1. Consider the problem of sentiment classification in Twitter, i.e. deciding whether a tweet expresses a positive or negative sentiment. A standard approach to the task is to learn a classifier using labelled data.

a) Using pseudocode describe the perceptron algorithm for binary classification learning [30%]

Input: $D_{train} = \{(x^1, y^1) \ldots (x^M, y^M)\}$
set $\mathbf{w}_0 = \mathbf{0}; c = 1$
for $i = 1$ **to** *maxIter* **do**
  $shuffle(D_{train})$
  for $(x, y) \in D_{train}$ **do**
    predict $\hat{y} = sign(\mathbf{w}_{c-1} \cdot \phi(x))$
    if $\hat{y} \neq y$ **then**
      update $\mathbf{w}_c = \mathbf{w}_{c-1} + y\phi(x)$
    else
      $\mathbf{w}_c = \mathbf{w}_{c-1}$
    $c = c + 1$
return $\frac{1}{c} \sum_{i=1}^{c} \mathbf{w}_i$

W is the learned weight for each word, phi(x) is the count of the feature x (can be word or n-grams), c is the count of iterations.

b) What features would you use to represent the tweets? [20%]

Word can be the feature to represent tweets. In details, to make prediction of a class of a word sequence, the feature can be count of each word (bow representation).

c) Consider the following dataset:

label	tweet
negative	*Very sad about Iran.*
negative	*No Sat off...Need to work 6 days a week.*
negative	*I'm a sad panda today.*
positive	*such a beautiful satisfying day of bargain shopping. loves it.*
positive	*who else is in a happy mood??*
positive	*actually quite happy today.*

Given this dataset for training, the features you defined in question 1(b) and the perceptron algorithm to learn the weights for them, give 5 examples of features that should have weights indicative of the positive class and 5 examples of features that should have weights indicative of the negative class, explaining why they would be obtained and arguing about whether they are useful. Construct 2 examples where the model learned given this dataset and the features defined would predict the incorrect class; provide justifications for your examples. [30

The defined feature is word. In this case, positively weighted features: beautiful, satisfying, loves, happy, mood; negatively weighted: sad, Iran, no, off, work, week. As long as the training set is small, there are words only occurs in pos or neg sentences which is weighted high in this class, however, they are not necessarily pos or neg, such as, week and work only exists in neg sentences so they are neg weighted; mood only exists in pos so it is pos weighted but they don't represent any sentiment. So some of the features is useless.

In the model trained by this set, 'Iran' and 'panda' are negatively weighted, when predicting 'Very excited about Iran' and 'I'm a glad panda today', the class will be negative because 'excited' and 'glad' represent pos sentiment but they didn't present in the train set so they have no weight. The prediction will be wrong.

d) Extend your approach to predict 3-way sentiment labels: positive, negative, neutral. Describe the changes required to both the learning algorithm and the features. [20%]

The classes are extended to 3. The prediction sign function should be replaced by:

Predict: y_hat = argmax_y(w^y·phi(x))

The update should be:

If y_hat != y then
w^y += phi(x)
w^y_hat -= phi(x)

The features is still words but 'neutral' is added to sentence labels.

2. Consider the task of named entity recognition, i.e. the recognition of names of persons, organizations and locations, and the sentence: *Andrew Green, resident of Sheffield, is a supporter of the football club Sheffield Wednesday.*

ja) Give the correct token-level named entity recognition labels for the above sentence and use it to discuss why named entity recognition is a challenging task. [15%]

The NER in example sentence uses the IOB format.

Andrew Green, resident of Sheffield, is a supporter of the football club Sheffield Wednesday.
B-PER I-PER O O B-LOC O O O O O O O B-ORG I-ORG

NER requires NE detection and NE classification.
For NE classification, it is hard to decide an NE like Sheffield to be a LOC or a B-ORG.
For NE detection, there are a lot of new NE appearing everyday, no corpus can cover fully (i.e. new football club established).

b) What would be the drawbacks of learning a classifier compared to sequence learning for the task of named entity recognition? Use the sentence above to explain them. [20%]

A classifier predicts individual labels at each position which cannot handle tag interactions (transition probabilities) so there is no probability included. NER is sequence-relative, for example, Green in the example sentence is I-PER but it can be a kind of colour which is O. Sheffield can be both LOC or ORG and Wednesday can be both O or ORG. To learn which label is more likely, sequence learning would be better.

c) Describe in detail how the structured perceptron can be applied to named entity recognition. Your answer should consist of three parts: inference, learning and features. You should describe the algorithms and equations required to specify the approach with reference to the above sentence.[45%]

