CS189Hw2

Trustin Nguyen

October 31, 2024

Multivariate Gaussians: A review

Multivariate Gaussian distributions crop up everywhere in machine learning, from priors on model parameters to assumptions on noise distributions. Being able to manipulate multivariate Gaussians also becomes important for analyzing correlations in data and preprocessing it for better regression and classification. We want to make sure to first cover the MVG fundamentals here.

Note that the probability density function of a non-degenerate (i.e. the covariance matrix is positive definite and, thus, invertible) multivariate Gaussian RV with mean vector, $\mu \in \mathbb{R}^2$, and covariance matrix, $\Sigma \in \mathbb{R}^{2\times 2}$, is:

$$f(z) = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} \exp\left(-\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)\right)$$

(a) Consider a two dimensional, zero mean random variable $Z = \begin{bmatrix} Z_1 & Z_2 \end{bmatrix}^T \in \mathbb{R}^2$. In order for the random variable to be jointly Gaussian, a necessary and sufficient condition which we call the *first characterization* is that

- Z_1 and Z_2 are each marginally Gaussian, and
- $Z_1 \mid Z_2 = z$ is Gaussian, and $Z_2 \mid Z_1 = z$ is Gaussian.

A *second characterization* of a jointly Gaussian zero mean RV $Z \in \mathbb{R}^2$ is that it can be written as $Z = AX$, where $X \in \mathbb{R}^2$ is a collection of i.i.d. standard normal RVs and $A \in \mathbb{R}^{2\times 2}$ is a matrix.

Let X_1 and X_2 be i.i.d. standard normal RVs. Let U denote a binary random variable uniformly that is equal to 1 with probability $\frac{1}{2}$ and -1 with probability $\frac{1}{2}$, independent of everything else.

For each of the below subproblems, complete the following *two* steps: (1) Using one of the characterizations given above, determine whether the RVs are jointly Gaussian. If using the second characterization, clearly specify the A matrix. (2) Calculate the covariance matrix of Z (regardless of whether the RVs are jointly Gaussian or not).

(i.) $Z_1 = X_1$ and $Z_2 = X_2$

Answer. I will be using the first characterization. To show that Z_1 and Z_2 are marginally Gaussian, we need to show that:

$$p_{Z_2}(z) = \int_{-\infty}^{\infty} p(Z_1 = z', Z_2 = z)\, dz'$$

is Gaussian. Since the RVs are independent:

$$\begin{aligned} p_{Z_2}(z) &= \int_{-\infty}^{\infty} p(Z_1 = z')p(Z_2 = z)\, dz' \\ &= p(Z_2 = z)\int_{-\infty}^{\infty} p(Z_1 = z')\, dz' \\ &= p(Z_2 = z) \end{aligned}$$

so p_{Z_2} is Gaussian. The other way is symmetric.

To see that $Z_1 \mid Z_2 = z$ is Gaussian, we have:

$$p(Z_1 = z_1 \mid Z_2 = z) = \frac{p(Z_1 = z_1, Z_2 = z)}{p(Z_2 = z)}$$

Since they are independent:

$$p(Z_1 = z_1 \mid Z_2 = z) = \frac{p(Z_1 = z_1)p(Z_2 = z)}{p(Z_2 = z)} = p(Z_1 = z_1)$$

So the distribution of $Z_1 \mid Z_2 = z$ is the same as that of Z_1, and is therefore Gaussian. The other case is symmetric.

The covariance matrix of Z is the variance of Z_1 along the diagonal because the distributions are independent.

(ii.) $Z_1 = X_1$ and $Z_2 = X_1 + 2X_2$. If using the first characterization, assume that you already know $(Z_1 \mid Z_2 = z)$ is Gaussian.

Answer. It is jointly Gaussian because

$$\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

by the second classification. From lecture, the covariance matrix is given by AA^T.

(iii.) $Z_1 = X_1$ and $Z_2 = -X_1$.

Answer. Yes these are joint Gaussian because $Z = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$. Same covariance matrix in the previous question: AA^T.

(iv.) $Z_1 = X_1$ and $Z_2 = UX_1$.

Answer. We will use the first characterization here. To show marginal Gaussian:

$$p_{X_1}(z) = p(X_1 = z) = \int_{-\infty}^{\infty} p(X_1 = z, UX_1 = z')\, dz'$$

must be Gaussian. We know that it has non-zero values when $z' = \pm z$:

$$p_{X_1}(z) = \sum_{z'=\pm z} p(X_1 = z, UX_1 = z')$$

We can expand:

$$p(X_1 = z, UX_1 = z) + p(X_1 = z, UX_1 = -z) = p(UX_1 = z)p(U = 1) + p(X_1 = z)p(U = -1)$$

we know that:

$$p(U = \pm 1) = 1/2$$

So this turns out to just $p(X_1 = z)$. So $p_{X_1}(z)$ is gaussian.

Now for the other marginal:

$$p_{UX_1}(z) = p(UX_1 = z) = \int_{-\infty}^{\infty} p(X_1 = z', UX_1 = z)\, dz'$$

Again, nonzero when $X_1 = \pm z$:

$$\sum_{z'=\pm z} p(X_1 = z', UX_1 = z) = p(X_1 = z)p(U = 1) + p(X_1 = -z)p(U = -1)$$

which is

$$\frac{1}{2}(p(X_1 = z) + p(X_1 = -z))$$

Since X_1 is standard normal, this becomes $p(X_1 = z)$ which is normal.

Now on to the conditional probabilities. We have

$$p(X_1 = x \mid UX_1 = x') = \frac{p(X_1 = x, UX_1 = x')}{p(UX_1 = x')}$$

expand:

$$\frac{p(X_1 = x, UX_1 = x') + p(X_1 = x, UX_1 = -x')}{p(X_1 = x')(p(U = -1) + p(U = 1))}$$

which is

$$\frac{p(X_1 = x)p(X_1 = x')}{p(X_1 = x')} = p(X_1 = x)$$

which is gaussian.

