NNDL2024: Exercices & Assignments
Session 1 (21 Mar)

1 Basic definitions

Mathematical exercices

1. (6 pts) One proposal as activation function is the ”leaky ReLU”:

$$\psi(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{if } x < 0 \end{cases} \tag{1}$$

for some constant $0 \leq \alpha \leq 1$, typically small.

✓ (a) Show that the basic ReLU is a special case of this.

✓ (b) Show that the linear activation function is also a special case.

✓ (c) Show that the leaky ReLU can also be expressed as:

$$\psi(x) = \max(\alpha x, x) \tag{2}$$

✓2. (6 pts) Show that the tanh function is a very simple transformation of the logistic function. More precisely, denoting the logistic function by σ, we have

$$\tanh(x) = 2\sigma(2x) - 1. \tag{3}$$

3. (6 pts) Take a multi-layer neural network with linear activation function:

$$\mathbf{y}_K = \mathbf{W}_K \mathbf{W}_{k-1} \dots \mathbf{W}_1 \mathbf{x} = \mathbf{M}\mathbf{x} \ =: \mathbf{g}(\mathbf{x}) \tag{4}$$

where the total function given by the NN is denoted by $\mathbf{g}$. Assume the dimensions of $\mathbf{x}$ and $\mathbf{y}$ are both n, and the dimensions of all hidden layers are also equal to n.

✓ (a) Give a necessary and sufficient condition on $\mathbf{M}$ such that this $\mathbf{g}$ is injective, meaning that for any output $\mathbf{y}$ you can compute the original $\mathbf{x}$.

✓ (b) Give a necessary and sufficient condition on $\mathbf{M}$ such that $\mathbf{g}$ is bijective.

✓ (c) Give a necessary and sufficient condition on the individual weight matrices $\mathbf{W}_1, \dots, \mathbf{W}_K$ such that $\mathbf{g}$ is bijective.

✓ **Computer assignments** (15 pts) Construct a neural network using PyTorch. As architecture, take a fully connected neural network with three layers of weights, and five neurons in each layer. (Thus, the input and the output are also 5-dimensional.) Try out the tanh, ReLU, and linear activation functions, always the same activation everywhere in the NN. Create three sets of random weight vectors (for different plots to be made). Plot y_1, that is the first entry of the output vector, as a function of x_1 from -10.0 to 10.0, when all the other $x_i, i > 1$ are fixed to random values. **Report** three plots (corresponding to the three sets of random weights) for each of the three activation functions; in total, nine plots.
You can use the following code for the plotting:

```
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(15,10))
for i in range(3):
    ...
    for j in range(3):
        ...
        axs[i,j].plot(...,...)
        axs[i,j].set_title(...)
        axs[i,j].set_xlabel('x1')
        axs[i,j].set_ylabel('y1')
plt.tight_layout()
plt.show()
```

2 Optimization

Mathematical exercices

In the following exercices, when the exercise says ”calculate”, it means you should show the detailed derivation.

✓ 1. (3 pts) Calculate the gradient of the function $f(\mathbf{w}) = \|\mathbf{w}\|^2$.

✓ 2. (3 pts) Calculate the Hessian of the function $f(\mathbf{w}) = \|\mathbf{w}\|^2$.

✓ 3. (3 pts) Derive Newton's method for optimizing $f(\mathbf{w}) = \|\mathbf{w}\|^2$. It works incredibly well; why?

✓ 4. (2 pts) Calculate the gradient of the function $f(\mathbf{w}) = \mathbf{w}^T\mathbf{z}$ for some (constant) vector $\mathbf{z}$.

✓ 5. (2 pts) Calculate the Hessian of the function $f(\mathbf{w}) = \mathbf{w}^T\mathbf{z}$ for some (constant) vector $\mathbf{z}$.

✓ 6. (2 pts) Try to derive Newton's method for optimizing the function $f(\mathbf{w}) = \mathbf{w}^T\mathbf{z}$ for some (constant) vector $\mathbf{z}$. It clearly does not work, why?

✓ 7. (4 pts) Consider the function f_M defined as

$$f_M(\mathbf{W}) = \sum_i g(\mathbf{w}_i^T \mathbf{z}) \tag{5}$$

where $\mathbf{w}_i$ is the i-th row of $\mathbf{W}$, and $g : \mathbb{R} \to \mathbb{R}$ is some differentiable function. Calculate the partial derivative of f_M with respect the the i,j-th entry in the matrix $\mathbf{W}$, which we denote by w_{ij}. (Hint: first write out all vector operations in the objective function, and then take the partial derivate considering all the other variables constant; the tricky part is the indices, especially under a summation operator.)