Inference: The decoding (predicting) of a multiclass perceptron is:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} (\mathbf{w} \cdot \phi(x, y))$$

Extend it to Y^N:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}^{\mathcal{N}}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y})$$

Where Phi generates features capturing the compatibility between x and y. (i.e. word_tag count)

Learning:

Input: $D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1) \ldots (\mathbf{x}^M, \mathbf{y}^M)\}$
set $\mathbf{w} = 0$
for $(\mathbf{x}, \mathbf{y}) \in D_{train}$ **do**
  predict $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}^{\mathcal{N}}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y})$
  if $\hat{\mathbf{y}} \neq \mathbf{y}$ **then**
    update $\mathbf{w} = \mathbf{w} + \Phi(\mathbf{x}, \mathbf{y}) - \Phi(\mathbf{x}, \hat{\mathbf{y}})$
return $\mathbf{w}$

W is the weight of each feature. Phi(x,y) generates features capturing the compatibility between sentence x and tag sequence y.

Features:

If we take 1st markov assumption, the feature can be:

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{N} \phi(y_n, y_{n-1}, \mathbf{x}, n)$$

d) Conditional random fields can also be used for named entity recognition. Explain how they are related to the structured perceptron using appropriate equations and discuss their relative strengths and weaknesses. [20%]

Structured perceptron generalized classification problem to predicting structured objects (sequence prediction) :

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} (\mathbf{w} \cdot \phi(x, y)) \rightarrow \hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}^{\mathcal{N}}} \mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y})$$

while CRF extends logistic regression to predict arbitrary structures:

$$P_{MLR}(y = \hat{y}|x; \mathbf{w}) = \frac{\exp(\mathbf{w} \cdot \phi(x, \hat{y}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w} \cdot \phi(x, y'))} \rightarrow P_{CRF}(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}^{\mathcal{N}}} \exp(\mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y}'))}$$

When using linear chain CRFs, The feature Phi is the same with 1st order M perceptron:

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{n=1}^{N} \phi(y_n, y_{n-1}, \mathbf{x}, n)$$

Although the predicting result inference with viterbi is the same between SP and CRF, ignoring the normalization factor:

$$\arg\max_{\mathbf{y} \in \mathcal{Y}^{\mathcal{N}}} P_{CRF}(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}^{\mathcal{N}}} \sum_{n=1}^{N} \mathbf{w} \cdot \phi(y_n, y_{n-1}, \mathbf{x}, n)$$

The normalization factor has to score all possible sequences which is expensive even with viterbi:

$$L_{reg}(\mathbf{w}; D_{train}) = L(\mathbf{w}; D_{train}) + \lambda R(\mathbf{w})$$

In conclusion, both SP and CRF is capable of structured object prediction and has flexible features. SP is faster but doesn't include probabilities while CRF has the same strength with SP (feature flexibility) and it's score interpretation is richer (not only one is larger than another one but also represents probability) but the computational cost is higher.

3. a)What are distributed word representations? How do they compare against one-hot encoding? [20%]

Distributed word representations are known as embedding. It involves a mathematical embedding from space with many dimensions per word to a continuous vector space with a much lower dimension.

In one-hot code, every word is equally different from every other word. So, we can not get the relation between words. In distributed word representation, it is clear to calculate the similarity between the word vector.

b)Describe how one can obtain sparse distributed word representations from a corpus by counting word contexts, giving details on how the choices of context window size, context representation and count post-processing affect the representations learned.[30%]

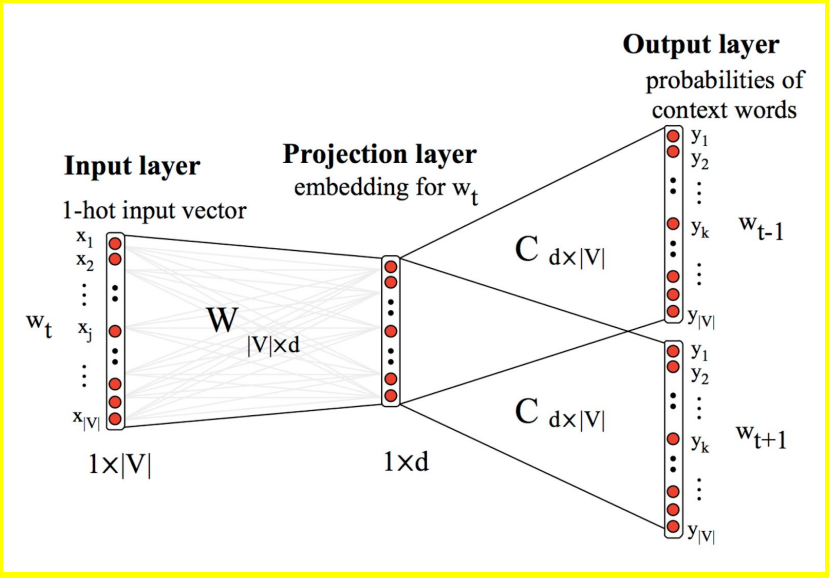
The first step is building the word-word matrix, and its size is the |V|*|V|. |V| is the size of the corpus. Then, choose the window size of the context. The shorter the windows, the more syntactic the representation is, and the longer the windows, the more semantic the representation is. Then, each row in this matrix is the vector of each word which contains many 0. So, the representation is sparse. A word is now defined by a vector over counts of context words.