For the other conditional:

$$p(UX_1 = x \mid X_1 = x') = \frac{p(UX_1 = x, X_1 = x')}{p(X_1 = x')}$$

expand:

$$\frac{p(X_1 = x')(p(UX_1 = -x) + p(UX_1 = x))}{p(X_1 = x')} = p(UX_1 = x) + p(UX_1 = -x)$$

so this is gaussian also.

Now for the covariance,

$$Cov(UX, X) = \mathbb{E}[(UX - \mathbb{E}(UX))(X)]$$

or

$$Cov(UX, X) = \mathbb{E}[UX^2 - X] = -\mathbb{E}[X] = 0$$

So the matrix is

$$\begin{bmatrix} 1 & 0 \\ 0 & Var(UX) \end{bmatrix}$$

and

$$Var(UX) = \mathbb{E}[U^2X^2] - \mathbb{E}[UX]^2 = \mathbb{E}[U^2X^2] = \mathbb{E}[U^2]\mathbb{E}[X^2] = \mathbb{E}[X^2] = 1$$

(b) Show that two Gaussian random variables can be uncorrelated, but not independent. On the other hand, show that two uncorrelated, jointly Gaussian RVs are independent.

Answer. In part (d) of last example, we computed a covariance matrix of $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ but UX, X are not independent. So we see that non correlation does not mean independence.

If two Gaussians are uncorrelated, then from the joint pdf:

$$p_{Z_1,Z_2}(z) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left\{-\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)\right\}$$

Σ is a diagonal matrix, so the exponent part factors into

$$Var(z_1)\cdot(z_1-\mu_1)^2 + Var(z_2)\cdot(z_2-\mu_2)^2$$

The determinant in the denominator also factors into: $Var(z_1)Var(z_2)$. So the joint pdf factors into their marginal distributions:

$$p_{Z_1,Z_2}(z) = \frac{1}{\sqrt{2\pi}\sigma_1}\exp\left\{-\frac{1}{2}\left(\frac{z_1-\mu_1}{\sigma_1}\right)^2\right\}\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left\{-\frac{1}{2}\left(\frac{z_2-\mu_2}{\sigma_2}\right)^2\right\}$$

(c) With the setup in (a), let $Z = VX$, where $V \in \mathbb{R}^{2\times 2}$, and $Z, X \in \mathbb{R}^2$. What is the covariance matrix Σ_Z? If X is not a multivariate Gaussian but has the identity matrix $I \in \mathbb{R}^{2\times 2}$ as its covariance matrix, is your computed Σ_Z still the covariance of Z?

Answer. Recall from the previous homework that if $Z = AX + b$, then

$$Var(Z) = A\Sigma A^T$$

where Σ is the covariance matrix of X. Then the covariance matrix of Z is

$$V\Sigma V^T$$

Here, Σ is the identity because X is standard normal. This also implies that if X has an identity matrix as the covariance matrix, the computed covariance does not change.

(d) Given a jointly Gaussian zero mean RV $Z = \begin{bmatrix}Z_1 & Z_2\end{bmatrix}^T \in \mathbb{R}^2$ with covariance matrix $\Sigma_Z = \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{12} & \Sigma_{22}\end{bmatrix}$, derive the conditional distribution of $(Z_1 \mid Z_2 = z)$.

Hint: The following identity may be useful

$$\begin{bmatrix}a & b\\ b & c\end{bmatrix}^{-1} = \begin{bmatrix}1 & 0\\ -\frac{b}{c} & 1\end{bmatrix}\begin{bmatrix}\left(a-\frac{b^2}{c}\right)^{-1} & 0\\ 0 & \frac{1}{c}\end{bmatrix}\begin{bmatrix}1 & -\frac{b}{c}\\ 0 & 1\end{bmatrix}$$

Answer. The joint pdf is:

$$\frac{1}{\sqrt{2\pi|\Sigma|}}\exp\left\{-\frac{1}{2}(Z-\mu)^T\Sigma^{-1}(Z-\mu)\right\}$$

And from Bayes's rule:

$$p_{Z_1}(Z_1 = z_1, Z_2 = z) = p(Z_1 \mid Z_2 = z)p(Z_2 = z)$$

What we want to solve for is $p(Z_1 \mid Z_2 = z)$ on the RHS. So the LHS is:

$$\text{LHS} = \frac{1}{\sqrt{2\pi\left|\begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{12} & \Sigma_{22}\end{bmatrix}\right|}}\exp\left\{-\frac{1}{2}\begin{bmatrix}z_1 & z\end{bmatrix}\begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{12} & \Sigma_{22}\end{bmatrix}^{-1}\begin{bmatrix}z_1\\ z\end{bmatrix}\right\}$$

$$= \frac{1}{\sqrt{2\pi(\Sigma_{11}\Sigma_{22}-\Sigma_{12}^2)}}\exp\left\{-\frac{1}{2}\begin{bmatrix}z_1 & z\end{bmatrix}\begin{bmatrix}1 & 0\\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1\end{bmatrix}\begin{bmatrix}\frac{\Sigma_{22}}{\Sigma_{11}\Sigma_{22}-\Sigma_{12}^2} & 0\\ 0 & \frac{1}{\Sigma_{22}}\end{bmatrix}\begin{bmatrix}1 & -\frac{\Sigma_{12}}{\Sigma_{22}}\\ 0 & 1\end{bmatrix}\begin{bmatrix}z_1\\ z\end{bmatrix}\right\}$$

Recall that we still had the $p(Z_2 = z)$ term on the RHS which is:

$$p(Z_2 = z) = \frac{1}{\sqrt{2\pi\Sigma_{22}}} \exp\left\{-\frac{1}{2}\begin{bmatrix} z_1 & z \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix}\begin{bmatrix} z_1 \\ z \end{bmatrix}\right\}$$

