu}_i$:

$$p(\mathbf{x}|y_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)}$$

This is called the *class-conditional Gaussian* model. Arrange the posterior such that

$$p(y_0|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$

and write out the expressions for $\mathbf{w}$ and b.

(d) (1 point) If we just parameterize $p(y_0|\mathbf{x})$ directly, what model do we have?

2 Practice

You'll be revisiting the "StumbleUpon Evergreen Classification Challenge" from `kaggle.com`. The data that was provided for Assignment 1 contained only the numerical features from the StumbleUpon dataset. This time you'll be integrating some additional features based on the text content of each training/validation case, as well as implementing some techniques for speeding up mini-batch training. We highly encourage you to visit the webpage for the challenge, so that you understand the difference between the numerical features and the raw text data.

Download the zipped code/data file (`A2.zip`). You'll find 6 files in the archive:

- `A2.py`/`A2_tf.py`: a modified version of the code you've seen in `A1.py`
- `train.tsv`: training data from `kaggle.com`
- `test.tsv`: test data from `kaggle.com`
- `prep_data.py`: python script for parsing `train.tsv` and `test.tsv` and saving the data to pickled numpy arrays.
- `train_lda.py`: python script for learning text based features using Latent Dirichlet allocation (LDA)

NOTE: The `train_lda.py` script depends on the gensim package for training the LDA model. This can be installed via `pip` (e.g. `pip install gensim`).

Running the `prep_data.py` script will read the `tsv` files, parse their contents, split the training data into training and validation sets and save three files in python's pickle serialization format:

- `numerical_features.pkl`: pickle of the numerical features for training, validation, and test data along with the targets. Note that the `test.tsv` file does not provide targets for the test examples. Instead of targets, we save the `urlids` for each test case so that we can generate a submission file if you choose to submit your predictions to the `kaggle.com` competition.
- `word_features.pkl`: 'bag of words' features for each training and test case. These are saved as a tuple of `(num_document x vocab_size)` sparse matrices for training, validation, and test data.
- `word_names.pkl`: a list of words associated with each dimension of the bag of word representations.

1. (2 points) Run the `prep_data.py` script in order to create the files described above. Although we provide this preprocessing script, it is important that you go through it and make sure

you understand what each line does. If a method is unclear, read its documentation. If it's still unclear, ask one of your peers. Unpickle each of the files created by `prep_data.py` and examine the data structure(s) that are stored in each file. Describe the preprocessing step in 200 words or less. Is there anything you might do differently?

2. (5 points) Run `train_lda.py` to fit an LDA topic model to the bag of words representations of the text attributes of the training data. Once the model is finished training (it should just take a couple minutes), the script will print the top 15 words of each of the 30 latent (ie. hidden) topics discovered.

   Remember that in LDA, each document is assumed to be a mixture of a number of topics. What do the discovered topics tell you about the types of web pages in the training data? Do any topics stand out? Are there any 'topics' that were discovered by LDA that don't fit with the conventional notion of what a topic is?

   [*Optional*] Feel free to experiment with various settings for training the LDA model. For example, how do the topics change with the number of topics? If you do chose to experiment, be sure to use the original settings for the remainder of your experiments.

3. (10 points) The `train_lda.py` script will create a `lda_features.pkl` file with the learned topic features for the training, validation, and test data. The `load_evergreen` function in `A2.py` will append these features to the numerical features that were used in Assignment 1, creating a total of 66 input dimensions (36 numerical features + 30 topic features). As you did in Assignment 1, modify `A2.py`, experimenting with the number of hidden units, learning rate, and the number of epochs to find a setting that works well with the new data. Report your results. Do the additional features help classification accuracy? Why or why not?

4. (20 points) Five supplemental videos have been posted on Courselink along with this assignment handout. They give practical advice around training neural networks using mini-batch gradient descent. Specifically, videos 6c, 6d, and 6e describe a number of techniques for speeding up gradient descent. These include:

   - Momentum
   - Nesterov accelerated gradient
   - Adaptive learning rates
   - Rmsprop

   Implement any *two* of these methods, modifying `A2.py` as needed. For each method, experiment with different values of the associated hyperparameters and report your results (note: you may also need to tweak the learning rate). Include plots of cross entropy vs. epoch for training and validation data. How do the training and validation accuracies and/or convergence times compare to what you found in question 2? Do the cross entropy and/or accuracy differ between training and validation data as training progresses? Why or why not? If the methods can be combined, try using both. Does this improve accuracy or convergence time?