✓ 8. (3 pts) Based on the preceding, express the gradient of f_M compactly using matrix/vector notation.

✓ 9. (3 pts) Calculate the *stochastic* gradient of the function

$$f_{M'}(\mathbf{W}) = \mathrm{E}\{\sum_i g(\mathbf{w}_i^T \mathbf{z})\} \tag{6}$$

for some *random* vector $\mathbf{z}$.

Computer assignments

You should do the calculations of the gradients by hand and implement them in numpy.

1. (22 pts) Consider the function

$$f(\mathbf{w}) = \exp(-w_1^2 - 2(w_2 - 1)^2) + \exp(-(w_1 - 1)^2 - 2w_2^2) \tag{7}$$

✓ (a) To begin with, plot the isocontours of the function f using the following code (use your implementation of $f(\mathbf{w})$, which takes in an array and output a scalar)

```
vec_f = np.vectorize(f, signature=”(n)->()”)
w1 = np.linspace(-2.0, 3.0, 200)
w2 = np.linspace(-2.0, 3.0, 200)
W1, W2 = np.meshgrid(w1, w2)
Fs = vec_f(np.stack((W1, W2), axis=2))
plt.figure(dpi=150)
plt.contour(W1, W2, Fs)
plt.xlabel(”w1”)
plt.ylabel(”w2”)
plt.show()
```

✓ (b) Calculate the gradient of f

✓ (c) Maximize this with the gradient method. Start with three ”random” initial points which we define here as $(0.2, 0.5)$, $(0.5, 0.2)$, $(1.0, 1.0)$ for convenience. Note: You have to try different step sizes to find a good one. (Hint: you may have to try wildly different step sizes of completely different magnitudes to find a good one). Use some stopping criterion given in the lecture material; as the threshold something like 10^{-4} might work well.

✓ (d) Plot the trajectories for the three runs corresponding to the three initial points above, using the best step size you found (doesn't have to be the the same for all initial points). More precisely, show the points $\mathbf{w}_i$ on the 2D plane where i is the iteration count, joining the points for each run using

```
plt.arrow(x, y, dx, dy,
length_includes_head=True, width=0.03)
```

for better visualization. (Note: the above command draws an arrow from (x, y) to $(x+dx, y+dy)$; replace them with the correct variables.) **Report** three such plots, one for each initial point, *overlaying* these trajectories on the contour plot plotted above.

✓ (e) Plot the objective function f as a function of iteration count, using the following code for plotting. The following code expects fs to be a dictionary containing all the intermediate function values

```
fig, axs = plt.subplots(1, 3, figsize=(15, 4))
for i, ... in
enumerate(...):
    axs[i].plot(...)
    axs[i].set_xlabel(”Number of iterations”)
    axs[i].set_ylabel(”Function value”)
plt.show()
```

Report one figure in which all the three curves are shown.

(f) **Report** a brief discussion on: What can you conclude from the above in terms of local vs. global maxima?

✓ 2. (20 pts) Consider the bivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Create a sample of $N = 1,000$ points with the following command

```
samples = sps.multivariate_normal.rvs(mean=mean,
cov=cov, size=size)
```

using the values

$$\boldsymbol{\mu} = (0,0), \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \tag{8}$$

Consider the log-likelihood as a function the inverse of the covariance matrix $\mathbf{P} = \boldsymbol{\Sigma}^{-1}$, because that is computationally simpler than using the original covariance. Its derivative is

$$\nabla \log L(\mathbf{X}) = \frac{1}{2}\left(\sum_{i=1}^{N} \mathbf{P}^{-1} - \mathbf{x}_i\mathbf{x}_i^T\right) \tag{9}$$

(a) Optimize this with the *stochastic* gradient method; start with random initial points (think about how to choose them such that the optimization is numerically stable). Try different step sizes $10^{-5}, 10^{-4}, 10^{-3}$. Use three different initial points randomly chosen. Stop each run simply after 10,000 iterations. (Don't use minibatches, for simplicity.)

(b) For all the nine runs (3 step sizes $\times$ 3 initial points), plot the objective function (for the whole data set) at each iteration. **Report** three figures, one figure for each step size, with runs from the three initial points plotted together in one plot.

(c) Give the estimation errors defined as

$$\|\mathbf{\Sigma}^{-1} - \hat{\mathbf{P}}\|^2 \tag{10}$$

for all the nine runs above, where $\hat{\mathbf{P}}$ is the final value of $\mathbf{P}$. **Report** these as raw numbers.