So when we take $p_{Z_1}(Z_1 = z_1, Z_2 = z)/p(Z_2 = z)$, we would end up subtracting the exponents in the exp() term:

$$\begin{bmatrix} z_1 & z \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1 \end{bmatrix}\begin{bmatrix} \frac{\Sigma_{22}}{\Sigma_{11}\Sigma_{22}-\Sigma_{12}^2} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix}\begin{bmatrix} 1 & -\frac{\Sigma_{12}}{\Sigma_{22}} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} z_1 \\ z \end{bmatrix} - \begin{bmatrix} z_1 & z \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix}\begin{bmatrix} z_1 \\ z \end{bmatrix}$$

and notice that:

$$\begin{bmatrix} z_1 & z \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix}\begin{bmatrix} z_1 \\ z \end{bmatrix} = \begin{bmatrix} z_1 & z \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix}\begin{bmatrix} 1 & -\frac{\Sigma_{12}}{\Sigma_{22}} \\ 0 & 1 \end{bmatrix}$$

So subtracting yields:

$$\begin{bmatrix} z_1 & z \end{bmatrix}\begin{bmatrix} 1 & 0 \\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1 \end{bmatrix}\begin{bmatrix} \frac{\Sigma_{22}}{\Sigma_{11}\Sigma_{22}-\Sigma_{12}^2} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & -\frac{\Sigma_{12}}{\Sigma_{22}} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} z_1 \\ z \end{bmatrix}$$

We can simplify the matrix in the middle down to:

$$\begin{bmatrix} \frac{\Sigma_{22}}{|\Sigma|} & -\frac{\Sigma_{12}}{|\Sigma|} \\ -\frac{\Sigma_{12}}{|\Sigma|} & \frac{\Sigma_{12}^2}{\Sigma_{22}|\Sigma|} \end{bmatrix}$$

The rest is just algebraic manipulation.

Projections and Linear Regression

We are given $X \in \mathbb{R}^{n \times d}$ where $n > d$ and $\mathrm{rank}(X) = d$. We are also given a vector $y \in \mathbb{R}^n$. Define the orthogonal projection of y onto $\mathrm{range}(X)$ as $P_{\mathrm{range}(X)}(y)$.

Background on orthogonal projections: For any finite-dimensional subspace W (here, $\mathrm{range}(X)$) of a vector space V (here, $\mathbb{R}^n$), any vector $v \in V$ can be decomposed as

$$v = w + u,\ w \in W,\ u \in W^\perp$$

where $W^\perp$ is the orthogonal complement of W. Furthermore, this decomposition is unique: if $v = w' + u'$ where $w' \in W, u' \in W^\perp$, then $w' = w$ and $u' = u$. These two facts allow us to define P_W, the orthogonal projection operator onto W. Given a vector v with decomposition $v = w + u$, we define

$$P_W(v) = w.$$

It can also be shown using these two facts that P_W is linear.

(a) Prove that $P_{\mathrm{range}(X)}(y) = \mathrm{argmin}_{w \in \mathrm{range}(X)} \|y - w\|_2^2$

Answer. We know that $y = x + w$ for $x \in \mathrm{range}(X)$ and $w \in X^\perp$. So the equation becomes:

$$\operatorname*{argmin}_{x' \in \mathrm{range}(X)} \|x + w - x'\|_2^2$$

The norm is:

$$(x + w - x') \cdot (x + w - x')$$

a dot product:

$$\begin{aligned}((x - x') + w) \cdot ((x - x') + w) &= (x - x') \cdot (x - x') + 2(w \cdot (x - x')) + w \cdot w \\ &= \|x - x'\| + \|w\|\end{aligned}$$

We want the x' which minimizes this. So $x' = x = P_{\mathrm{range}(X)}(y)$.

(b) An orthogonal projections is a linear transformation. That is, $P_{\mathrm{range}(X)}(y) = Py$ for some projection matrix P

Specifically, given $1 \leqslant d \leqslant n$, a matrix $P \in \mathbb{R}^{n \times n}$ is said to be a rank-d orthogonal projections matrix if

- $\mathrm{rank}(P) = d$
- $P = P^T$
- $P^2 = P$

Prove that P is a rank-d projection matrix if an only if there exists a $U \in \mathbb{R}^{n \times d}$ such that $P = UU^T$ and $U^TU = I$.

Answer. ($\rightarrow$) From the three statements above, $P = P^TP$, which means that P is symmetric, there is an eigenvalue decomposition. The eigenvalues are 1 for elements in the basis of X and 0 in the subspace W/X. So:

$$P = U'IU'^T$$

but U' has d non-zero vectors in its column space. So $P = UU^T$, $U^TU = I$ because U consists of orthonormal vectors.

($\leftarrow$) Since $U^TU = I$, U is full rank. We know this because the dot products not along the diagonal are 0, so there cant be any linear dependence. This means that the rank of P is also d because:

$$U^TU = I \implies UU^TU = I \implies PU = U$$

and therefore, the rank of P cannot be any less. It also cannot be any more because $P = UU^T$ and is restricted by $rank(U)$. The other two properties of an orthogonal projection matrix can be shown by algebraic manipulation.

(c) The Singular Value Decomposition theorem states that we can write any matrix X as

$$X = \sum_{i=1}^{\min(n,d)} \sigma_i u_i v_i^T = \sum_{i:\sigma_i>0} \sigma_i u_i v_i^T$$

where $\sigma_i \geqslant 0$, and $\{u_i\}_{i=1}^n$ and $\{v_i\}_{i=1}^d$ are orthonormal bases for $\mathbb{R}^n$ and $\mathbb{R}^d$ respectively. Some of the singular values σ_i may equal 0, indicating that the associated left and right singular vectors u_i and v_i do not contribute to the sum, but sometimes it is still convenient to include them in the SVD so we have complete orthonormal bases for $\mathbb{R}^n$ and $\mathbb{R}^d$ to work with. Show that

(i) $\{u_i : \sigma_i > 0\}$ is an orthonormal basis for the column space of X.

