# Homework 2

Due date: Thursday, April 4, 2019

*Submit* ***a single .zip file*** *online using the link in the PSI wiki. You must submit all of your code as well as a .pdf summary that contains and discusses all relevant plots. Acknowledge any references you use as well as any other students with whom you collaborate.*

## 1 Backpropagation

In this problem, you will derive the equations needed to calculate the gradient of the cost function in a feedforward neural network using backpropagation. This gradient is used within learning algorithms (such as gradient descent) that train the neural network.

Recall that the output from the $j^{\text{th}}$ neuron in layer $\ell$ is given by

$$a_j^{(\ell)} = g_\ell\left(z_j^{(\ell)}\right),$$

where $g_\ell$ is a non-linear activation function and

$$z_j^{(\ell)} = \sum_{i=1}^{n_{\ell-1}} a_i^{(\ell-1)} W_{ij}^{(\ell)} + b_j^{(\ell)},$$

with $W_{ij}^{(\ell)}$ representing the weights and $b_j^{(\ell)}$ the biases of the network. Assume that the cost function $C$ can be expressed as a sum over the dataset $\mathcal{D}$ as

$$C = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} c(\mathbf{x}),$$

and define the notation

$$\delta_j^{(\ell)} = \frac{\partial c}{\partial z_j^{(\ell)}}.$$

a) Show that the quantity $\delta_j^{(\ell)}$ can be expressed as

$$\delta_j^{(L)} = \frac{\partial c}{\partial a_j^{(L)}}\, g'_L\left(z_j^{(L)}\right),$$

$$\delta_j^{(\ell)} = \sum_{k=1}^{n_{\ell+1}} \delta_k^{(\ell+1)} W_{kj}^{(\ell+1)\,T} g'_\ell\left(z_j^{(\ell)}\right) \qquad (\text{for } \ell < L).$$

b) Show that the partial derivatives of the cost function with respect to the network's weights and biases can be calculated as

$$\frac{\partial c}{\partial W_{ij}^{(\ell)}} = a_i^{(\ell-1)} \delta_j^{(\ell)}, \qquad \frac{\partial c}{\partial b_j^{(\ell)}} = \delta_j^{(\ell)}.$$

## 2 Monte Carlo simulation of the Ising model

The objective of this problem is to generate spin configuration samples and calculate expectation values of macroscopic quantities for the two-dimensional classical Ising model. The data that you generate will then be used as input to a neural network in Problem 3.

Recall that the Hamiltonian for the classical Ising model is given by

$$H = -J \sum_{\langle i,j \rangle} s_i s_j,$$

where $s_i = \pm 1$ are the degrees of freedom living on the sites of the lattice, $\sum_{\langle i,j \rangle}$ is a sum over nearest neighbours and $J$ is a coupling strength. Here we consider a two-dimensional square lattice of length $L$, with $N = L^2$ sites and periodic boundary conditions.

In Lecture 5, we discussed general Monte Carlo methods for generating a chain of $M$ configurations $\mu_1, \mu_2, \ldots, \mu_M$ sampled from the probability distribution $p_\mu = \frac{1}{Z} e^{-\beta E_\mu}$. Here we will examine a "single spin-flip" algorithm, where each sample $\mu_i$ is related to the next sample $\mu_{i+1}$ by up to one spin flip. For this algorithm, the selection probabilities are given by

$$g(\mu \to \nu) = \begin{cases} \dfrac{1}{N} & \text{if } \mu \text{ and } \nu \text{ differ by a single spin flip,} \\ 0 & \text{otherwise.} \end{cases}$$

a) Prove that if we choose the acceptance probabilities such that

$$A(\mu \to \nu) = \begin{cases} e^{-\beta(E_\nu - E_\mu)} & \text{if } E_\nu > E_\mu, \\ 1 & \text{otherwise,} \end{cases}$$

then this algorithm satisfies detailed balance.

b) One issue with this algorithm is that it can get stuck in local energy minima at low temperatures. As we can see from the acceptance probabilities in part a), the probability of accepting moves that increase the energy is very small at low temperatures since $A(\mu \to \nu) \to 0$ as $T \to 0$ (*i.e.*, $\beta \to \infty$) when $E_\nu > E_\mu$.

