EECS 182 Deep Neural Networks
Fall 2025 Anant Sahai and Gireeja Ranade

Homework 1

This homework is due on Friday, Sep 12, 2025, at 10:59PM.

1. Why Learning Rates Cannot be Too Big

To understand the role of the learning rate, it is useful to understand it in the context of the simplest possible problem first.

Suppose that we want to solve the scalar equation

$$\sigma w = y \tag{1}$$

where we know that $\sigma > 0$. We proceed with an initial condition $w_0 = 0$ by using gradient descent to minimize the squared loss

$$L(w) = (y - \sigma w)^2 \tag{2}$$

which has a derivative with respect to the parameter w of $-2\sigma(y - \sigma w)$.

Gradient descent with a learning rate of η follows the recurrence-relation or discrete-time state evolution of:

$$\begin{aligned} w_{t+1} &= w_t + 2\eta\sigma(y - \sigma w_t) \\ &= (1 - 2\eta\sigma^2)w_t + 2\eta\sigma y. \end{aligned} \tag{3}$$

(a) **For what values of learning rate $\eta > 0$ is the recurrence (3) stable?**
(HINT: Remember the role of the unit circle in determining the stability or instability of such recurrences. If you keep taking higher and higher positive integer powers of a number, what does that number has to be like for this to converge?)

(b) The previous part gives you an upper bound for the learning rate η that depends on σ beyond which we cannot safely go. **If η is below that upper bound, how fast does w_t converge to its final solution $w^* = \frac{y}{\sigma}$? i.e. if we wanted to get within a factor $(1 - \epsilon)$ of w^*, how many iterations t would we need?**
(HINT: The absolute value of the error of current w to the optimality might help.)

(c) Suppose that we now have a vector problem where we have two parameters $w[1], w[2]$. One with a large σ_ℓ and the other with a tiny σ_s. i.e. $\sigma_\ell \gg \sigma_s$ and we have the vector equation we want to solve:

$$\begin{bmatrix} \sigma_\ell & 0 \\ 0 & \sigma_s \end{bmatrix} \begin{bmatrix} w[1] \\ w[2] \end{bmatrix} = \begin{bmatrix} y[1] \\ y[2] \end{bmatrix}. \tag{4}$$

We use gradient descent with a single learning rate η to solve this problem starting from an initial condition of $\mathbf{w} = \mathbf{0}$.

For what learning rates $\eta > 0$ will we converge? Which of the two σ_i is limiting our learning rate?

(d) **For the previous problem, depending on $\eta, \sigma_\ell, \sigma_s$, which of the two dimensions is converging faster and which is converging slower?**

(e) The speed of convergence overall will be dominated by the slower of the two. **For what value of η will we get the fastest overall convergence to the solution?**

(f) Comment on what would happen if we had more parallel problems with σ_i that all were in between σ_ℓ and σ_s? **Would they influence the choice of possible learning rates or the learning rate with the fastest convergence?**

(g) Using what you know about the SVD, **how is the simple scalar and parallel scalar problem analysis above relevant to solving general least-squares problems of the form $X\mathbf{w} \approx \mathbf{y}$ using gradient descent?**

2. Stochastic Gradient Descent (when it is possible to interpolate)

This is a problem about the convergence of SGD for least-squares problems when there is actually a solution that achieves zero loss.

For simplicity, suppose that the problem we are given is

$$X\mathbf{w} = \mathbf{y} \tag{5}$$

where $X = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$ is a wide matrix with $\mathbf{x}_i$ being d-dimensional vectors and $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$ is an n-dimensional vector. Here, we assume that X has full row-rank (i.e. the $\mathbf{x}_i$ are linearly independent) and so (5) indeed has solutions. (Note that as $d > n$, there there would be infinitely many solutions.)

While we already know lots of ways of solving this problem, it is an illustrative toy example to make sure we understand why SGD works in such settings. (This material was covered in lecture but you really do need to understand it yourself so you can deal with variations you might encounter as well as internalize the kinds of manipulations required.) This problem has a Demo Notebook (**demo link**) associated with it to help you play around with things to get an even deeper set of intuitions.

In this problem, we will just initialize $\mathbf{w}_0 = \mathbf{0}$ for simplicity.

(a) Let's do some preliminaries. First, we want to change coordinates to notationally simplify our analysis of SGD.