Answer. We see that the action of X is equivalent to the action of U on the image of V^T. Since V is orthonormal, $V^TV = I$, V^T has is full rank. So the image of V^T is just the domain of X. Now by

$$X = \sum_{i:\sigma_i>0} \sigma_i u_i v_i^T$$

for some vector w:

$$Xw = \sum_{i:\sigma_i>0} \sigma_i u_i v_i^T w = \sum_{i:\sigma_i\geqslant 0} \sigma_i w' u_i$$

So the column space of X has the $u_i : \sigma_i > 0$ as a basis.

(ii) Similarly, $\{v_i : \sigma_i > 0\}$ is an orthonormal basis for the row space of X.

Answer. Taking the transpose, this becomes:

$$X^T = \sum_{i:\sigma_i>0} \sigma_i v_i u_i^T$$

the argument is the same.

(d) Let $X \in \mathbb{R}^{n\times d}$ such that $rank(X) = d$. Prove that $X(X^TX)^{-1}X^T$ is a rank-d orthogonal projection matrix.

Answer. Using the SVD of X, we see that

$$X^TX = \sum_{j:\sigma_j>0} \sigma_j v_j u_j^T \sum_{i:\sigma_i>0} \sigma_i u_i v_i^T$$

for the cross terms, where $i \neq j$, because the columns of U are orthonormal, the dot product goes to 0. So this can be simplified to:

$$(X^TX) = \sum_{i:\sigma_i>0} \sigma_i^2 v_i v_i^T = V\Sigma V^T$$

The inverse is:

$$V\Sigma^{-1}V^T$$

Now

$$X(X^TX)^{-1}X^T = XV\Sigma^{-1}V^TX^T = (XV)\Sigma^{-1}(XV)^T = \Sigma U\Sigma^{-1}U^T = UU^T$$

(e) Prove that $X(X^TX)^{-1}X^T$ is a projection onto range(X).

Answer. Suppose $z = Xw$. Then

$$X(X^TX)^{-1}X^T(Xw) = Xw = z$$

so any element in $\mathrm{range}(X)$ is fixed by the above projection matrix. Formally, $\mathrm{range}(X) \subseteq S$ where S is the subspace $X(X^TX)^{-1}X^T$ projects on. We know that the dimension of $\mathrm{range}(X)$ is d. So $S = \mathrm{range}(X)$.

(f) Show that $w^* = (X^TX)^{-1}X^Ty$ is the solution to the optimization problem

$$\underset{w}{\mathrm{argmin}} \|y - Xw\|_2^2$$

Answer. We know that $z = Xw \in \mathrm{range}(X)$. So by the first problem (a),

$$P_{\mathrm{range}(X)}(y) = \underset{z \in \mathrm{range}(X)}{\mathrm{argmin}} \|y - z\|_2^2$$

So $z = X(X^TX)^{-1}X^Ty$. This means $w = (X^TX)^{-1}X^Ty$ minimizes the norm.

Some MLEs

For this question, assume you observe n (data point, label) pairs $(x_i, y_i)_{i=1}^n$, with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for all $i = 1, \ldots, n$. We denote X as the data matrix containing all the data points and y as the label vector containing all the labels:

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \in \mathbb{R}^{n \times d}, \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

(a) Ignoring y for now, suppose we model the data points as coming from a d-dimensional Gaussian with diagonal covariance:

$$\forall i = 1, \ldots, n,\ x_i \sim N(\mu, \Sigma);\ \Sigma = \begin{bmatrix} \sigma_1^2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_d^2 \end{bmatrix}$$

If we consider $\mu \in \mathbb{R}^d$ and $(\sigma_1^2, \ldots, \sigma_d^2)$, where each $\sigma_i^2 > 0$, to be unknown, the parameter space here is $2d$-dimensional. When we refer to Σ as a parameter, we are referring to the d-tuple $(\sigma_1^2, \ldots, \sigma_d^2)$, but inside a linear algebraic expression, Σ denotes the diagonal matrix $\mathtt{diag}(\sigma_1^2, \ldots, \sigma_d^2)$.

Solve the following problems:

(i) Prove that log-likelihood $l(\mu, \Sigma) = \log p(X \mid \mu, \Sigma)$ is equal to

$$-\frac{n}{2}\left(d \log 2\pi - \sum_{j=1}^{d} \log \frac{1}{\sigma_j^2}\right) - \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu)$$

Answer. The log likelihood is:

$$\begin{aligned} l(\mu, \Sigma) &= \sum_i \log\left((2\pi)^d \prod \sigma_i^2\right)^{-1/2} - \sum_i \frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu) \\ &= -\frac{n}{2}\log(2\pi)^d \prod \sigma_i^2 - \frac{1}{2}\sum_i (x_i - \mu)^T \Sigma^{-1}(x_i - \mu) \\ &= -\frac{n}{2}\left(d \log 2\pi + \sum_i \log \sigma_i^2\right) - \frac{1}{2}\sum_i (x_i - \mu)^T \Sigma^{-1}(x_i - \mu) \end{aligned}$$

(ii) Find the MLE of μ assuming Σ is known.

Answer. Take the partial with respect to μ:

$$-\frac{1}{2}\sum_i \frac{\partial}{\partial \mu}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu) = -\frac{1}{2}\sum_i -2\Sigma^{-1}(x_i - \mu)$$

which is

$$\sum_i \Sigma^{-1}(x_i - \mu) = \Sigma^{-1}\sum_i (x_i - \mu) = 0$$

Solving for μ:

$$n\mu = \sum_i x_i \implies \mu = \frac{1}{n}\sum_i x_i$$

Computing the second partial derivative:

$$\frac{\partial}{\partial \mu}\sum_i (x_i - \mu) = -n\mu < 0$$

so for the μ we found, it is at a maximum.