On a finite-sized lattice at low enough temperatures, we know that our algorithm should ideally (almost always) sample states that have the ground state energy $E_{\text{gs}} = -JN$, which corresponds to either the all-spins-up or all-spins-down configuration. Give an example of a configuration $\mu^*$ with $E_{\mu^*} > E_{\text{gs}}$ where this single-spin flip algorithm could potentially get stuck at low temperatures such that $A(\mu^* \to \nu) \to 0$ for all $\nu$ as $T \to 0$.

c) Run the code `ising_mc.txt`, which implements the single-spin flip algorithm for various temperatures on an $L \times L$ lattice. Upon running this code, you should see the sampled configurations in a pop-up window. Each spin 'up' state is stored as +1 and each spin 'down' state is stored as -1. After the code finishes running, you should find data stored in the following three files:

- `x_L6.txt`: sampled spin configurations (one configuration per line),
- `y_L6.txt`: the labels corresponding to each configuration (0 when $T < T_c$, 1 when $T > T_c$),
- `T_L6.txt`: the temperature corresponding to each configuration (measured in units of $J$).

Write code that reads in the files `x_L6.txt` and `T_L6.txt` and plots the average magnetization of the sampled configurations as a function of temperature. Discuss whether or not your results behave as you would expect in the high- and low-temperature limits.

d) Change the parameters in the Monte Carlo code such that `num_T = 40`, `L=30`, `n_eqSweeps = 1000` and `n_measSweeps=500`. Also set `animate = False` so that the pop-up window no longer appears (since it slows down the calculations). Run the code again (which will require approximately 10 minutes) to generate the files `x_L30.txt`, `y_L30.txt` and `T_L30.txt`. These files will contain the 20 000 data points that you will use to train and analyze a neural network in Problem 3. As in part c), plot the average magnetization of the sampled configurations as a function of temperature. Discuss how this plot differs from the one in part c).

## 3 Classifying phases of the Ising model using neural networks

The objective of this problem is to train a feedforward neural network to classifying ferromagnetic (FM) and paramagnetic (PM) phases of the two-dimensional classical Ising model. You are encouraged to start by reading Reference [1].

Your dataset is the one generated in part d) of Problem 2 and consists of 20 000 Ising spin configurations on a $30 \times 30$ lattice. The labels represent whether a configuration was generated with $T < T_\mathrm{c}$ (label 0, corresponding to the FM phase in the thermodynamic limit) or $T > T_\mathrm{c}$ (label 1, corresponding to the PM phase in the thermodynamic limit). The temperature data file will not be used during the training, but will be used to study the accuracy of the network as a function of temperature in part e).

a) Modify your code from Tutorial 1 such that it reads in the Ising data and trains a neural network with one layer of 100 hidden neurons to learn the labels. Your neural network should output two-dimensional predictions and compare with labels in the one-hot representation. Modify the code such that it divides the data into three sets for training (70% of the data), validation (20% of the data) and testing (10% of the data) as discussed in Lecture 4. Be sure to shuffle the data in the files from part d) of Problem 2 in some way such that there are samples corresponding to each temperature in each of the three datasets. Run the code until the training accuracy converges and plot both the accuracy and the cost for the training and validation data as a function of training epoch. Discuss whether or not your plots show signatures of overfitting.

   **Note:** There is no need to tune the hyperparameters for this part of the problem since you will study hyperparameter tuning in parts b) and c).

b) Add in L2 regularization to the cost function in the code. Examine plots of the accuracy and cost versus training epoch to study the effect of adjusting the regularization hyperparameter, and discuss the effects that you observe.

c) Study the effects of adjusting other hyperparameters such as

   - the learning rate,
   - the training algorithm (gradient descent, stochastic gradient descent, Adam algorithm, etc.),
   - the number of neurons in the hidden layer,

   or any other hyperparameters such as the ones discussed at the end of Lecture 5. Summarize what you observe as you vary these hyperparameters. What is the highest accuracy you can achieve on the validation data?

d) After you have finished tuning the hyperparameters in part c), check the accuracy for the remaining 10% of data in the testing dataset using your best-trained neural network. How does this accuracy compare to that of the validation data?

e) Since we know the temperature corresponding to each of the test data points, we can study the neural network's performance as a function of temperature. With your best-trained network from part c), use the given testing data to make plots of

- the average accuracy as a function of temperature,
- the average output from each of the two output neurons as a function of temperature.

Compare your plots with Figure 1 of Reference [1]. Discuss the behaviour that you observe as you approach the critical temperature $T_c/J \approx 2.269$.

## References

[1] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017), `https://arxiv.org/abs/1605.01735`.