FTDL — Problem Set 1

Fundamentals and Basic Tools for Deep Neural Networks

Regression, Classification and Multilayer Perceptrons

Solutions

Subham Shome / Hyeon Yu

subham.shome@estudiante.uam.es / hyeon.yu@estudiante.uam.es

IPCV 2022-24

Escuela Politécnica Superior — Universidad Autónoma de Madrid

March 20, 2023

1. **State precisely what it is meant by a positive semi-definite and a positive definite matrix. State also how these types of matrices can be diagonalized.**

Ans: Let A be a symmetric matrix. The quadratic function

$$Q(z) := z^T Az \tag{1}$$

is called the quadratic form associated to A, where $z \in \mathbb{R}^n - \{0\}$, and z^T is the transpose of z. Now, the matrix A is said to be positive definite if the real number $z^T Az > 0$. Moreover, the matrix A is said to be positive semi-definite if $z^T Az \geq 0$ for every real $z \neq 0$.

Diagonalization: Since the matrix A is symmetric, it has to be orthogonally diagonalizable, which means

$$A = PDP^{-1} = PDP^T$$

for some orthogonal matrix P and diagonal matrix D. Now, replacing them in (1), we get

$$Q(z) = z^T PDP^T z = (P^T z)^T D(P^T z) \tag{2}$$

Here, we can notice that both $P^T z$ as well as $Q(z)$ can be represented by an ellipse of the general form.

Also, a positive definite and a positive semi-definite matrix can be diagonalized by Eigenvalue decomposition, Cholesky decomposition, Singular value decomposition and other ways.

2. **Prove that a positive definite matrix must be invertible.**
Hint: show, for instance, that its kernel reduces to the 0 vector.

Ans: If A is a positive definite matrix, then we have, its quadratic form $Q(z) = z^T Az > 0 \ \forall z \neq 0$. This essentially means that $Az \neq 0 \ \forall z \neq 0$, which in turn implies that A has full rank. Thus, we can say that A is invertible.

Also, if $z \in ker(A)$, then $Az = 0$, which in turn means that $z^T Az = 0$. Now, since A is positive definite, we can say that $z = 0$. Therefore, $ker(A) = 0$. Since, the kernel of the positive definite matrix reduces to zero vector, we can safely say that it is invertible.

3. **Prove that if a matrix A is semi–positive definite, then $A + \lambda$I is positive definite for any $\lambda > 0$.**

Ans: By definition of a positive semi-definite (or semi-positive definite) matrix, we know that for a matrix A, the quadratic form $Q(z) = z^T Az \geq 0$. Now, we consider $\lambda > 0$. Hence, $\lambda I > 0$, where I is the identity matrix. since, $Q(z) \geq 0$ for A, for $A + \lambda I$,

$$Q(z) = z^T (A + \lambda I) z$$

$$\Rightarrow Q(z) = z^T Az + z^T \lambda I z$$

. Here, we can say that definitely $Q(z) > 0$ since the first term $z^T Az \geq 0$ and the second term $z^T \lambda I z > 0$ (since $\lambda I > 0$). Thus, $A + \lambda I$ is a positive definite matrix.

4. **State in a precise way what is meant by the singular value decomposition (SVD) of a matrix M in the form $M = U\Sigma V^t$.**

Ans: In general, decomposition of a matrix is essentially a factorization of the matrix into multiple matrices of certain types, which can be then implemented in various applications. Singular Value Decomposition (SVD) of a matrix M is performed in the following way:

$$M = U\Sigma V^t$$

if M is a matrix consisting of only real numbers. For complex numbers, the equation changes to $M = U\Sigma V^*$, where, V^* is the complex conjugate of V and V^T is the transpose of V.

The matrices are defined in the following ways along with their shapes:

- M is the input matrix, of shape $m \times n$ (say),
- U are the left singular vectors, of shape $m \times k$,
- Σ are the diagonal eigenvalues (also known as singular values), of shape $k \times k$, and

- V are the right singular vectors, of shape $n \times k$.