(iii) Find the MLE of Σ assuming μ is known.

Answer. Make the substitution $v_j = \frac{1}{\sigma_j^2}$. Then we have:

$$\frac{-n}{2}\left(d\log 2\pi - \sum_{i=1}^{d}\log v_i\right) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)$$

Take the partial with respect to σ_j^2.

$$\begin{aligned}\frac{\partial}{\partial\sigma_j^2}l(\mu,\Sigma) &= \frac{n}{2}\cdot\frac{1}{v_j}v_j^2 - \frac{1}{2}v_j^2\sum_{i=1}^{n}(x_i-\mu)^T(x_i-\mu)\\ &= v_j n - v_j^2\sum_{i=1}^{n}(x_i-\mu)^T(x_i-\mu)\end{aligned}$$

Set it equal to zero and solve for v_j:

$$\begin{aligned}v_j n &= v_j^2\sum_{i=1}(x_i-\mu)^T(x_i-\mu)\\ n &= v_j\sum_{i=1}(x_i-\mu)^T(x_i-\mu)\\ v_j &= \frac{n}{\sum_{i=1}(x_i-\mu)^T(x_i-\mu)}\end{aligned}$$

Since $v_j = \frac{1}{\sigma_j^2}$,

$$\sigma_j^2 = \frac{\sum_{i=1}(x_i-\mu)^T(x_i-\mu)}{n}$$

(iv) Find the joint MLE of (μ,Σ) in terms of the maximum likelihood estimates computed above.

Answer. We see that the MLE of μ does not depend on Σ. So we immediately know the MLE of Σ from part (iii). So the joint MLE is given by our answer to the previous two parts.

(b) Suppose that we have a training set $\{(x_i, y_i) : i = 1,\dots,n\}$ of n independent examples but in which the residual terms had different variances. That is, we assume

$$y_i \sim N(w^T x_i, \sigma_i^2)$$

Show that the MLE estimate of w can be found by solving the following optimization problem

$$w_{MLE} = \mathrm{argmin}_w \|A(Xw-y)\|_2^2$$

Clearly state what the matrix A equals.

Answer. By definition:

$$w_{MLE} = \mathrm{argmax}_w p(D \mid w) = \mathrm{argmax}_w \prod p(y_i \mid w)$$

the probability density of y_i is given by a normal distribution:

$$\begin{aligned}
\operatorname*{argmax}_w \prod p(y_i \mid w) &= \operatorname*{argmax}_w \prod \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{-\frac{1}{2}(y_i - w^T x_i)^T \frac{1}{\sigma_i^2}(y_i - w^T x_i)\right\} \\
&= \operatorname*{argmax}_w -\frac{1}{2}\sum_i (y_i - w^T x_i)^T \frac{1}{\sigma_i^2}(y_i - w^T x_i) \\
&= \operatorname*{argmin}_w \sum_i (y_i - w^T x_i)^T \frac{1}{\sigma_i^2}(y_i - w^T x_i) \\
&= \operatorname*{argmin}_w \|\Sigma^{-1}(y^T - w^T X)^T\|_2^2 \\
&= \operatorname*{argmin}_w \|\Sigma^{-1}(X^T w - y)\|_2^2
\end{aligned}$$

and we're done.

(c) Consider the Categorical$(\theta_1, \theta_2, \ldots, \theta_k)$ distribution. Recall, for categorical distributions, there are two constraints on θ_k:

- $\theta_k \geqslant 0$ for all k
- $\sum_{k=1}^K \theta_k = 1$

The distribution describes a random process that selects one of the K possible categories, with category k being chosen with probability θ_k.

Ignoring the data points X, suppose that for all i from 1 to n, we sample y_i from a categorical distribution:

$$y_i \sim \text{Categorical}(\theta_1, \ldots, \theta_K)$$

Compute the MLE of $\theta = (\theta_1, \ldots, \theta_K)$. Use the fact that the KL divergence is non-negative:

$$\text{KL}(\pi \,||\, \theta) = \sum_{\omega \in \Omega} \pi(\omega) \log\left(\frac{\pi(\omega)}{\theta(\omega)}\right) \geqslant 0$$

Answer.

(d) Again consider X fixed. This time, we suppose that each y_i binary-valued (0 or 1). We choose a model y as

$$y_i \sim \text{Ber}(s(x_i^T w)),\ \forall i = 1, \ldots, n$$

where $s(z) = \frac{1}{1+e^{-z}}$ is the *sigmoid* function and $\text{Ber}(p)$ denotes the Bernoulli distribution which takes value 1 with probability p and 0 with probability $1 - p$.

(i) Write down the log-likelihood $l(w) = \log p(y \mid w)$ and show that finding the MLE of w is equivalent to minimizing the cross entropy between $\text{Ber}(y_i)$ and $\text{Ber}(s(x_i^T w))$ for each i:

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^n H(\text{Ber}(y_i), \text{Ber}(s(x_i^T w))).$$

Definition of cross entropy: given two discrete probability distributions $\pi : \Omega \to [0, 1]$ and $\theta : \Omega \to [0, 1]$ on some outcome space Ω, we define the cross entropy between π and θ as

$$H(\pi, \theta) = \sum_{\omega \in \Omega} -\pi(\omega) \log \theta(\omega)$$

Answer. Using the formula for MLE:

$$\text{MLE}_\omega = \operatorname*{argmax}_\omega \prod p(y_i \mid \omega) = \operatorname*{argmax}_\omega \sum_i \log p(y_i \mid \omega)$$

This is equal to:

$$\operatorname*{argmax}_w \sum_i \log \text{Ber}(s(w^\top x_i))(y_i)$$

Now consider our outcome space $\Omega = \{0, 1\}$. Notice that

$$\text{Ber}(y_i)(x) = I_{y_i}(x) = \begin{cases} 1 & \text{if } x = y_i \\ 0 & \text{otherwise} \end{cases}$$

So we can rewrite the MLE as:

$$\operatorname*{argmax}_w \sum_i \sum_{\omega \in \{0,1\}} I_{y_i}(x) \log \text{Ber}(s(w^\top x_i))(x)$$

This reduces to:

$$\operatorname*{argmax}_w = -\sum_i H(\text{Ber}(y_i), \text{Ber}(s(w^\top x_i))$$

and we can change the argmax to an argmin because of the negative sign.

