CPSC 540 Assignment 2 (due September 24)

Gaussians and Linear Regression

This assignment has three purposes:

1. To give you practice using and manipulating Gaussian distributions.
2. To give you some practice computing gradients and Hessians.
3. To let you experiment with issues related to linear regression.

As before, updates/clarifications following the first version of this document will be highlighted in red.

1 Further logistics information

Write your name, student number, and whether you are registered/auditting/crashing CPSC 540. Also, please staple your homework together for the sake of the marker.

2 Properties of Gaussians

2.1 Maximum Likelihood Estimation

Assume we have a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ where each $x_i \in \mathbb{R}$ and $y_i \in \{1, 2\}$ (i.e., we have a single continuous feature and a binary label). We will consider a Gaussian discrimnant analysis (GDA) model of this data under the linear discriminant analysis (LDA) assumption that $\sigma_1 = \sigma_2$, so that

$$x_i | y_i = 1 \sim \mathcal{N}(\mu_1, \sigma^2), \quad x_i | y_i = 2 \sim \mathcal{N}(\mu_2, \sigma^2), \quad y_i \sim \text{Ber}(\theta),$$

and thus

$$p(x_i | y_i = 1, \mu_1, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_1)^2}{2\sigma^2}\right),$$

$$p(x_i | y_i = 2, \mu_2, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_2)^2}{2\sigma^2}\right).$$

Derive the maximum likelihood estimate (MLE) for the parameters $\{\mu_1, \mu_2, \sigma\}$ in the corresponding Gaussian discriminant analysis (GDA) model. Show your work..

Hint: The log-likelihood can be written

$$\begin{aligned}
\log p(\mathcal{D}|\mu_1,\mu_2,\sigma^2,\theta) &= \sum_{i=1}^{N} \log p(x_i,y_i|\mu_1,\mu_2,\sigma^2,\theta)\\
&= \sum_{i=1}^{N} \left[\log\left(p(x_i|y_i,\mu_1,\mu_2,\sigma^2,\theta)p(y_i|\mu_1,\mu_2,\sigma^2,\theta)\right)\right] \qquad \text{(product rule)}\\
&= \sum_{i=1}^{N} \left[\log p(x_i|y_i,\mu_1,\mu_2,\sigma^2,\theta) + \log p(y_i|\mu_1,\mu_2,\sigma^2,\theta)\right] \qquad (\log(ab)=\log(a)+\log(b))\\
&= \sum_{i=1}^{N} [\log p(x_i|y_i,\mu_{y_i},\sigma^2) + \log p(y_i|\theta)] \qquad (x_i|y_i \text{ depends on } \{\mu_{y_i},\sigma^2\},\ y_i \text{ depends on } \theta)\\
&= \sum_{i=1}^{N} [I(y_i=1)\log p(x_i|y_i=1,\mu_1,\sigma^2) + I(y_i=2)\log p(x_i|y_i=2,\mu_2,\sigma^2) + \log p(y_i|\theta)].
\end{aligned}$$

We already know that the MLE for θ is N_1/N (where N_1 is the number of times $y_i = 1$). You can find the MLE for the other parameters by first equating its derivative with respect to μ_1 to 0 and solving (observe that the derivative of $\log p(y_i|\theta)$ with respect to μ_1 is 0), then doing the same for μ_2 and finally for σ (this stationary point happens to be the maximizer of the likelihood, but you don't need to show this).

2.2 Self-Conjugacy for the Mean Parameter

If x is distributed according to a Gaussian with mean μ,

$$x \sim \mathcal{N}(\mu, \sigma^2),$$

and we assume that μ itself is distributed according to a Gaussian

$$\mu \sim \mathcal{N}(\alpha, \gamma^2),$$

then the posterior $\mu|x$ also follows a Gaussian distribution.[1] Derive the form of the (Gaussian) distribution for $p(\mu|x,\alpha,\sigma^2,\gamma^2)$. You can assume that $\sigma = 1$ and $\gamma = 1$.

Hint: Use Bayes rule to get

$$\begin{aligned}
p(\mu|x,\alpha,\sigma^2,\gamma^2) &= \frac{p(x|\mu,\alpha,\sigma^2,\gamma^2)p(\mu|\alpha,\sigma^2,\gamma^2)}{p(x|\alpha,\sigma^2,\gamma^2)}\\
&= \frac{p(x|\mu,\sigma^2)p(\mu|\alpha,\gamma^2)}{p(x|\alpha,\sigma^2,\gamma^2)}\\
&\propto p(x|\mu,\sigma^2)p(\mu|\alpha,\gamma^2).
\end{aligned}$$

From here, the 'proportional to' sign ($\propto$) is your friend on this question. Above we used it to get rid of $p(x|\alpha,\sigma^2,\gamma^2)$ since it doesn't depend on μ (it just contributes to making sure that $\int_{-\infty}^{\infty} p(\mu|x,\alpha,\sigma^2,\gamma^2) = 1$). Compute the product above but use $\propto$ to get rid of other terms that don't depend on μ.

