Exercise Sheet 7

Zena Al Khalili (7009151)
Sangeet Sagar (7009050
{zeal00001,sasa00001}@stud.uni-saarland.de

January 19, 2021

Exercise 7.1 - Norm penalty

(a)

In neural networks, we typically choose to use a parameter norm penalty Ω that penalizes only the weights at each layer and leaves the biases unregularized. The motivation behind Norm penalty is to restrict the weights (constraining the network) so it's less likely to overfit. It makes little sense to restrict the weights of the biases since the biases are fixed (e.g. b = 1). And the bias parameters don't contribute to the curvature of the model as much as weights, so there is usually no point in regularising them.

(b)

In directions that do not contribute to reducing the objective function, Components of the weight vector corresponding to such unimportant directions are decayed away through the use of the regularization throughout training. Since Age do not contribute as much to minimize cost function, it will be penalized more (large alpha) so the weight corresponding to Age will shrunk to have nearly zero magnitude, so the weight of the age feature will be decayed away. The cost function is already curved so adding alpha will not change its curve

(c)

The sparsity property induced by L1 regularization (L1 pushes some weights to be zero) has been used extensively as a feature selection mechanism. Feature selection simplifies a machine learning problem by choosing which subset of the available features should be used. L1 will cause the weight of feature Age to be zero, meaning the feature won't be used in the model. Figure 1 explains the effect of L1 Norm.

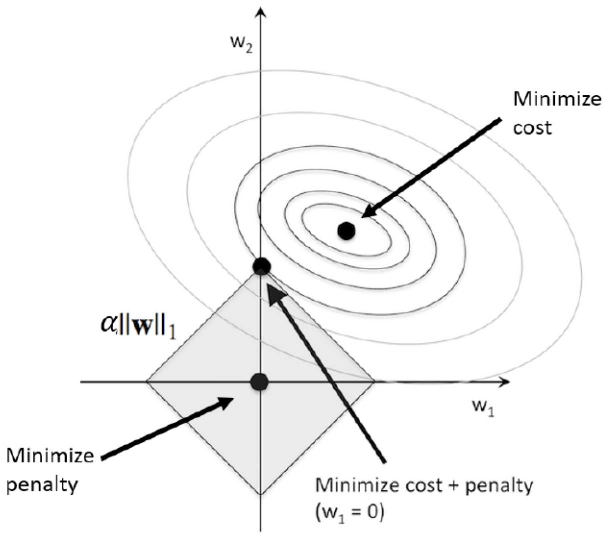

Figure 1: *L1 Norm penalty effect on some weights*

Exercise 7.2 - Dataset Augmentation

(a)

In the natural language processing (NLP) field, it is hard to augment text due to high complexity of language. Not every word can be replaced by other words such as a, an, the. Also, not every word has synonyms. Even changing a word might change the context of the sentence. That's why augmenting should be done carefully before training.

(b)

Applying Random Deletion from a sentence might affect the correct Sentiment analysis of the sentence. When a random word is removed from a sentence the possible meaning might change or can be classified as another class than what it should be.
For example: the Sentence: I feel not good today because yesterday I ate sushi.
Removing random words like 'not' might change the class of the sentence from negative to positive!

(c)

We should do data augmentation after splitting the data , so that the original data and the fake data are in the same fold, this way we avoid overfitting. Also, validation set are used to try to estimate how a method works on real world data, thus it should only contain real world data, not fake data.

(d)

Yes, the proportion of augmented data depends on the training set size, if we have enough training data to prevent the model from overfitting (i.e: learn too much from the training data) then we won't need much augmented data. On the other hand, if the training set size is small, we need to do more data augmentation to increase training set size ,to a good enough level, to prevent overfitting

Exercise 7.3 - Bagging and Dropout

a

In order to answer this question, I will lay some foundation and how does bagging work. Bagging in an ensamble method where we have different instances of the same classifier, which are trained on different samples of the same training data. So for a given dataset we construct multiple training sets $(T_1, T_2, ..., T_k)$ by *sampling with replacement* method. The learning algorithm is then applied to this training set and the procedure is repeated k times and the resulting k models are aggregated by uniform voting.
Now dropout works in a similar fashion. In dropout (refer to dropping units) we temporarily remove some nodes and all its connections. This way we get many thinned sub-networks onto which the dropout trains. This is approximately what we did in bagging- we are actually training these different networks on different subsets of the training data.

b

Adding to the above discussion, given a total of n nodes, we have a total of 2^n thinned-networks. So during inference its practically impossible to aggregate the outputs pf 2^n networks. What we do instead is we use a full neural network and scale the output of each node by the fraction of times it was on during training time. This infact significantly reduces overfitting.

c

Algorithm 1: Inverted dropout

```
Result: $D$: inverted dropout array
initialization: keep probability: P(keep) and a weight array: $w$;
$n$ <= length of weight array;
Here we compare each element of weight array with the P(keep);
for $k = 1...n$ do
    if $w[k] > P(keep)$ then
        binary array[k] <= True
    end
    binary array[k] <= False
end
Perform a element wise multiplication to get activations;
$D$ <= element-wise-multiply(binary array, $w$ );
For inverted dropout we scale the activations by the inverse of the keep
 probability;
$D <= D/P(keep)$;
```

Exercise 7.4 - Adverserial Training in NLP

Techniques of adverserial training

- [1] fooled Reading Comprehension models by inserting sentences to SQuAD without altering the answer of the question.
- [3] used gradients to determine the sensitivity of model towards certain words/characters and manually replacing them with common misspellings
- [4] a removal-addition strategy that constrained replacements such that grammar is preserved.
  Removal Example:
  This article will focus on summarizing data augmentation techniques in NLP.
  After Removing:
  This article will focus on summarizing data augmentation in NLP.
- [2] ranked words by their importance and replaced them with synonyms while maintaining grammatical sense and sentence meaning.
  Synonym Replacement example:
  This article will focus on summarizing data augmentation techniques in NLP.
  After Synonym Replacement:
  This write-up will focus on summarizing data augmentation methods in NLP.

References

[1] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328, 2017.

[2] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932, 2019.

[3] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4208–4215. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[4] Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *CoRR*, abs/1707.02812, 2017.