Let $\mathbf{w}^*$ be the min-norm solution to (5).

Write out what $\mathbf{w}^*$ is explicitly with respect to X and $\mathbf{y}$ and then, change coordinates to $\mathbf{w}' = \mathbf{w} - \mathbf{w}^*$ to write the new equations as:

$$X\mathbf{w}' = \mathbf{0} \tag{6}$$

What is the new initial condition for $\mathbf{w}'_0$?

(b) Next, let's leverage SVD coordinates to further simplify the problem. **Show that there exists an orthonormal transformation V of variables $\mathbf{w}'' = V\mathbf{w}'$ so that (6) looks like**

$$[\widetilde{X} \;\; \mathbf{0}_{n\times(d-n)}]\mathbf{w}'' = \mathbf{0} \tag{7}$$

and furthermore, **show that the initial condition for $\mathbf{w}_0'$ you computed in the previous part, when viewed as $\mathbf{w}_0''$ has all zeros in the final $(d-n)$ positions.**

(c) **Argue why what you have seen in the previous parts allows us to now focus on a square system of equations:**

$$\widetilde{X}\widetilde{w} = \mathbf{0} \tag{8}$$

and furthermore show that each of the n constituent equations (corresponding to rows) of (8) can be obtained by means of coordinate changes from the same indexed equation in (5).

(d) Let's now engage with SGD itself. Here, we will just use a minibatch length of 1 and batch sampling with replacement. This means that at every iteration of SGD, we roll a fair n-sided die and choose the single equation in (5) that corresponds to the row that came up on the die. Let I_t be the iid uniform random variable on $\{1, \ldots, n\}$ that we roll after iteration t.

At the $t+1$-th iteration, we compute

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla \mathcal{L}_{I_t}(\mathbf{w}_t) \tag{9}$$

where $\mathcal{L}_i(\mathbf{w}) = (y[i] - \mathbf{x}_i^\top \mathbf{w})^2$ is the squared loss on the i-th equation and η is the step-size (learning rate).

Show that an SGD step taken in (9) for the original optimization problem matches exactly to an SGD step taken for $\widetilde{w}$ for solving (8), and that in particular these steps look like:

$$\widetilde{w}_{t+1} = \widetilde{w}_t - 2\eta \widetilde{x}_{I_t} \widetilde{x}_{I_t}^\top \widetilde{w}_t \tag{10}$$

(e) At this point, we can focus entirely on the simplified square system (8) and the stochastic evolution of the iterations described by (10).

To show convergence to zero, we need to pick a suitable stochastic Lyapunov function $\mathcal{L}(\widetilde{w})$ that is bounded below by zero and will decrease in expectation at every time step. In particular, we want to establish

$$E[\mathcal{L}(\widetilde{w}_{t+1}) | \widetilde{w}_t] < (1-\rho)\mathcal{L}(\widetilde{w}_t) \tag{11}$$

with a $1 > \rho > 0$ so that this Lyapunov function tends to decrease exponentially to zero. We will have to have a suitably small step-size/learning-rate η for this to happen, of course.

Show that (11) indeed implies that for every $\epsilon > 0$ and $\delta > 0$, there exists a $T > 0$ for which

$$P(\mathcal{L}(\widetilde{w}_T) < \epsilon) \geq 1 - \delta. \tag{12}$$

(f) One natural guess for a stochastic Lyapunov function is

$$\mathcal{L}(\widetilde{w}) = \widetilde{w}^\top \widetilde{X}^\top \widetilde{X} \widetilde{w}. \tag{13}$$

Argue why the candidate Lyapunov function $\mathcal{L}(\widetilde{w})$ in (13) is non-negative and is only equal to zero at $\widetilde{w} = \mathbf{0}$.

(g) Now, with a guessed stochastic Lyapunov function in hand, we can try to show (11). The first step will be to decompose the evolution of $\mathcal{L}(\widetilde{w})$ into three parts:

$$\mathcal{L}(\widetilde{w}_{t+1}) = \mathcal{L}(\widetilde{w}_t) + A + B \tag{14}$$

where the term A is linear in the actual stochastic update $(\widetilde{w}_{t+1} - \widetilde{w}_t)$ and the term B is quadratic in that update.

