CS189: Introduction to Machine Learning

Homework 2

Due: September 27th, 2016, 12:00 noon, NOT MIDNIGHT

Problem 1: Visualizing Eigenvectors of Gaussian Covariance Matrix

We have two one dimensional random variables $X_1 \sim \mathcal{N}(4,4)$ and $X_2 \sim 0.5X_1 + \mathcal{N}(3,9)$, where $\mathcal{N}(\mu,\sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2. In software, draw $N = 100$ random samples of X_1 and of X_2.

(a) Compute the mean of the sampled data.

(b) Compute the covariance matrix of the sampled data.

(c) Compute the eigenvectors and eigenvalues of this covariance matrix.

(d) On a two dimensional grid with a horizontal axis for X_1 ranging from $[-15, 15]$ and a vertical axis for X_2 ranging from $[-15, 15]$, plot the following:

i) All $N = 100$ data points

ii) Arrows representing both covariance eigenvectors. The eigenvector arrows should originate from the mean and have magnitude equal to their corresponding eigenvalues.

(e) By placing the eigenvectors of the covariance matrix into the columns of a matrix $U = [v_1\ v_2]$, where v_1 is the eigenvector corresponding to the largest eigenvalue, we can use U' as a rotation matrix to rotate each of our sampled points from our original (X_1, X_2) coordinate system to a coordinate system aligned with the eigenvectors (without the transpose, U can rotate back to the original axes). Center your data points by subtracting the mean and then rotate each point by U', specifically $x_{\text{rotated}} = U'(x - \mu)$. Plot these rotated points on a new two dimensional grid with both axes ranging from [-15,15].

Problem 2: Covariance Matrixes and Decompositions

As described in lecture, a covariance matrix $\Sigma \in \mathbb{R}^{N \times N}$ for a random variable $X \in \mathbb{R}^N$ with the following values, where $\text{cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)]$ is the covariance between the i-th and j-th elements of the random vector X:

$$\Sigma = \begin{bmatrix} \text{cov}(X_1, X_1) & ... & \text{cov}(X_1, X_n) \\ ... & & ... \\ \text{cov}(X_n, X_1) & ... & \text{cov}(X_n, X_n) \end{bmatrix} . \tag{1}$$

For now, we are going to leave the formal definition of covariance matrices aside and focus instead on some transformations and properties. The motivating example we will use is the N dimensional Multivariate Gaussian Distribution defined as follows when Σ is positive definite:

$$f(x) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)} . \tag{2}$$

Here, $|\Sigma|$ denotes the determinant of the matrix Σ.

(a) We usually assume that Σ^{-1} exists, but in many cases it will not. Describe the conditions for which Σ_X^{-1} corresponding to random variable X will not exist. Explain how to convert the random variable X into a new random variable X' without loss of information where $\Sigma_{X'}^{-1}$ does exist.

(b) Consider a data point x drawn from a zero mean Multivariate Gaussian Random Variable $X \in \mathbb{R}^N$ like shown above. Prove that there exists matrix $A \in R^{N,N}$ such that $x^\top \Sigma^{-1} x = \|Ax\|_2^2$ for all vectors x. What is the matrix A?

(c) In the context of Multivariate Gaussians from the previous problem, what is the intuitive meaning of $x^\top \Sigma^{-1} x$ when we transform it into $\|Ax\|_2^2$?

(d) Lets constrain $\|x\|_2 = 1$. In other words, the ℓ_2 norm (or magnitude) of vector x is 1. In this case, what is the maximum and minimum value of $\|Ax\|_2^2$? If we have $X_i \perp\!\!\!\perp X_j \; \forall i, j$, then what is the intuitive meaning for the maximum and minimum value of $\|Ax\|_2^2$? To maximize the probability of $f(x)$, which x should we choose?

Problem 3: Isocontours of Normal Distributions

Let $f(\mu, \Sigma)$ denote the density function of a Gaussian random variable. Plot isocontours of the following functions:

a) $f(\mu, \Sigma)$, where $\mu = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$.

b) $f(\mu, \Sigma)$, where $\mu = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$.

c) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ and $\Sigma_1 = \Sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$.

d) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$.

e) $f(\mu_1, \Sigma_1) - f(\mu_2, \Sigma_2)$, where $\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\mu_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$, $\Sigma_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ and $\Sigma_2 = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

Problem 4: Featurized Linear Classifers and Gradient Descent

In the last homework, we fit a classifier that was linear in the pixel intensities to the MNIST data. For classification of digits the raw pixel values are very, very bad features: it's pretty hard to separate digits with linear functions in pixel space. The standard solution to the this is to come up with some transform ψ of the original pixel values such that the transformed points are (more easily) linearly separable. Somewhat surprisingly, *random* nonlinear functions of the original pixel values can work quite well.

In this problem, you'll use the feature transform $\psi : \mathbf{R}^p \to \mathbf{R}^d$ defined by two parameters, $G \in \mathbf{R}^{(p,d)}$ and $b \in \mathbf{R}^d$ by:

$$\psi(x) = \cos(G^T x + b).$$

Basically, we're transforming our original p pixel intensities into $d = 5000$ new features. We'll choose G to be a *random* matrix, with each entry sampled i.i.d. with mean 0 and variance σ^2, and b to be a random vector sampled i.i.d. from the uniform distribution on $[0, 2\pi]$. These features look pretty simple, but are deceptively powerful.