To add to this, if we want to perform diagonalization of positive definite and positive semi-definite matrices using SVD, we can note the following things:

- M Assuming A is a real symmetric and positive definite matrix, meaning that all of its eigenvalues are strictly positive, we can represent its eigenvalues in a diagonal matrix Σ and have $U = V$.
- U Assuming A is a real symmetric and positive semi-definite matrix, meaning that all of its eigenvalues are positive but some can be zero, we can represent its eigenvalues in a diagonal matrix Σ. However, there is no guarantee that $U = V$. In fact, for the part of U and V corresponding to the zero eigenvalues, we can have any orthonormal decomposition of the null space of A, with sign flips allowed independently on U and V.

5. **Given a quadratic form $q(w) = w^t Q w + b \cdot w + c$, with Q a symmetric $d \times d$ matrix, $w, b, d \times 1$ vectors and c a real number, derive its gradient**

$$\nabla q(w) = 2Qw + b$$

Hint: Expand $q(w) = \sum_{p=1}^{d} \sum_{q=1}^{d} Q_{pq} w_p w_q + \sum_{1}^{d} b_p w_p + c$***, and take the partials*** $\partial q / \partial w_p$**.**

Ans: We are given the following equations:

$$q(w) = w^t Q w + b \cdot w + c \tag{3}$$

$$q(w) = \sum_{p=1}^{d} \sum_{q=1}^{d} Q_{pq} w_p w_q + \sum_{1}^{d} b_p w_p + c \tag{4}$$

It is also mentioned that Q is a symmetric $d \times d$ matrix, and w, b, d are column vectors.

We can take the partial derivative of $\partial q / \partial w_p$ in (4) to get the gradient of (3) as $\nabla q(w)$.

Now,

$$q(w) = \sum_{p=1}^{d} \sum_{q=1}^{d} Q_{pq} w_p w_q + \sum_{1}^{d} b_p w_p + c$$

Taking partial derivatives on both sides, and re-writing $w_q = w_p$ (since the index is not a parameter of dependency here),

$$\nabla q(w) = \frac{\partial q}{\partial w_p} = \sum_{p=1}^{d} \frac{\partial q}{\partial w_p}(Q_{p^2} w_p^2) + \sum_{1}^{d} \frac{\partial q}{\partial w_p}(b_p w_p) + \frac{\partial q}{\partial w_p}(c)$$

$$\Rightarrow \nabla q(w) = 2Q \sum_{p=1}^{d} w_p + \sum_{1}^{d} b_p + 0$$

$$\therefore \nabla q(w) = 2Qw + b \tag{5}$$

6. **Use this gradient expression to derive the gradient of the linear regression loss:**

$$\nabla \hat{e}(w) = \frac{1}{N} X^t X w - \frac{1}{N} X^t Y = \hat{R} w - \hat{b}$$

Ans: From the equation of linear regression loss, we get

$$\hat{e}(w) = \frac{1}{2N} \sum_{p} (w \cdot x^p - y^p)^2 \tag{6}$$

$$\Rightarrow \hat{e}(w) = \frac{1}{2N} (Xw - Y)^t (Xw - Y)$$

$$\therefore \hat{e}(w) = \frac{1}{2N} (w^t X^t X w - 2 w^t X^t Y + Y^t Y) \tag{7}$$

Now, replacing the values in (7) by values from (3), we get $Q = \frac{1}{2N} X^t X$, $b = -\frac{1}{2N} 2 X^t Y$ and $c = \frac{1}{2N} Y^t Y$.

So, by applying partial derivative (5) on (7), we get the gradient of linear regression loss as follows:

$$\nabla \hat{e}(w) = \frac{1}{2N} (2 X^t X w - 2 X^t Y)$$

$$\therefore \nabla \hat{e}(w) = \frac{1}{N} X^t X w - \frac{1}{N} X^t Y = \hat{R} w - \hat{b} \tag{8}$$

7. **Use the above to derive the minimum w^* of the ridge regression loss**

$$e_R(w) = \frac{1}{2N} \sum_{p} (y^p - w \cdot x_p^p)^2 + \frac{\alpha}{2} \|w\|^2$$

Ans: The equation given in the question is the equation of Ridge Regression Loss. To find the minimum value of ridge regression loss function (w^*), we need to take the derivative of the loss function with respect to w, set it equal to zero, and solve for w, where w is the ridge regression model's coefficient vector.