Expand out $\mathcal{L}(\widetilde{w}_t + (\widetilde{w}_{t+1} - \widetilde{w}_t))$ to give explicit forms for A and B.

(h) We are counting on the term A in (14) to give us actual contraction in expectation since this looks like a gradient-descent step. **Show:**

$$E[A|\widetilde{w}_t] \leq -c_1\eta\mathcal{L}(\widetilde{w}_t) \tag{15}$$

where the $c_1 > 0$ is a positive constant that depends on the problem.

(Hint: you are going to want to leverage the actual updates in (10) as well as the singular value structure for $\widetilde{X}$. Can the smallest singular value be zero?)

(i) We need to make sure that the "quadratic" term B in (14) cannot undo the progress made by A in expectation. **Show:**

$$E[B|\widetilde{w}_t] \leq c_2\eta^2\mathcal{L}(\widetilde{w}_t) \tag{16}$$

where $c_2 > 0$ is another positive constant that depends on the problem.

(Hint: you are going to want to leverage the actual updates in (10), the singular value structure for $\widetilde{X}$, and the fact that the rows of $\widetilde{X}$ can only be so big. You will want to leverage the largest singular value of $\widetilde{X}$ for one bounding step and then let β be the largest norm of the rows of $\widetilde{X}$ to do another bounding step.)

(j) Finally, we can put the pieces together to see that

$$E[\mathcal{L}(\widetilde{w}_{t+1})|\widetilde{w}_t] \leq (1 - c_1\eta + c_2\eta^2)\mathcal{L}(\widetilde{w}_t) \tag{17}$$

where $c_1 > 0$ and $c_2 > 0$ as well. **Show that this means that there exists a small enough η so that $1 - c_1\eta + c_2\eta^2 < 1$.**

(k) In earlier problem set, you saw how you could reinterpret ridge-regression using feature-aumentation. The earlier parts of this problem have now established that leveraging that trick, you can get SGD to converge exponentially for ridge regression. Check out Jupyter notebook in this **demo link**, and **report what you observed in terms of the convergence rate.**

One of the lessons that you will observe from the code is that the implementation details matter. If you do ridge regression and just treat it as an optimization problem, you won't just be able to use SGD and get exponential convergence with a constant step size. (You would have to adjust the step sizes to make them smaller, but this would slow down your convergence considerably.) But if you intelligently use the feature-aumentation perspective on ridge regression, you'll get exponential convergence.

This is why it is vital for people in EECS to really understand machine learning at the level of detail that we are teaching you. Because in the real world, even if you are a practicing machine learning engineer, if you are working on cutting-edge systems, you need to understand how to implement what you want to do so that it works fast. Equivalent formulations mathematically need not be equivalent from the point of view of implementation – this is one dramatic example of a case when they are not. Take EE227C and beyond if you want to understand these things more deeply.

3. Accelerating Gradient Descent with Momentum

Consider the problem of finding the minimizer of the following objective:

$$\mathcal{L}(w) = ||y - Xw||_2^2 \tag{18}$$

In an earlier problem, we proved that gradient descent (GD) algorithm can converge and derive the convergence rate. In this problem, we will add the momentum term and see how it affects to the convergence rate. The optimization procedure of gradient descent+momentum is given below:

$$\begin{aligned} w_{t+1} &= w_t - \eta z_{t+1} \\ z_{t+1} &= (1-\beta) z_t + \beta g_t, \end{aligned} \tag{19}$$

where $g_t = \nabla \mathcal{L}(w_t)$, η is learning rate and β defines how much averaging we want for the gradient. Note that when $\beta = 1$, the above procedure is just the original gradient descent.

Let's investigate the effect of this change. We'll see that this modification can actually 'accelerate' the convergence by allowing larger learning rates.

(a) Recall that the gradient descent update of (18) is

$$w_{t+1} = \left(I - 2\eta(X^T X)\right) w_t + 2\eta X^T y \tag{20}$$

and the minimizer is

$$w^* = (X^T X)^{-1} X^T y \tag{21}$$

The geometric convergence rate (in the sense of what base is there for convergence as rate^t) of this procedure is

$$\text{rate} = \max_i |1 - 2\eta\sigma_i^2| \tag{22}$$

You already saw f we choose the learning rate that maximizes (22), the optimal learning rate, η^* is

$$\eta^* = \frac{1}{\sigma_{\text{min}}^2 + \sigma_{\text{max}}^2}, \tag{23}$$

where σ_{max} and σ_{min} are the maximum and minimum singular value of the matrix X. The corresponding optimal convergence rate is

$$\text{optimal rate} = \frac{(\sigma_{\text{max}}/\sigma_{\text{min}})^2 - 1}{(\sigma_{\text{max}}/\sigma_{\text{min}})^2 + 1} \tag{24}$$

Therefore, how fast ordinary gradient descent converges is determined by the ratio between the maximum singular value and the minimum singular value as above.