(ii) Show that (1) and therefore finding the MLE) is equivalent to the following problem:

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^n \log\left(1 + \exp\left(-z_i x_i^\top w\right)\right)$$

where $z_i = 1$ if $y_i = 1$ and $z_i = -1$ if $y_i = 0$.

Note: both (1) and (2) are referred to as logistic regression.

Answer. The previous statement is equivalent to:

$$\operatorname*{argmin}_w \sum_i^n -\log \text{Ber}(s(x_i^\top w))$$

We have the following cases:

$$\text{Ber}(s(x_i^\top w))(x) = \begin{cases} \dfrac{1}{1 + e^{-x_i^\top w}} & \text{if } x = 1 \\ \dfrac{1}{1 + e^{x_i^\top w}} & \text{if } x = 0 \end{cases}$$

because $1 - \frac{1}{1+e^{-z}} = \frac{1}{1+e^{z}}$. So the statement follows algebraically:

$$\begin{aligned} \operatorname*{argmin}_w \sum_i^n -\log \text{Ber}(s(x_i^\top w))(y_i) &= \operatorname*{argmin}_w \sum_i^n \log \text{Ber}(s(x_i^\top w))(y_i)^{-1} \\ &= \operatorname*{argmin}_w \sum_i^n \log 1 + \exp\left(-z_i x_i^\top w\right) \end{aligned}$$

(iii) Let $J(w) = \log\left(1 + \exp\left(-z x^\top w\right)\right)$ where, again, $z = 1$ if $y = 1$ and $z = -1$ if $y = 0$ (we are only considering a single (x, y) pair in this subpart) Prove the following:

(i) J is not strictly convex

(ii) The gradient descent update rule for minimizing $J(w)$ with learning rate ε is

$$w' = w - \varepsilon\left(\frac{1}{1 + e^{-x^\top w}} - y\right)x$$

Geometry of Ridge Regression

You recently learned ridge regression and how it differs from ordinary least squares. In this question we will explore the properties of ridge regression in more depth. Recall that the ridge regression problem is given by the following optimization problem:

$$\underset{w}{\operatorname{argmin}} \|y - Xw\|_2^2 + \nu\|w\|_2^2$$

The solution to ridge regression is given by

$$\hat{w}_r = (X^T X + \nu I)^{-1} X^T y$$

(a) One reason why we might want to have small weights w has to do with the sensitivity of the predictor to its input. Let x be a d-dimensional list of features corresponding to a new test point. Our predictor is $w^T x$. What is an upper bound on how much our prediction could change if we added noise $\varepsilon \in \mathbb{R}^d$ to a test point's features x, in terms of $\|w\|_w$ and $\|\varepsilon\|_w$?

Answer. Our original prediction is

$$w^T x$$

and the new prediction is

$$w^T(x + \varepsilon)$$

Taking the difference:

$$\|w^T x - w^T(x + \varepsilon)\| = \|w^T \varepsilon\| \leqslant \|w^T\|\|\varepsilon\|$$

(b) Note that in computing $\hat{w}_r$, we are trying to invert the matrix $X^T X + \nu I$ instead of the matrix $X^T X$. If $X^T X$ has eigenvalues $\sigma_1^2, \ldots, \sigma_d^2$, what are the eigenvalues of $(X^T X + \nu I)^{-1}$? Comment on why adding the regularizer term νI can improve the inversion operation numerically.

Answer. The new eigenvalues are $\sigma_1^2 + \nu, \sigma_2^2 + \nu, \ldots, \sigma_d^2 + \nu$. This is because we have the decomposition of X as $P^T \Sigma P$. Then:

$$P^T \Sigma P + \nu I = P^T \Sigma P + \nu I P^T P = P^T \Sigma P + P^T(\nu I)P = P^T(\Sigma + \nu I)P$$

The regularizer term adds stability because if σ is a small value, σ^2 is even smaller and taking the inverse $1/\sigma^2$ can be a very large number. Adding a first order term ν makes it so that we don't have to worry about very small values of σ.

(c) Let the number of parameters $d = 4$ and the number of data points $n = 6$, and let the eigenvalues of $X^T X$ be given by $500, 10, 1$, and 0.001. We must now choose between two regularization parameters $\nu_1 = 50$ and $\nu_2 = 0.1$. Which do you think is a better choice for this problem and why?

Answer. $\nu_2 = 0.1$ is probably the better choice because choosing 50 can throw off our original data. For example, choosing 50 would result in eigenvalues of $550, 60, 51, 50.001$. Now it is apparent that there is not much of a difference in the eigenvalues $51, 50.001$. So choosing a ν too high might interfere with the original data.

(d) Another advantage of ridge regression can be seen for under-determined systems. Say we have the data drawn from a $d = 5$ parameter model, but only have $n = 4$ training samples of it, i.e. $X \in \mathbb{R}^{4\times 5}$. Now this is clearly an under determined system, since $n < d$. Show that ridge regression with $\nu > 0$ results in a unique solution, whereas ordinary least squares has an infinite number of solutions.