We have,

$$e_R(w) = \frac{1}{2N} \sum_{p} (y^p - w \cdot x_p^p)^2 + \frac{\alpha}{2} \|w\|^2 \tag{9}$$

we first need to expand (9) using the properties of the norm and transpose operators. We get

$$e_R(w) = \frac{1}{2N}(Y - Xw)^t(Y - Xw) + \frac{\alpha}{2}w^t w \tag{10}$$

Now, we differentiate (10) with respect to w.

$$\nabla e_R(w) = \frac{\partial e_R(w)}{\partial w} = \frac{1}{2N}(-2X^t(Y - Xw)) + \frac{\alpha}{2}(2w)$$

$$\therefore \nabla e_R(w) = \frac{1}{N}(X^t Xw - X^t Y) + \alpha w \tag{11}$$

Now, putting (11) equal to 0, and solving for w, we can find w^*.

$$\frac{1}{N}(X^t Xw - X^t Y) + \alpha w = 0$$

Taking all terms with w on one side and all other terms on the other side, we get

$$w(X^t X + \alpha N I) = X^t Y$$

where I is the identity matrix. Finally, replacing w by w^*, we get

$$w^* = (X^t X + \alpha N I)^{-1} X^t Y \tag{12}$$

One thing to notice here in (12) is that, we have the sample size N in the formula for minimization of ridge regression loss, which means that it corresponds to normalized ridge regression, which scales the penalty term by N. This version of ridge regression is also known as "L2 regularization". The normalization by N helps to keep the penalty term in balance with the data term as the sample size grows, and it leads to a unique solution for the coefficients w. It is more appropriate when the sample size is large, and when the features have different scales or variances.

Another way of expressing (12) is

$$w^* = (X^t X + \alpha I)^{-1} X^t Y$$

which is the original formulation of Ridge Regression and does not include the normalization by sample size N. This version is also known as "Tikhonov Regularization". It is more appropriate when the sample size is small, and when the features have similar scales or variances.

In both cases, the penalty-term is directly proportional to the square of the L2-norm of the coefficient of the ridge regression model ($\|w\|_2^2$).

8. **An important step when trying to build ensembles is to introduce randomness and independence while building the underlying models. How can we achieve that when working with MLPs? And how can we exploit that randomness?**

Ans: There are several ways to introduce randomness and independence when building an ensemble of MLPs. Here are some commonly used methods:

- *Random Initialization:* MLP depends on the random initial w^0; if the perceptrons have the same initial weights, the weight will constantly be updated identically causing the network to not learn anything. In addition, by using the same weights when training your MLP, there is a chance that the model will be stuck in the same local minima.
- *Dropout:* Dropout is a regularization technique that involves randomly dropping out some units in the MLP during training. This can help prevent overfitting and improve the independence of the models in the ensemble.
- *Bagging:* Bagging is a technique that involves training each MLP on a random subset of the training data. This can help introduce randomness and independence into the models.
- *Random Subspace Method:* The Random Subspace Method involves randomly selecting a subset of the input features for each MLP in the ensemble. This can help introduce independence between the models and improve their overall performance.
- *Mini-batch Training:* This technique also adds randomness to the model making sure that the MLP is never the same.
- *Stochastic weight averaging:* In stochastic weight averaging (SWA), multiple snapshots of the model's weights are saved during training. These weights are then averaged to obtain a single "averaged" model that is more robust to noise and can generalize better to unseen data.

Once we have introduced randomness and independence into our ensemble of MLPs, we can exploit it in several ways. One way is to use a voting mechanism, where each MLP in the ensemble makes a prediction, and the final prediction is based on the majority vote. Another way is to use a weighted average of the predictions, where each MLP's prediction is weighted by its performance on the validation set. We can also use a more sophisticated approach, such as stacking, where the predictions of the MLPs are used as inputs to a higher-level model, which makes the final prediction. This randomness can be exploited since each training will converge to a different optimum.