Now, let's consider using momentum to smooth the gradients before taking a step in (19).

$$\begin{aligned} w_{t+1} &= w_t - \eta z_{t+1} \\ z_{t+1} &= (1-\beta) z_t + \beta(2X^T X w_t - 2X^T y) \end{aligned} \tag{25}$$

We can use the SVD of the matrix $X = U\Sigma V^T$, where $\Sigma = \text{diag}(\sigma_{\text{max}}, \sigma_2, \ldots, \sigma_{\text{min}})$ with the same (potentially rectangular) shape as X. This allows us to reparameterize the parameters w_t and averaged gradients z_t as below:

$$x_t = V^T(w_t - w^*)$$

$$a_t = V^T z_t. \tag{26}$$

Please rewrite (25) with the reparameterized variables, $x_t[i]$ and $a_t[i]$. ($x_t[i]$ and $a_t[i]$ are i-th components of x_t and a_t respectively.)

(b) Notice that the above 2×2 vector/matrix recurrence has no external input. We can derive the 2×2 system matrix R_i from above such that

$$\begin{bmatrix} a_{t+1}[i] \\ x_{t+1}[i] \end{bmatrix} = R_i \begin{bmatrix} a_t[i] \\ x_t[i] \end{bmatrix} \tag{27}$$

Derive R_i.

(c) **Use the computer to symbolically find the eigenvalues of the matrix R_i.**
When are they purely real? When are they repeated and purely real? When are they complex?

(d) **For the case when they are repeated, what is the condition on η, β, σ_i that keeps them stable (strictly inside the unit circle)? What is the highest learning rate η as a function of β and σ_i that results in repeated eigenvalues?**

(e) **For the case when the eigenvalues are real, what is the condition on η, β, σ_i that keeps them stable (strictly inside the unit circle)? What is the range of the learning rate? Express with β, σ_i**

(f) **For the case when the eigenvalues are complex, what is the condition on η, β, σ_i that keeps them stable (strictly inside the unit circle)? What is the highest learning rate η as a function of β and σ_i that results in complex eigenvalues?**

(g) (This question might take more time than others) Now, apply what you have learned to the following problem. Assume that $\beta = 0.1$ and we have a problem with two singular values $\sigma^2_{\text{max}} = 5$ and $\sigma^2_{\text{min}} = 0.05$. **What learning rate η should we choose to get the fastest convergence for gradient descent with momentum? Compare how many iterations it will take to get within 99.9% of the optimal solution (starting at 0) using this learning rate and momentum with what it would take using ordinary gradient descent.**

(h) The 2 questions below are based on the Jupyter Notebook given in the notebook. Please open the corresponding notebook and follow the instructions to answer the following questions. You don't need to submit the `ipynb` file.

How does σ_i (the eigenvalues) influence the gradients and parameters updates?

(i) Question: Comparing gradient descent and gradient descent with momentum, **which one converges faster for this task? Why?**

4. Optimizers

Algorithm 1 SGD with Momentum

Given $\eta = 0.001, \beta_1 = 0.9$
Initialize:
  time step $t \leftarrow 0$
  parameter $\theta_{t=0} \in \mathbb{R}^n$
Repeat
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla f_t(\theta_{t-1})$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $\theta_t \leftarrow \theta_{t-1} - \eta m_t$
Until the stopping condition is met

Algorithm 2 Adam Optimizer (without bias correction)

Given $\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$
Initialize time step $t \leftarrow 0$, parameter $\theta_{t=0} \in \mathbb{R}^n$, $m_{t=0} \leftarrow 0$, $v_{t=0} \leftarrow 0$
Repeat
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla f_t(\theta_{t-1})$
  $m_t \leftarrow$ ____**(A)**____
  $v_t \leftarrow$ ____**(B)**____
  $\theta_t \leftarrow \theta_{t-1} - \eta \cdot \frac{m_t}{\sqrt{v_t}}$
Until the stopping condition is met

(a) Complete part **(A)** and **(B)** in the pseudocode of Adam.