[1]We say that the Gaussian distribution is the 'conjugate prior' for the Gaussian mean parameter (we'll formally discuss conjugate priors later in the course). Another reason the Gaussian distribution is important is that is the only (non-trivial) continuous distribution that has this 'self-conjugacy' property.

You can then ‘complete the square’ to make the product look like a Gaussian distribution, e.g. when you have $\exp(ax^2 - bx + \text{const})$ you can factor out an a and add/subtract $(b/2a)^2$ to re-write it as

$$\begin{aligned}\exp\left(ax^2 - bx + const\right) &\propto \exp\left(ax^2 - bx\right) = \exp\left(a(x^2 - (b/a)x)\right)\\ &\propto \exp\left(a(x^2 - (b/a)x + (b/2a)^2)\right) = \exp\left(a(x - (b/2a))^2\right).\end{aligned}$$

Note that multiplying by factors that do not depend on μ within the exponent does not change the distribution. In this question you will want to complete the square to get the distribution on μ, rather than x.

Once it looks proportional to a Gaussian, use that the normalizing constant is going to be the value that satisfies $\int_{-\infty}^{\infty} p(x) = 1$. And since we've already been told it will be a Gaussian, we can use

$$\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = 1,$$

or equivalently that

$$\int_{-\infty}^{\infty} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \sigma\sqrt{2\pi}.$$

You may find it easier to solve thie problem if you parameterize the Gaussians in terms of their ‘precision’ parameters (e.g., $\lambda = 1/\sigma^2$, $\lambda_0 = 1/\gamma^2$) rather than their variances σ and γ.

2.3 Unsupervised Gaussian Density Models

Expand the file *ass2.zip*, switch to the created directory, and (in Matlab) run the script *gaussianDemo*. This script:

1. Loads two datasets, $X1$ and $X2$, each containing samples of two variables.
2. Displays a 3 by 2 plot, where the left column entries show $X1$ and the right entries show $X2$.
3. The top row displays the contours of the Gaussian probability density function (PDF) using the MLE assuming a covariance Σ that is a scalar times the identity matrix, $\Sigma = \beta I$ (analogous to Section 2.1 above).

Modify this demo so that the middle row displays the PDF contours for the MLE when Σ is only constrained to be a diagonal matrix (e.g., we make the independent variables assumption as in naive Bayes), and so that the bottom row displays the PDF contours for the MLE when Σ is a general matrix ($\hat{\Sigma}$ from class). Hand in your updated 3 by 2 plot.

3 Gradients and Hessians

Let f be a real-valued function with domain $\mathbb{R}^d$ (e.g., $f(x) = x^T Ax + b^T x$). Let $\frac{\partial f}{\partial x_i}$ be the partial derivative of f with respect to variable i. The *gradient* vector is defined as the length-d (column-)vector of first partial derivatives.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}$$

The *Hessian* is defined as the d-by-d matrix of second partial derivatives,

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 x_1} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_d} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_d} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_d x_1} & \frac{\partial^2 f}{\partial x_d x_2} & \cdots & \frac{\partial^2 f}{\partial x_d x_d} \end{bmatrix}$$

3.1 Linear and Quadratic Functions

Derive the gradient and Hessian of the following linear and quadratic real-valued functions (where α is a scalar, a and b are vectors of lenth d, A is a matrx of size d-by-d, and W is a symmetric matrix of size d-by-d), and express the result in vector/matrix notation (e.g., there should be no summation notation in the final results).

1.	$f(x) = x^T a + \alpha$	(linear)
2.	$f(x) = a^T x + a^T Ax + x^T Ab$	(more linear forms)
3.	$f(x) = \|x\|^2$	(squared Euclidean norm)
4.	$f(x) = x^T Ax$	(quadratic - N.B., A may not be symmetric)
5.	$f(x) = \frac{1}{2}(Ax-b)^T W(Ax-b)$	(weighted least squares)

Hint: As a sanity check make sure that the dimensions of your result are correct. For linear terms: use that $x^T a = \sum_{i=1}^d a_i x_i$, take the partial derivative with respect to a general element i, then write out the vector containing these elements (remember that Ab is a vector so $x^T Ab$ is a linear term too). For quadratic terms: use that

$$x^T Ax = x^T \begin{bmatrix} \sum_{j=1}^d a_{1j}x_j \\ \sum_{j=1}^d a_{2j}x_j \\ \vdots \\ \sum_{j=1}^d a_{dj}x_j \end{bmatrix} = \sum_{i=1}^d x_i \sum_{j=1}^d a_{ij}x_j.$$