We then learn a linear classifier on top of these nonlinear features by solving the following optimization problem:

$$W^* = \text{argmin}_{W \in \mathbb{R}^{d \times k}} \sum_{i=0}^{n} \|W^\mathsf{T} \psi(x_i) - y_i\|_2^2 + \lambda \|W\|_F^2$$

Here $x_i \in \mathbf{R}^p$ are the original pixel intensities, and $\psi(x_i) \in \mathbf{R}^d$ are the new features. This objective function is *the same* as the objective function for least-squares classification in the last homework, we've simply transformed the data.

Remember that you are **NOT** allowed to use any of the prebuilt classifiers in `sklearn`. Feel free to use any method from `numpy` or `scipy`.

a) In the file `hw2.py`, complete the featurization function `phi` using the scheme described above. `phi` should take a *matrix* of pixel intensities and return a matrix of new features. Use the closed form solution for W^* from last time to classify MNIST digits.

b) Write out the gradient update equation, then apply gradient descent using the gradient ∇_W computed with all training examples. Our method was able to converge in approximately 15000 iterations for our choice of d and σ^2, but feel free to experiment with α, λ, d and σ to see the effect on training.

c) Similarly, write out the stochastic gradient descent update equation, then apply stochastic gradient descent using one example at a time to compute the gradient.

d) Plot the training error as a function of the number of iterations for gradient descent and stochastic gradient descent. Report the choice of learning rate and regularization term, and explain the difference in behavior between the two algorithms.

e) You have been training on the 60000 digits in the training data and validating on the 10000 digits in the official MNIST test set. We've prepared a never-before-seen set of 10000 digits that will serve as your test set. Download this test set from Kaggle, generate predictions for the Kaggle data, and save your predictions to a CSV file. Your CSV file should have the header line `Id,Category`, and the ID's are zero-indexed. Submit these predictions to Kaggle. You can only submit twice per day, so get started early! Finally, write your Kaggle score in your writeup.

Like last time, remember to include a screenshot or LaTeX snippet of your code as the last page of your PDF writeup, and to submit a zip file of your code as well.

Problem 5: Regularization and Risk Minimization

a) Let A be a $d \times n$ matrix. For any $\mu > 0$, show that $(AA^\top + \mu I)^{-1}A = A(A^\top A + \mu I)^{-1}$.

b) Let $(x_1, y_1), \cdots, (x_n, y_n)$ be a sequence of data points. Each y_i is a scalar and each x_i is a vector in $\mathbb{R}^d$. Let $X = [x_1, \ldots, x_n]^\top$ and $y = [y_1, \ldots, y_n]^\top$. Consider the *regularized* least squares problem.

$$\min_{w \in \mathbb{R}^d} \|Xw - y\|_2^2 + \mu \|w\|_2^2$$

Show that the optimum w_* is unique and can be written as the linear combination $w_* = \sum_{i=1}^n \alpha_i x_i$ for some scalars α. What are the coefficients α_i?

c) More generally, consider the general regularized empirical risk minimization problem

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \text{loss}(w^\top x_i, y_i) + \mu \|w\|_2^2$$

where the loss function is convex in the first argument. Prove that the optimal solution has the form $w_* = \sum_{i=1}^n \alpha_i x_i$ If the loss function is not convex, does the optimal solution have the form $w_* = \sum_{i=1}^n \alpha_i x_i$? Justify your answer.

Problem 6: MLE For Simple Linear Regression
Simple linear regression refers to the case of linear regression in which the input is a scalar quantity.

Let the data set be $\{(x_i, y_i)\}_{i=1}^n$, where each sample is drawn independently from a joint distribution over input and output: $(x_i, y_i) \sim (X, Y)$. Assume the Gaussian noise setting:

$$y_i | x_i \sim \mathcal{N}(w_0 + w_1 x_i, \sigma^2)$$

.
Show that the MLE in this simple linear regression model is given by the following equations, which may be familiar from basic statistics classes:

$$\begin{aligned} w_1 &= \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{\sum_i x_i y_i - n\bar{x}\bar{y}}{\sum_i x_i^2 - n\bar{x}^2} \\ w_0 &= \bar{y} - w_1 \bar{x}. \end{aligned}$$

From statistics, we know by the Law of Large Numbers that $w_1 \approx \frac{\mathrm{cov}(X,Y)}{\mathrm{var}(X)}$ and $w_0 \approx \mathbb{E}[y] - w_1 \mathbb{E}[X]$ as the number of samples increases (you don't have to prove this).

Problem 7: Independence vs. Correlation

(a) Consider the random variables X and Y in $\mathbb{R}$ with the following conditions.

  (i) X and Y can take values $\{-1, 0, 1\}$.

  (ii) When X is 0, Y takes values 1 and -1 with equal probability ($\frac{1}{2}$). When Y is 0, X takes values 1 and -1 with equal probability ($\frac{1}{2}$).

  (iii) Either X is 0 with probability ($\frac{1}{2}$), or Y is 0 with probability ($\frac{1}{2}$).

Are X and Y uncorrelated? Are X and Y independent? Prove your assertions. *Hint:* Graph these points onto the Cartesian Plane. What's each point's joint probability?

(b) Consider three Bernoulli random variables B_1, B_2, B_3 which take values $\{0, 1\}$ with equal probability. Lets construct the following random variables X, Y, Z: $X = B_1 \oplus B_2$, $Y = B_2 \oplus B_3$, $Z = B_1 \oplus B_3$, where $\oplus$ indicates the XOR operator. Are X, Y, and Z pairwise independent? Mutually independent? Prove it.

Submission Instructions

You will submit:

- A PDF write-up containing your answers and plots to Gradescope.
- A zip file of your code to Gradescope.
- Your predictions CSV to Kaggle.