(b) This question asks you to establish the relationship between

- **L2 regularization** for vector-valued weights θ refers to adding a squared Euclidean norm of the weights to the loss function itself:

$$f_t^{reg} = f_t(\theta) + \frac{\lambda}{2}||\theta||_2^2$$

- **Weight decay** refers to explicitly introducing a scalar γ in the weight updates assuming loss f:

$$\theta_{t+1} = (1 - \gamma)\theta_t - \eta \nabla f(\theta_t)$$

  where $\gamma = 0$ would correspond to regular SGD since it has no weight-decay.

Show that SGD with weight decay using the original loss $f_t(\theta)$ is equivalent to regular SGD on the L2-regularized loss $f_t^{reg}(\theta)$ when γ is chosen correctly, and **find such a γ in terms of λ and η.**

5. Regularization and Instance Noise

Say we have m labeled data points $(\mathbf{x}_i, y_i)_{i=1}^m$, where each $\mathbf{x}_i \in \mathbb{R}^n$ and each $y_i \in \mathbb{R}$. We perform data augmentation by adding some noise to each vector every time we use it in SGD. This means for all points i, we have a true input $\mathbf{x}_i$ and add noise $\mathbf{N}_i$ to get the effective random input seen by SGD:

$$\check{\mathbf{X}}_i = \mathbf{x}_i + \mathbf{N}_i$$

The i.i.d. random noise vectors $\mathbf{N}_i$ are distributed as $\mathbf{N}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_n)$.

We can conceptually arrange these noise-augmented data points into a random matrix $\check{X} \in \mathbb{R}^{m \times n}$, where row $\check{\mathbf{X}}_i^\top$ represents one augmented datapoint. Similarly we arrange the labels y_i into a vector $\mathbf{y}$.

$$\check{X} = \begin{bmatrix} \check{\mathbf{X}}_1^\top \\ \check{\mathbf{X}}_2^\top \\ \dots \\ \check{\mathbf{X}}_m^\top \end{bmatrix}, \text{ where } \check{\mathbf{X}}_i \in \mathbb{R}^n, \text{ and } \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} \in \mathbb{R}^m$$

One way of thinking about what SGD might do is to consider learning weights that minimize the *expected* least squares objective for the **noisy** data matrix:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \mathbb{E}[\|\check{X}\mathbf{w} - \mathbf{y}\|^2] \tag{28}$$

(a) **Show that this problem (28) is equivalent to a regularized least squares problem:**

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{m}\|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2 \tag{29}$$

You will need to determine the value of λ.

Hint: write the squared norm of a vector as an inner product, expand, and apply linearity of expectation.

Now consider a simplified example where we only have a single scalar datapoint $x \in \mathbb{R}$ and its corresponding label $y \in \mathbb{R}$. We are going to analyze this in the context of gradient descent. For the t-th step of gradient descent, we use a noisy datapoint $\check{X}_t = x + N_t$ which is generated by adding different random noise values $N_t \sim \mathcal{N}(0, \sigma^2)$ to our underlying data point x. The noise values for each iteration of gradient descent are i.i.d. We want to learn a weight w such that the squared-loss function $\mathcal{L}(w) = \frac{1}{2}(\check{X}w - y)^2$ is minimized. We initialize our weight to be $w_0 = 0$.

(b) Let w_t be the weight learned after the t-th iteration of gradient descent with data augmentation. **Write the gradient descent recurrence relation between $\mathbb{E}[w_{t+1}]$ and $\mathbb{E}[w_t]$ in terms of x, σ^2, y, and learning rate η.**

(c) **For what values of learning rate η do we expect the expectation of the learned weight to converge using gradient descent?**

(d) Assuming that we are in the range of η for which gradient-descent converges, **what would we expect $\mathbb{E}[w_t]$ to converge to as $t \to \infty$? How does this differ from the optimal value of w if there were no noise being used to augment the data?**

(HINT: You can also use this to help check your work for part (a).)