3.2 Logistic Regression

The logistic regression likelihood for a single data point has the form

$$p(y_i|x_i,w) = \frac{1}{1+\exp(-y_i w^T x_i)},$$

where each $x_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$, and $w \in \mathbb{R}^d$. Thus, the log-likelihood for an IID data-set will be

$$\begin{aligned}
\log p(Y|X,w) &= \log \prod_{i=1}^{N} p(y_i|x_i,w) && \text{(IID assumption)}\\
&= \sum_{i=1}^{N} \log p(y_i|x_i,w) && (\log(ab) = \log(a) + \log(b))\\
&= \sum_{i=1}^{N} \log \left(\frac{1}{1+\exp(-y_i w^T x_i)}\right) && \text{(logistic regression assumption)}\\
&= \sum_{i=1}^{N} \log(1) - \log(1+\exp(y_i w^T x_i)) && (\log(a/b) = \log(a) - \log(b))\\
&= \sum_{i=1}^{N} -\log(1+\exp(y_i w^T x_i)), && (\log(1) = 0)
\end{aligned}$$

and the negative log-likelihood (NLL) has the form

$$f(w) = \sum_{i=1}^{N} \log(1+\exp(-y_i w^T x_i)).$$

Compute the gradient and Hessian of this function. Express the result in terms a matrix X (where each row contains and x_i), a vector Y (where each element contains a y_i), and a vector $p(w)$ (where each element contains $p(y_i|x_i,w)$). Use the notation $\text{diag}(v)$ to denote a diagonal matrix that has the elements of a vector v along its diagonal.

Hint: you may want to use these formulas:

$$\frac{1}{1+\exp(-y_i w^T x_i)} = \frac{\exp(y_i w^T x_i)}{1+\exp(y_i w^T x_i)}, \quad p(y_i|x_i,w) = 1 - p(-y_i|x_i,w).$$

3.3 Sum of compositions of general and linear functions

Let A be an n-by-d matrix, x be a column vector with length d, g be a twice-differentiable function from $\mathbb{R}$ to $\mathbb{R}$, and f be defined as

$$f(x) = \sum_{i=1}^{n} g(x^T a_i),$$

where a_i is a (column-)vector formed from the rows i of A. Express the gradient and Hessian of f in terms of A, a vector G' where element i of the vector contains the derivative $g'(u_i)$ with $u_i = x^T a_i$, and a vector G'' containing the deriatives $g''(u_i)$.

4 Linear Regression

The script *linearRegressionDemo* does the following

1. Loads a design matrix X and target vector y, and corresponding $Xtest$ and $ytest$.

2. Fits a least-squares linear regression model to the dataset,

$$\arg\min_w \sum_{i=1}^{N} \frac{1}{2}(w^T x_i - y_i)^2$$

using the function *fitL2*.

3. Plots the dataset and the fitted model, and reports the test error.

Unfortunately, the fit of the model to this data set is quite poor because the dataset contains *outlier* (points that do not fit the overall trend in the dataset).

4.1 Bias variable

Notice that in the first column of X, all values are set to 1. What is the purpose of including this column?

4.2 Weighted Linear Regression

The outlier datapoints are the last 100 rows of X. Write a new function *fitWeightedL2(X,y,z)* that solves the weighted linear regression problem,

$$\arg\min_w \sum_{i=1}^{N} \frac{z_i}{2}(w^T x_i - y_i)^2,$$

where z is a vetcor with the weights z_i. Hand in this new function. Set the weight z_i is set to 1/10 for the outlier datapoints and 1 for the remaining datapoints. Report the test set error under this weighting.

4.3 Robust Linear Regression

In high-dimensions, it may be difficult to determine whether points are indeed outliers. In such cases, it is preferable to replace the squared error with an error that is more robust to outliers. Write a new function, *fitL1(X,y)*, that minimizes the absolute error instead of the square error,

$$\arg\min_w \sum_{i=1}^{N} |w^T x_i - y_i|.$$

You can use the formulation from class or the formulation in Section 7.4 of MLAPP to formulate this problem as a linear program. To solve the linear program, you can use *linprog*. Hand in the new function and report the test set error under this weighting.

5 Non-Linear Regression

The script *basisDemo* load another regression training set and displays the result of applying a simple linear regression model. This time, the model is problematic because the target is a non-linear function of the input. Write a new function, *polyBasis(X,degree)*, that takes a (column-vector) X and the polynomial order *degree*, and returns a new design $Xpoly$ where each row contains the values $(X_i)^j$ for $j = 0$ up to *degree*. E.g., *polyBasis(X,3)* should return

$$Xpoly = \begin{bmatrix} 1 & X_1 & (X_1)^2 & (X_1)^3 \\ 1 & X_2 & (X_2)^2 & (X_2)^3 \\ \vdots & & & \\ 1 & X_N & (X_N)^2 & (X_N)^3 \end{bmatrix}$$

Hand in the new function and a 3 by 3 plot showing the fit for $degree = 0$ up to $degree = 8$.