Subsampling is a method of diluting very frequent words, akin to removing stop-words. We found their impact on performance comparable, and report results of only the “dirty” variant.

Each word has two embeddings. One is the word embedding, the other one is context embedding. When dealing the final embedding, it can discard the context word embeddings, add them, or concatenate them with target word embedding.

c)Describe in detail the skip-gram model and how it can be used to learn dense distributed word representations from a corpus, using appropriate equations and diagrams. [30%]

A skip-gram is a 3 layers language model to predict context word in a given position.



The input layer is one-hot vector. With the word embedding W, it projects the input layer into a new vector with lower dimension so the vector is denser. Then, using context embedding C and softmax function, it can output the context word probabilities:

$$P(w_c|w_t) = \frac{\exp(\mathbf{w_c} \cdot \mathbf{w_t})}{\sum_{w_c' \in V} \exp(\mathbf{w_c}' \cdot \mathbf{w_t})} \qquad \text{(each word } w \text{ is a word vector } \mathbf{w}\text{)}$$

Word embedding for each word W is trained by backpropagation.

d)Give one example of intrinsic word representation evaluation and one example of extrinsic word representation evaluation. [20%]

Intrinsic:

in-context similarity: substitute a word in a sentence without changing its meaning.

analogy: Athens is to Greece what Rome is to ...?

Extrinsic:

use the result to improve performance in a task: the bag of word vector.

1. a) What is a language model? Describe the data required train a language model, and what a trained model is expected to return.

[15%]

A language model is a probability distribution over the sequence of words. It takes a set of documents x as train data. D_train = {X1…..XM}, Xm = [x1….xn].

The trained model should return the probability of a given sequence:

p(X) = p(x1,x2…..xn)

b) What is the equation for the probability of a sentence? How is this probability approximated in an n-gram language model? Explain your terms.

[25%]

Given sentence x, probability p(x) = p(x1…..xn) = ∏ p(x_n|x_n-1….x1) which is the product of all the features in this sentence.

In an n-gram model, a k-th order markov assumption (k is the number of grams) is made:

p(x_n|x_n-1…..x_1) = p(x_n|x_n-1….x_n-k)

p(x) = ∏ p(x_n|x_n-1….x_n-k)

$$P(x|context) = \frac{P(context, x)}{P(context)} = \frac{counts(context, x)}{counts(context)}$$

c) What is add-1 smoothing? Why is it important for language modelling? Describe using equations how add-1 smoothing is applied to the bigram language model.

[20%]

Add-1 smoothing is to assume unseen features have been seen once. No matter how large a training corpus is, there are always unseen features in test set (zero probability), smoothing is important because it helps avoid zero probabilities.

For a bigram model, add-1 smoothing is applied:

P_add-1 = (count(x_n-1,x_n) + 1) / (counts(x_n-1) + |V|)

Where V is the vocabulary, |V| is the size of vocabulary.

d) Describe back-off using equations to illustrate the key concepts. Explain how back-off is applied to language modelling making clear how we ensure that the resulting model is a valid probability distribution?

[20%]

When count(context, target) is at low freq (or even zero prob), the prediction is unreliable. Using back-off when the count is lower than cut-off freq can avoid this:

P_BO(x_n|x_n-1….x_n-k) =

P*(x_n|x_n-1….x_n-k) if count>cut-off freq

α^x_n-1...x_n-k·P_BO(x_n|x_n-1….x_n-k+1) otherwise

Note that P* is discounted probability and alpha is back-off weight.

We discount P* to make sure the summed prob is 1 so that the resulting model is a valid probability distribution.

e) Language models can be evaluated intrinsically and extrinsically. Discuss the advantages and disadvantages for each approach and describe two methods for intrinsic and three for extrinsic evaluation.