Answer. If X was not full rank, then there is a vector w^* in the null space. So notice that in another prediction:

$$w + w^* \implies X(w + w^*) = Xw$$

$w + w^*$ gives a projection similar to just taking w. So once we have one solution, we have infinitely many.

To see that the ridge regression yields a unique solution, we can take the partial derivative with respect to w:

$$\begin{aligned}\frac{\partial}{\partial w}\|y - Xw\|_2^2 + v\|w\|_2^2 &= \frac{\partial}{\partial w}(y - Xw)^T(y - Xw) + 2vw \\ &= \frac{\partial}{\partial w}\left(-y^TXw - w^TX^Ty + w^TX^TXw\right) + 2vw \\ &= -2X^Ty + 2X^TXw + 2vw\end{aligned}$$

Now set the equation to 0 and solve for w:

$$\begin{aligned}-2X^Ty + 2X^TXw + 2vw &= 0 \\ (2X^TX + 2vI)w &= 2X^Ty \\ w &= (X^TX + 2vI)^{-1}X^Ty\end{aligned}$$

so the solution is unique.

(e) What will the solution to ridge regression (4) converge to if you take the limit $v \to 0$? Your answer should be a simple expression in terms of U, Σ, V, y, and v where $X = U\Sigma V^T$ is the SVD of X.

Answer. It will converge to the regular linear regression: $(X^TX)^{-1}X^Ty$. If we write it in terms of U, Σ, V, y:

$$(V\Sigma U^TU\Sigma V^T)^{-1}(V\Sigma U^T)y = (V\Sigma^{-2}V^T)V\Sigma U^Ty = V\Sigma^{-1}U^Ty$$

(f) Tikhonov regularization is a general term for ridge regression, where the implicit constraint set takes the form of an ellipsoid instead of a ball. In other words, we solve the optimization problem

$$w = \underset{w}{\operatorname{argmin}}\|y - Xw\|_2^2 + v\|\Gamma w\|_2^2$$

for some full rank matrix $\Gamma \in \mathbb{R}^{d\times d}$. Derive a closed form solution for w.

Answer. Take the partial of the constraint term with respect to w:

$$\frac{\partial}{\partial w}v\|\Gamma w\|_2^2 = \frac{\partial}{\partial w}vw^T\Gamma^T\Gamma w = 2v\Gamma^T\Gamma w$$

Now recall that we had taken the partial of the error term with respect to w:

$$\frac{\partial}{\partial w}\|y - Xw\|_2^2 = -2X^Ty + 2X^TXw$$

Add both these together and solve for w:

$$\begin{aligned}-2X^Ty + 2X^TXw + 2v\Gamma^T\Gamma w &= 0 \\ (2X^TX + 2v\Gamma^T\Gamma)w &= 2X^Ty \\ w &= (X^TX + v\Gamma^T\Gamma)^{-1}X^Ty\end{aligned}$$

Robotic Learning of Controls from Demonstrations and Images

Huey, a home robot, is learning to retrieve objects from a cupboard. The goal is to push obstacle objects out of the way to expose a goal object. Huey's robot trainer, Anne provides demonstrations via tele-operation. When tele-operating the robot, Anne can look at the images captured by the robot and provide controls to Huey remotely.

During a demonstration, Huey records the RGB images of the scene for each of the n timesteps, $x_1, \ldots, x_n$, where $x_i \in \mathbb{R}^{30\times30\times3}$ and the controls for his body for each of the n timesteps, $u_1, \ldots, u_n$ where $u_i \in \mathbb{R}^3$. The controls correspond to making small changes in the 3D pose (i.e. translation and rotation) of his body. Examples of the data are shown in the figure.

Under an assumption (sometimes called the Markovian assumption) that all that matters for the current control is the current image, Huey can try to learn a linear *policy* π (where $\pi \in \mathbb{R}^{2700\times3}$) which linearly maps image states to controls (i.e. $\pi^T x = u$). We will now explore how Huey can recover this policy using linear regression.

Note the dimensions in this problem! Previously, you saw linear regression in problems in which the learned weight w^* was a vector and the predicted value y was a scalar. Here, we are predicting 3D controls. This means that the learned policy is a matrix. In essence, we are performing 3 regressions at the same time, one for each element of the predicted control u.

Please stick to `numpy` (and `numpy.linalg`) only or performing any computations in this assignment. We will ask that you edit the file `robotic_ridge_code.py` directly, instead of working in a Python notebook, and submit it to the Gradescope autograder after you are finished. Please don't rename the file, or change any of the function signatures!

(a) To get familiar with the structure of the data, **please visualize the 0th, 10th and 20th images in the training dataset. Also find their corresponding control vectors**.

Note: the training and testing images are currently stored as float32 numpy arrays, with pixel values in the range [0.0, 255.0]. You may have to convert these images to the np.uint8 format to visualize them.

(b) Load the n training examples from `x_train.p` and compose the matrix X, where $X \in \mathbb{R}^{n\times2700}$. Note that you will need to flatten the images and reduce them to a single vector. The flattened image vector will be denoted by $\bar{x}$ (where $\bar{x} \in \mathbb{R}^{2700\times1}$). Next, lead the n examples from `y_train.p` and compose the matrix U, where $U \in \mathbb{R}^{n\times3}$. Try to perform ordinary least squares by forming the matrix $(X^TX)^{-1}X^T$ for solving

$$\underset{\pi}{\operatorname{argmin}} \|X\pi - U\|_F$$

in order to learn the optimal *policy* $\pi^* \in \mathbb{R}^{2700\times3}$. **Report what happens as you attempt to do this and explain why**.