Latent Dirichlet Allocation

This section provides a bit of background on the method used in Part 2 to extract low-dimensional features from the raw bag of words data. There are no additional assignment questions. The following description and notation is adapted from Kevin Murphy's text "Machine Learning: A Probabilistic Perspective" (section 27.3).

Consider the task of modeling a collection of N documents made of words $y_{il} \in \{1 \dots V\}$, where V is the number of words in a vocabulary, i indexes documents and the i^{th} document has L_i words. We would like to construct a model of the form $p(\mathbf{y}_{i,1:L_i})$, that is, the probability of observing the sequence of words $1 \dots L$ in document i.

The simplest approach is to use what's called a finite mixture model. This associates a single discrete latent variable, $q_i \in \{1, \dots, K\}$ with every document, where K is the number of clusters. We will use a discrete prior, $q_i \sim \text{Cat}(\boldsymbol{\pi})$. For variable length documents, we can define $p(y_{il}|q_i = k) = b_{kv}$, where b_{kv} is the probability that cluster k generates word v. The value of q_i is called a **topic** and the vector $\mathbf{b}_k$ is the k^{th} topic's word distribution. The likelihood then has the form:

$$p(\mathbf{y}_{i,1:L_i}|q_i = k) = \prod_{i=1}^{L_i} \text{Cat}(y_{il}|\mathbf{b}_k) \tag{1}$$

The distribution over the observed data given by the mixture is given by:

$$p(\mathbf{y}_{i,1:L_i}) = \sum_k \pi_k \left[\prod_{i=1}^{L_i} \text{Cat}(y_{il}|\mathbf{b}_k) \right] \tag{2}$$

Generating data from this model is very simple. For document i we choose a topic q_i from the distribution $\boldsymbol{\pi}$, call it k, and then for each word $l = 1 : L_i$, we pick a word from $\mathbf{b}_k$.

The major limitation with the mixture of multinomials is that every document is assigned to a single topic, $q_i \in \{1 \dots K\}$, drawn from a global distribution $\boldsymbol{\pi}$. In Latent Dirichlet Allocation (LDA), every word is assigned to its own topic, $q_{il} \in \{1 \dots K\}$, drawn from a document-specific distribution $\boldsymbol{\pi}_i$. Each document thus belongs to a distribution over topics, rather than a single topic. The complete generative model is as follows:

$$\boldsymbol{\pi}_i|\alpha \sim \text{Dir}(\alpha \mathbf{1}_K) \tag{3}$$
$$q_{il}|\boldsymbol{\pi}_i \sim \text{Cat}(\boldsymbol{\pi}_i) \tag{4}$$
$$\mathbf{b}_k|\gamma \sim \text{Dir}(\gamma \mathbf{1}_V) \tag{5}$$
$$y_{il}|q_{il} = k, \mathbf{B} \sim \text{Cat}(\mathbf{b}_k) \tag{6}$$

In English, that means we first sample a document-specific distribution over topics $\boldsymbol{\pi}_i$ from a Dirichlet prior with hyperparameter α. Then, for each word in the document, we sample a word-specific topic q_{il} from $\boldsymbol{\pi}_i$ which is a multinomial over K topics. The topic-specific distributions

over words $\mathbf{b}_k$ are sampled from a Dirichlet prior with hyperparameter γ. Finally, a word, given its topic k is sampled from distribution $\mathbf{b}_k$ which is a multinomial over V words in the vocabulary. This process is illustrated in the graphical model in Figure 2.

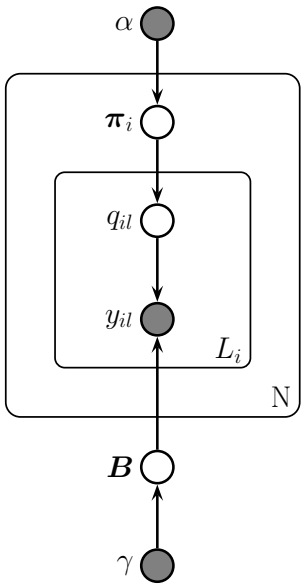


Figure 2: Latent Dirichlet allocation (LDA) graphical model.

The feature representation actually used by Part 2 for a given document i is the expected value of $\boldsymbol{\pi}_i$ given the data.

Note that we've completely skipped the details of inference in the model, but that is beyond the scope of this course. More information on LDA and other latent variable models for discrete data can be found in the Murphy text, section 27. Several of these models are implemented in the `gensim` package.