9. **We have worked out the backprop gradient equations for a regression MLP and here we want to do the same for a two class classification one that uses the sigmoid as the output function.**

(a) Work out first the gradient equation for the weights connecting the last hidden layer to the output one.

(b) What changes, if any, have to be done to the regression computations when computing the gradient equations for the other weights?

Ans:

(a) The sigmoid activation function is given by

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

Now the output of the last hidden layer of a multi-layer perceptron (MLP) is given by

$$z_2 = \sigma(w_2^T \cdot h + b_2) \tag{14}$$

where w_2 is the weight vector, h is the last hidden layer and b_2 is the bias. We can write y as an activation function of z (13) which is $y = \sigma(z)$.

Now, the loss function for two-class (binary) classification is essentially the cross-entropy loss, which is given by (15).

$$E(w_1, w_2, b_1, b_2) = -t \ln(y) - (1 - t) \cdot \ln(1 - y) \tag{15}$$

where t is the prediction (0 or 1) and y is the probability of the prediction. According to the question, we are required to find the gradient (the partial derivative) of E with respect to w_2, which is given by:

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial z_2} \times \frac{\partial z_2}{\partial w_2} \tag{16}$$

Using partial derivative of E with respect to y in (15), we get

$$\frac{\partial E}{\partial y} = -\frac{t}{y} + \frac{1 - t}{1 - y} \tag{17}$$

Also, Using partial derivative of y with respect to z_2 as $y = \sigma(z_2)$ in (13) and (14), we get

$$\frac{\partial y}{\partial z_2} = e^{-z_2}(1 + e^{-z_2})^{-2}$$

$$\Rightarrow \frac{\partial y}{\partial z_2} = \frac{1}{1 + e^{-z_2}} \left(1 - \frac{1}{1 + e^{-z_2}}\right)$$

$$\therefore \frac{\partial y}{\partial z_2} = y(1 - y) \tag{18}$$

Lastly, partially derivating z_2 with respect to w_2 in (14),

$$\frac{\partial z_2}{\partial w_2} = h \tag{19}$$

Finally, combining the equations (17), (18) and (19) in (16), we get

$$\frac{\partial E}{\partial w_2} = \left(-\frac{t}{y} + \frac{1-t}{1-y}\right) \times y(1-y) \times h$$

$$\Rightarrow \frac{\partial E}{\partial w_2} = \left(\frac{-t(1-y) + y(1-t)}{y(1-y)}\right) \times y(1-y) \times h$$

$$\Rightarrow \frac{\partial E}{\partial w_2} = (-t + ty + y - ty) \times h$$

$$\therefore \frac{\partial E}{\partial w_2} = -h(t-y) \tag{20}$$

Equation (20) gives the gradient equation for the weights connecting the last hidden layer to the output one.

(b) To compute the gradient equations for the remaining weights in the neural network, specifically those connecting the input layer to the first hidden layer, we must apply the chain rule to propagate the gradient backwards through the network. The only adjustment we need to make is in determining the derivative of the input to a hidden layer with respect to the weights. Instead of using the sigmoid function as we do for the output layer, we simply use the output of the previous layer. The gradient computation for the bias terms is similar to that of the weights, with the only difference being that the derivative of the input with respect to the bias terms is equal to 1.

10. ***Maximum likelihood estimation of regression coefficients.*** **Assume that a sample (x^p, y^p), with $x^p \in \mathbb{R}$, is obtained as**

$$y^p = a^* x^p + b^* + n^p,$$

where a^*, b^* are unknown parameters and n^p is Gaussian noise with zero mean and σ^2 variance.
Given that we assume $n^p = y^p - a^* x^p - b^*$ to have a density

$$e^{-\frac{(y^p - a x^p - b)^2}{2\sigma^2}},$$

a possible way to derive estimates $\hat{a}, \hat{b}, \hat{\sigma}$ for a^*, b^*, σ is to define the likelihood $\mathcal{L}(a, b, \sigma)$ of a parameter set (a, b, σ) as

$$\mathcal{L}(a, b, \sigma) = \prod_1^N \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^p - a x^p - b)^2}{2\sigma^2}}$$

and then obtain the scalars $\hat{a}, \hat{b}, \hat{\sigma}$ by maximizing it or, equivalently, minimizing the minus log–likelihood

$$\hat{a}, \hat{b}, \hat{\sigma} = argmin_{a,b,\sigma} \ell(a, b, \sigma) = argmin_{a,b,\sigma}(-log\mathcal{L}(a, b, \sigma)).$$

