Deep Structured Learning (IST, Fall 2019)

Homework 2

Instructors: André Martins and Vlad Niculae
TAs: Gonçalo Correia and Ben Peters

Deadline: Friday, November 1, 2019.

Please turn in the answers to the questions below, **in English**, together with the code you implemented to solve them (when applicable). Please email your solutions in **electronic format** (a single zip file) with the subject "Homework 2" to:

`deep-structured-learning-instructors@googlegroups.com`

Hard copies will not be accepted.

Question 1

OCR with an autodiff toolkit. In Homework 1, you were asked to implemented a optical character recognition system from scratch. In particular, you had to wrote gradient backpropagation by hand. This time, you will implement the same system using a deep learning framework with automatic differentiation. A skeleton code for PyTorch is provided but if you feel more comfortable with a different framework you are free to use it instead.

1. (10 points) Implement a linear model with ℓ_2-regularized logistic regression, using stochastic gradient descent as your training algorithm. Train your model for 30 epochs and tune the following hyperparameters in your validation data:

   - The learning rate for the following values: $\{0.001, 0.01, 0.1\}$.
   - The regularization constant for the following values: $\{0., 0.001, 0.01\}$.

   For the best configuration, report it and plot two things: the training loss and the validation accuracy, both as a function of the epoch number. Report the final accuracy in the test set.

   In the skeleton code, you will need to implement the method `train_batch()` and the class `LogisticRegression`'s `__init__()` and `forward()` methods.

2. (15 points) Implement a feed-forward neural network with a single layer, using dropout regularization. Make sure to include all the hyperparameters and training/model design choices shown in Table 1. Use the values presented in the Table as default. Tune each of these hyperparameters while leaving the remaining at their default value:

   - The learning rate: $\{0.001, 0.01, 0.1\}$.
   - The hidden size: $\{100, 200\}$.
   - The dropout probability: $\{0.3, 0.5\}$.
   - The activation function: `relu` and `tanh`.

Number of Epochs	30
Learning Rate	0.01
Hidden Size	200
Dropout	0.3
Batch Size	64
Activation	ReLU
Optimizer	Adam

Table 1: Default hyperparameters.

- The optimizer: `SGD` and `Adam`.

Report your best configuration, make similar plots as in the previous question, and report the final test accuracy.

In the skeleton code, you will need to implement the class `FeedforwardNetwork`'s `__init__()` and `forward()` methods.

3. (5 points) Using the same hyperparameters as in Table 1, increase the model to 2 and 3 layers. Report your best configuration, make similar plots as in the previous question, and report the final test accuracy.

Question 2

Dynamic Programming. Your friend John the Dynamic lives in Lisbon and, depending on the weather conditions, he enjoys surfing, going to the beach, playing video games, and studying machine learning. His activities are governed by a hidden Markov model, where the hidden variables correspond to the weather (`Sunny`, `Windy`, and `Rainy`) and the observed variables correspond to `Surf`, `Beach`, `Videogame`, and `Study`. The emission and transition probabilities are shown in Tables 2–3.

	`Sunny`	`Windy`	`Rainy`
`Surf`	0.4	0.5	0.1
`Beach`	0.4	0.1	0.1
`Video game`	0.1	0.2	0.3
`Study`	0.1	0.2	0.5

Table 2: Emission probabilities: rows conditioned on columns.

	`Sunny`	`Windy`	`Rainy`
`Sunny`	0.6	0.3	0.2
`Windy`	0.3	0.5	0.3
`Rainy`	0.1	0.2	0.5

Table 3: Transition probabilities: rows (timestep $t+1$) conditioned on columns (timestep t).

1. John's activities for the past week were like shown in Table 4. **Assume that the weather on October 7 was rainy, and on October 15 it was sunny.** In class we saw two dynamic programming algorithms: the forward-backward algorithm (for which we provide an implementation in `hw2_decoder.py`) and the Viterbi algorithm (which you may need to implement as part of this exercise).

Monday, Oct 8	`Videogame`
Tuesday, Oct 9	`Study`
Wednesday, Oct 10	`Study`
Thursday, Oct 11	`Surf`
Friday, Oct 12	`Beach`
Saturday, Oct 13	`Videogame`
Sunday, Oct 14	`Beach`

Table 4: John's activities for the past week.

(a) (15 points) Knowing John's activities, what was the most likely weather for the past week? Which algorithm did you use to answer this question?

(b) (10 points) Let's suppose you made a bet with John where you receive 1€ for every day you guess the weather correctly, and you lose 1€ if your prediction is wrong. As in the previous question, you observed John's activities and the weather in October 7 and October 15. What would be your bet for the weather in the days from October 8 to 14? Which algorithm maximizes your expected profit?

2. (5 points) Actually, John never surfs two days in a row because (despite his nickname) he gets exhausted and he needs to rest at least one day before going back to the water. Can we accommodate this extra piece of knowledge in a hidden Markov model? Justify.

Question 3

Sequential OCR. So far, all your OCR experiments used models that try to predict each character independently from the others. In this exercise, you will solve the problem with structured prediction, using a linear sequential model.

1. (25 points) Exploiting the sequential structure of the characters (as they form words), implement the `update_weight` method in the `StructuredPerceptron` class in `hw2-q3.py`. As unigram features, use the pairwise features for pixels you used in homework 1. As bigram features, use only the conjunction of the two consecutive labels with no dependency on the pixels (i.e., a total of 26^2 bigram features). How does test accuracy compare with not using any structure? Hint: use the Viterbi implementation from the previous exercise, and don't forget to account for the start and stop symbols.

2. (15 points) Repeat the exercise above using a conditional random field (trained with stochastic gradient descent) instead of the structured perceptron. Implement the `update_weight` method in the `CRF` class.