[20%]

Intrinsic approaches design a measurement to evaluate the LM, it is fast and good to use at the development stage but it doesn't directly on the task which means it doesn't necessarily lead to high performance on the given task. The measurement can be entropy (averaged log probabilities) or perplexity. There is another disadvantage of these two methods is that they cannot evaluate non-probability LMs.

Extrinsic approaches evaluate LMs directly to the task such as speech recognition, spelling correction and machine translation. It evaluates LM by these systems accuracy. It can use disjoint test set, cross-validate or leave one out error to determine the best LM. It has the most reliable evaluation result but time-consuming.

2. Information extraction is a well-known task in natural language processing. It is typically decomposed into two steps: named entity recognition and relation extraction. Consider the following sentences:

Michael Jordan, shooting guard for Chicago Bulls, was born in Brooklyn.
Michael Jordan never played for his home city, Brooklyn.
The hometown of Michael Jordan, CEO of PepsiCo, was Kansas City.
Michael Jordan, native of Kansas City, studied in Yale.

a) Using examples from the above sentences:

(i) Describe how you could extract data to train a named entity recognition model. [10%]

The required data is a set of word sequence x = [x_1...x_n] with label sequence y which contains NE tags. They are combined in tuples: D_train = {(x^1, y^1)...(x^M, y^M)}.

(ii) Name two algorithms that would typically be used to learn named entity recognition models. [10%]

Structured Perceptron, conditional random fields, hidden markov model.

(iii) Describe the features that would typically be used by a named entity recognition system. [10%]

In structured perceptron, there are several formats of features, like the current word with the current label(Micheal_PER), and the previous labels with the current label(PER_NONE).

b) Using examples from the sentences in the beginning of the question:

(i) Describe how you could extract training data to train a relation extraction model for the relation born in. [10%]

The required data is the rel_train which can be obtained by:

Input: $D_{train} = \{(\mathbf{x}^1, G^1_{\mathbf{x}}) \ldots (\mathbf{x}^M, G^M_{\mathbf{x}})\}$
set $rel_{train} = \emptyset$
for $(\mathbf{x}, G_{\mathbf{x}}) \in D_{train}$ **do**
  find entities/triggers $E = NER(\mathbf{x})$
  for $(e_1, e_2) \in E \times E$ **do**
    $rel_{train} = rel_{train} \cup (\phi(\mathbf{x}, e_1, e_2), label(e_1, e_2, G$
return classifier trained on rel_{train}

Where X is sentences and G is the graph of sentences.

(ii) Name two algorithms that would typically be used to learn relation extraction models. [10%]
SVM, CRF

(iii) Describe the features that would typically be used by a relation extraction model. [10%]

BoW representations between every 2 NEs.

3. Consider the problem of topical text classification, i.e. deciding whether a piece of text talks about certain topic or not. A standard approach to the task is to learn a classifier using labelled data.

a) Using equations, describe the model of logistic regression for classification. [20%]

$$P_{MLR}(y = \hat{y} \mid x; w) = \frac{exp(w \cdot \varphi(x, \hat{y}))}{\sum_{y' \in Y} exp(w \cdot \varphi(x, y'))}$$

phi(x,y) is the feature function.

b) Using equations and pseudocode, describe how logistic regression can be trained using gradient descent. [30%]

The aim of gradient descent is to get the lowest w of the function through the $\nabla xf(x)$.

Input: $D_{train} = \{(x_1, y_1) \ldots (x_N, y_N)\}$, *learning rate* α
initialize $\mathbf{w}$
while *not converged* **do**
    update $\mathbf{w} = \mathbf{w} - \alpha \nabla_{\mathbf{w}} NLL(\mathbf{w}; D_{train})$
return $\mathbf{w}$

NLL means negative log likelihood.

c) What features would be appropriate to represent the texts? [10%]

Word can be the feature to represent tweets. In details, to make a prediction of a class of a word sequence, the feature can be the count of each word (bow representation).

d) Consider the following dataset:

class	text
sports	Wayne Rooney has announced his retirement from international football
sports	Joel Campbell, has returned from loan to play football
politics	This international committee promotes the elimination of racial discrimination
politics	13 years of UN negotiations aimed at restricting cyberwarfare collapsed

Given this dataset for training, the features you defined in question 3(c), and the logistic regression model combined with gradient descent to learn the weights, give 5 examples of features that should have weights indicative of the sports class and 5 examples of features that should have weights indicative of the politics class. Explaining why these features would be obtained from this dataset, and discuss whether they are useful for this task. Construct 2 examples where the model learned given this dataset and the features defined would predict the incorrect class; provide justifications for your examples.[40%]

(This question is similar to the 2017-1-c)