(a) Write down the minus log–likelihood cost ℓ function. Can you relate it to other regression error function you may know?

(b) Derive the corresponding estimates for the optimum values of $\hat{a}, \hat{b}$ and $\hat{\sigma}$ (i.e., those that minimize ℓ).

(c) Relate the $\hat{a}, \hat{b}$ estimates you obtain to other estimation method for regression you may know.

Ans:

(a)

$$
\begin{aligned}
-\ln \mathcal{L}(a, b, \sigma) &= -\sum_{i=1}^{N} \ln \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i^p - a x_i^p - b)^2}{2\sigma^2}} \right) \\
&= \sum_{i=1}^{N} \left(\ln \frac{1}{\sqrt{2\pi}\sigma} + \frac{(y_i^p - a x_i^p - b)^2}{2\sigma^2} \right) \\
&= \frac{N}{2} \ln(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i^p - a x_i^p - b)^2
\end{aligned}
$$

Therefore, the minus log-likelihood is:

$$
\ell(a, b, \sigma) = -\ln \mathcal{L}(a, b, \sigma) = \frac{N}{2} \ln(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i^p - a x_i^p - b)^2 \quad (21)
$$

The equation (21) corresponds to the negative log-likelihood function for the linear regression model with homoscedastic Gaussian noise. Here, The second term is a penalty term that penalizes higher values of σ, and the first term in the equation is the mean squared error (MSE) of the regression model, which is a commonly used error function for regression problems.

(b) To get the optimum values of $\hat{a}, \hat{b}$ and $\hat{\sigma}$, we need to partially differentiate (21) with respect to $\hat{a}, \hat{b}$ and $\hat{\sigma}$.

We have the following partial derivatives:

$$
\frac{\partial \ell}{\partial \hat{a}} = \frac{1}{\sigma^2} \sum_{i=1}^{N} (y_i^p - a x_i^p - b) x_i^p
$$

$$
\frac{\partial \ell}{\partial \hat{b}} = \frac{1}{\sigma^2} \sum_{i=1}^{N} (y_i^p - a x_i^p - b)
$$

$$\frac{\partial \ell}{\partial \sigma} = \frac{N}{2}\frac{1}{2\pi\sigma^2}4\pi\sigma + (-1)\frac{1}{\sigma^3}\sum_{i=1}^{N}(y_i^p - ax_i^p - b) = 0$$

Setting the previous equations to 0 and solving for $\hat{a}, \hat{b}$ and $\hat{\sigma}$, we get:

$$\hat{a} = \frac{\sum_{i=1}^{N}(x_i^p y_i^p - \hat{b}x_i^p)}{\sum_{i=1}^{N} x_i^{2p}} \tag{22}$$

$$\hat{b} = \bar{y} - a\bar{x} \tag{23}$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{N}(y_i^p - ax_i^p - b)}{N}$$

$$\therefore \hat{\sigma} = \sqrt{\frac{\sum_{i=1}^{N}(y_i^p - ax_i^p - b)}{N}} \tag{24}$$

Equations (22), (23) and (24) provide the optimal values for $\hat{a}, \hat{b}$ and $\hat{\sigma}$ that minimizes ℓ.

(c) The estimation approach introduced here closely resembles Ordinary Least Squares (OLS) regression, where the estimates of $\hat{a}$ and $\hat{b}$ are identical to those obtained through it. However, this method also considers the variability of the noise term, and the estimate for $\hat{\sigma}^2$ serves as an unbiased estimator for its variance.