(c) Now try to perform ridge regression:

$$\underset{\pi}{\operatorname{argmin}} \|X\pi + U\|_F^2 + \lambda\|\pi\|_F^2$$

on the data set for regularization values $\lambda = \{0.1, 1.0, 10, 100, 1000\}$. Measure the average squared Euclidean distance for the accuracy of the policy on the training data:

$$\frac{1}{n}\sum_{i}^{n-1} \|\bar{x}_i^T\pi - u_i^T\|_2^2$$

In the expression above, we are taking the l_2 norm of a row vector, which here we take to mean the l_2 norm of the column vector we get by transposing it. **Report the training error results for each value of** λ.

(d) Next, we are going to try standardizing the states. For each pixel value in each data point, $\bar{x}$, perform the following operation:

$$\bar{x} \mapsto \frac{\bar{x}}{255} \times 2 - 1$$

We know that the maximum pixel value is 255, so this operation rescales the data to be in the range $[-1, 1]$. **Repeat the previous part and report the average squared training error for each value of** λ.

(e) Evaluate both *policies* (i.e. with and without standardization) on the new validation data `x_test.p` and `y_test.p` for the different values of λ. **Report the average squared Euclidean loss and qualitatively explain how changing the values of** λ **affects the performance in terms of bias and variance**.

(f) To better understand how standardizing improved the loss function, we are going to evaluate the *condition number* κ of the optimization problem above, which is defined as

$$\kappa = \frac{\sigma_{max}(X^T X + \lambda I)}{\sigma_{min}(X^T X + \lambda I)}$$

or the ratio of the maximum singular value to the minimum singular value of the relevant matrix. Roughly speaking, the condition number of the optimization process measures how stable the solution will be when some error exists in the observations. More precisely, given a linear system $Ax = b$, the condition number of the matrix A is the maximum ratio of the relative error in the solution x to the relative error of b.

For the regularization value of $\lambda = 100$, **report the condition number with the standardization technique applied and without**.

```python
import pickle

import matplotlib.pyplot as plt
import numpy as np
import numpy.linalg as LA
from PIL import Image


def load_data() -> tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray]:
    x_train = pickle.load(open("x_train.p", "rb"), encoding="latin1")
    y_train = pickle.load(open("y_train.p", "rb"), encoding="latin1")
    x_test = pickle.load(open("x_test.p", "rb"), encoding="latin1")
    y_test = pickle.load(open("y_test.p", "rb"), encoding="latin1")
    return x_train, y_train, x_test, y_test


def visualize_data(images: np.ndarray, controls: np.ndarray) -> None:
    """
    Args:
        images (ndarray): image input array of size (n, 30, 30, 3).
        controls (ndarray): control label array of size (n, 3).
    """
    # Current images are in float32 format with values between 0.0 and 255.0
    # Just for the purposes of visualization, convert images to uint8
    images = images.astype(np.uint8)
```

```
    controls = controls.astype(np.uint8)

    # TODO: Your code here!
    picked = [0, 10, 20]
    f, ax = plt.subplots(1, len(picked))
    for i in range(len(picked)):
        n = picked[i]
        ax[i].imshow(images[n])
    plt.show()


def compute_data_matrix(
    images: np.ndarray, controls: np.ndarray, standardize: bool = False
) -> tuple[np.ndarray, np.ndarray]:
    """
    Args:
        images (ndarray): image input array of size (n, 30, 30, 3).
        controls (ndarray): control label array of size (n, 3).
        standardize (bool): boolean flag that specifies whether the images should be st

    Returns:
        X (ndarray): input array of size (n, 2700) where each row is the flattened imag
        Y (ndarray): label array of size (n, 3) where row i corresponds to the control
    """
    # TODO: Your code here!
    n, w, h, c = images.shape
    X = np.reshape(images, (n, w * h * c))
    Y = controls
    if standardize:
        X = 2 * X / 255 - 1
        Y = 2 * Y / 255 - 1
    return X, Y


def ridge_regression(X: np.ndarray, Y: np.ndarray, lmbda: float) -> np.ndarray:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        Y (ndarray): label array of size (n, 3).
        lmbda (float): ridge regression regularization term

    Returns:
        pi (ndarray): learned policy of size (2700, 3)
    """
    # TODO: Your code here!
    return np.dot(np.linalg.inv(X.T @ X + lmbda * np.identity(X.shape[1])) @ X.T, Y)


def ordinary_least_squares(X: np.ndarray, Y: np.ndarray) -> np.ndarray:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        Y (ndarray): label array of size (n, 3).

    Returns:
        pi (ndarray): learned policy of size (2700, 3)
```

```
    """
    # TODO: Your code here!
    return np.dot(np.linalg.inv(X.T @ X) @ X.T, Y)


def measure_error(X: np.ndarray, Y: np.ndarray, pi: np.ndarray) -> float:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        Y (ndarray): label array of size (n, 3).
        pi (ndarray): learned policy of size (2700, 3)

    Returns:
        error (float): the mean Euclidean distance error across all n samples
    """
    # TODO: Your code here!
    n, c = Y.shape
    return np.linalg.norm(X @ pi - Y) / n


def compute_condition_number(X: np.ndarray, lmbda: float) -> float:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        lmbda (float): ridge regression regularization term

    Returns:
        kappa (float): condition number of the input array with the given lambda
    """
    # TODO: Your code here!
    n, p = X.shape
    m = X.T @ X + lmbda * np.identity(p)
    sigma = np.linalg.svdvals(m)
    return max(sigma) / min(sigma)


if __name__ == "__main__":

    x_train, y_train, x_test, y_test = load_data()
    print("successfully loaded the training and testing data")

    LAMBDA = [0.1, 1.0, 10.0, 100.0, 1000.0]

    # TODO: Your code here!
    # visualize_data(x_test, y_test)
    n, w, h, c = x_train.shape
    X, Y = compute_data_matrix(x_test, y_test, standardize=True)
    for lmbda in LAMBDA:
        w = ridge_regression(X, Y, lmbda)
        error = measure_error(X, Y, w)
        cond = compute_condition_number(X, lmbda)
        print(error, cond)
```