4. The perceptron and logistic regression are widely used classifiers in NLP.

a) Compare the perceptron with logistic regression in terms of feature flexibility and score interpretation with reference to the equations used to define these classifiers.[20%]

perceptron: $\hat{y} = \arg\max_{y \in \mathcal{Y}} (\mathbf{w} \cdot \phi(x, y))$

LR: argmax_y $P_{MLR}(y = \hat{y} | x; \mathbf{w}) = \frac{\exp(\mathbf{w} \cdot \phi(x, \hat{y}))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w} \cdot \phi(x, y'))}$

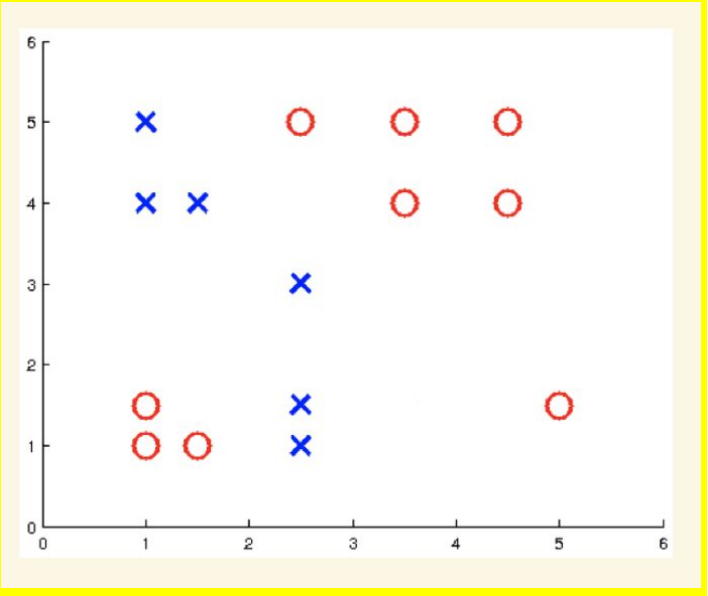
From equations, both algorithms are operating w*phi, the feature flexibility is the same.

Given same w, the decision is the same.

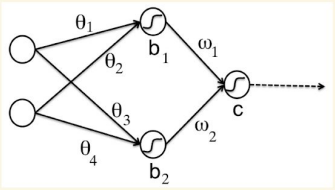
The score interpretation is different, a score in perceptron is just larger than another while in LR, the score also represents probability.

b) Both the perceptron and logistic regression are linear models. Explain the limitation of linear models using suitable equations and diagrams.[20%]

If the dataset is not linearly separable and cannot be learned with a linear model(like the diagram), the linear model will be useless.



c) Describe using equations and diagrams a multilayer perceptron with one hidden layer and explain how it can address the limitation of linear models. [30%]



$$h_1(\mathbf{x}, \theta_1, \theta_2, b_1) = \sigma(\theta_1 x_1 + \theta_2 x_2 + b_1)$$
$$h_2(\mathbf{x}, \theta_3, \theta_4, b_2) = \sigma(\theta_3 x_1 + \theta_4 x_2 + b_2)$$
$$f(\mathbf{x}, \theta, \mathbf{b}, \omega_1, \omega_2, c) = \sigma(\omega_1 h_1(\mathbf{x}, \theta, b_1) + \omega_2 h_2(\mathbf{x}, \theta, b_2) + c)$$

The first is the input layer. The b1 and b2 are bias of neurons of the hidden layer, and the c is the bias of the output layer. The output of one layer is the input of the next layer.

NN can draw several decision boundaries to classify data points.

d) Use pseudocode to describe the backpropagation algorithm for the multilayer perceptron. [20%]

Input: $D_{train} = \{(\mathbf{x}^1, y^1) \ldots (\mathbf{x}^M, y^M)\}$
initialise randomly $\theta, \omega, \mathbf{b}, c$
for $(\mathbf{x}, y) \in D_{train}$ **do**
  predict $\hat{y}, hidden_1, hidden_2 = f(\mathbf{x}, \theta, \mathbf{b}, \omega_1, \omega_2, c)$
  calculate loss $l = L(\hat{y}, y)$
  backpropagate l *via gradients : update* ω, c; *update* θ, β
return w

The aim of the backpropagation algorithm is to learn training data for the parameters, like phi,$\boldsymbol{\omega}$,$\boldsymbol{b}$ and c.

e) Explain why neural networks are prone to overfitting and how this problem can be avoided. [10%]

The neural network has a greater learning capacity. it makes overfitting more likely.

Regularization is the best way to handle this problem. It means adding a learning rate when processing the learning step.

Adding data. Cross validation