6. General Case Tikhonov Regularization

Consider the optimization problem:

$$\min_{\mathbf{x}} ||W_1(A\mathbf{x} - \mathbf{b})||_2^2 + ||W_2(\mathbf{x} - \mathbf{c})||_2^2$$

Where W_1, A, and W_2 are matrices and $\mathbf{x}$, $\mathbf{b}$ and $\mathbf{c}$ are vectors. W_1 can be viewed as a generic weighting of the residuals and W_2 along with c can be viewed as a generic weighting of the parameters.

(a) **Solve this optimization problem manually** by expanding it out as matrix-vector products, setting the gradient to $\mathbf{0}$, and solving for $\mathbf{x}$.

(b) **Construct an appropriate matrix $\mathbf{C}$ and vector $\mathbf{d}$ that allows you to rewrite this problem as**

$$\min_{x} \|C\mathbf{x} - \mathbf{d}\|^2$$

and use the OLS solution ($\mathbf{x}^* = (C^TC)^{-1}C^T\mathbf{d}$) to solve. Confirm your answer is in agreement with the previous part.

(c) **Choose a W_1, W_2, and c** such that this reduces to the simple case of ridge regression that you've seen in the previous problem, $\mathbf{x}^* = (A^TA + \lambda I)^{-1}A^T\mathbf{b}$.

7. An Alternate MAP Interpretation of Ridge Regression

Consider the Ridge Regression estimator,

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|^2$$

We know this is solved by

$$\hat{\mathbf{w}} = (X^TX + \lambda I)^{-1}X^T\mathbf{y} \tag{30}$$

An alternate form of the Ridge Regression solution (often called the Kernel Ridge form) is given by

$$\hat{\mathbf{w}} = X^T(XX^T + \lambda I)^{-1}\mathbf{y}. \tag{31}$$

We know that Ridge Regression can be viewed as finding the MAP estimate when we apply a prior on the (now viewed as random parameters) $\mathbf{W}$. In particular, we can think of the prior for $\mathbf{W}$ as being $\mathcal{N}(\mathbf{0}, I)$ and view the random Y as being generated using $Y = \mathbf{x}^T\mathbf{W} + \sqrt{\lambda}N$ where the noise N is distributed iid (across training samples) as $\mathcal{N}(0, 1)$. At the vector level, we have $\mathbf{Y} = X\mathbf{W} + \sqrt{\lambda}\mathbf{N}$, and then we know that when we try to maximize the log likelihood we end up minimizing

$$\underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{\lambda}\|X\mathbf{w} - y\|_2^2 + \|\mathbf{w}\|^2 = \underset{\mathbf{w}}{\operatorname{argmin}} \|X\mathbf{w} - y\|_2^2 + \lambda\|\mathbf{w}\|^2.$$

The underlying probability space is that defined by the d iid standard normals that define the $\mathbf{W}$ and the n iid standard normals that give the n different N_i on the training points. Note that the X matrix whose rows consist of the n different inputs for the n different training points are not random.

Based on what we know about joint normality, it is clear that the random Gaussian vectors $\mathbf{W}$ and $\mathbf{Y}$ are jointly normal. **Use the following facts to show that the two forms of solution are identical.**

- (30) is the MAP estimate for $\mathbf{W}$ given an observation $\mathbf{Y} = \mathbf{y}$ (We showed this in HW1 last week, and in discussion section)
- For jointly normal random variables, when you condition one set of variables on the values for the others, the resulting conditional distribution is still normal.
- A normal random variable has its density maximized at its mean.
- For jointly normal random vectors that are zero mean, the formula for conditional expectation is

$$E[\mathbf{W}|\mathbf{Y} = \mathbf{y}] = \Sigma_{WY}\Sigma_{YY}^{-1}\mathbf{y} \tag{32}$$

  where the Σ_{YY} is the covariance $E[\mathbf{Y}\mathbf{Y}^T]$ of $\mathbf{Y}$ and $\Sigma_{WY} = E[\mathbf{W}\mathbf{Y}^T]$ is the appropriate cross-covariance of $\mathbf{W}$ and $\mathbf{Y}$.

8. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework?**

Contributors:

- Anant Sahai.
- Sheng Shen.
- Gireeja Ranade.
- Yaodong Yu.
- Suhong Moon.
- Gabriel Goh.
- Peter Wang.
- Yuxi Liu.
- Kevin Li.
- Saagar Sanghavi.

