UC Berkeley
Department of Electrical Engineering and Computer Science
Department of Statistics

EECS 281A / STAT 241A Statistical Learning Theory

Problem Set 2
Fall 2012

Issued: Tues. Sep. 4, 2012 **Due:** Thurs. Sep. 13, 2012

Reading: Chapters 6 and 8

Problem 2.1

The course homepage has a data set named `lms.dat` that contains twenty rows of three columns of numbers. The first two columns are the components of an input vector x and the last column is an output y value. (We will not use a constant term for this problem; thus the input vector and the parameter vector are both two dimensional.)

(a) Solve the normal equations for these data to find the optimal value of the parameter vector. (I recommend using MATLAB or R.)

Solution: By solving the normal equations, we have:

$$\theta^* = (X^\top X)^{-1} X^\top y = \begin{pmatrix} 1.0395 \\ -0.9764 \end{pmatrix}.$$

(b) Find the eigenvectors and eigenvalues of the covariance matrix of the input vectors and plot contours of the cost function $\mathcal{L}(\theta) = \|y - X\theta\|_2^2$ in the parameter space. These contours should of course be centered around the optimal value from part (a).

Solution: The covariance matrix of the data is $C = \frac{1}{n} X^\top X$, where $n = 20$ is the number of data points. Note that the covariance matrix of a random vector x is defined as $\mathbb{E}[xx^\top]$. To make the link, let the distribution of x be uniform over the rows of X.

The eigenvalues of C are $\lambda_1 = 2.8671$ and $\lambda_2 = 1.0466$, with the corresponding eigenvectors

$$v_1 = \begin{pmatrix} -0.9001 \\ 0.4356 \end{pmatrix} \quad \text{and} \quad v_2 = \begin{pmatrix} -0.4356 \\ -0.9001 \end{pmatrix}.$$

Note that you can also define the covariance matrix as $C = \frac{1}{n-1} X^\top X$ or $C = X^\top X$. The contours of $\mathcal{L}$ should be ellipses centered around

θ^* with axes corresponding to the eigenvectors. The larger eigenvector λ_1 should correspond to the minor axis and the smaller eigenvector λ_2 to the major axis.

The contour plot of $\mathcal{L}(\theta)$ is given in Figure 1, along with the trajectory of the LMS algorithm from part (c).

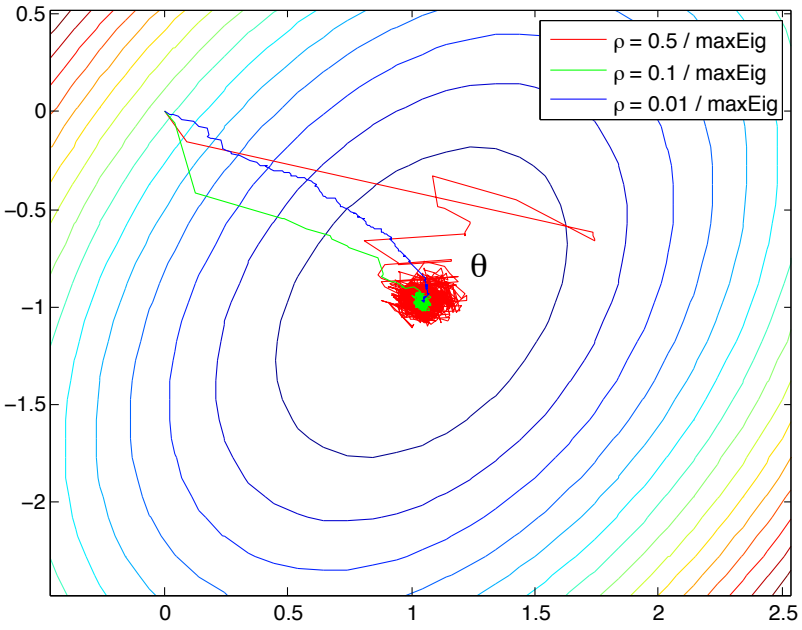


Figure 1: The LMS trajectory with learning rates ρ that depend on the maximum eigenvalue of the covariance matrix.

(c) Initializing the LMS algorithm at $\theta = 0$ plot the path taken in the parameter space by the algorithm for three different values of the step size ρ. In particular let ρ equal the inverse of the maximum eigenvalue of the covariance matrix, one-half of that value, and one-quarter of that value.

Solution: LMS is an online algorithm: at each iteration we pick a point (x_i, y_i) and make the update

$$\theta \leftarrow \theta + \rho(y_i - \theta^\top x_i)x_i.$$

To improve performance, it is generally advisable to choose a random ordering rather than go through the points in order.

Note that it may take many iterations for θ to approach θ^*. Even then, LMS is not guaranteed to converge at all, and in general, will not converge. The following batch update (which corresponds to gradient descent on $\mathcal{L}$):

$$\theta \leftarrow \theta + \rho \sum_{i=1}^{n} (y_i - \theta^\top x_i) x_i$$

does converge given an appropriate step size ρ.

For larger ρ, the algorithm takes bigger steps in the parameter space but tends to overshoot and is quite noisy. For smaller ρ, the algorithm takes smaller steps but is more stable. In practice, decreasing the step size ρ over time and monitoring the progress on the objective $\mathcal{L}$ is a good strategy.

Figure 1 shows the trajectory of the LMS algorithm for several different values of ρ. Note that the maximum eigenvalue in this case is small since we use the normalized covariance matrix, so we choose smaller values of ρ than what is asked in the problem in order to better illustrate the behavior of the algorithm.

The code that implements this problem is as follows.

```
%% CS 281A/Stat241A Homework 2.1 demo code

%% Part (a)

% Load dataset
load lms.dat
X = lms(:,1:2);
y = lms(:,3);
n = length(y);

% Solve normal equation
theta = (X'*X)\(X'*y);

%% Part (b)

% Find eigenvalues and eigenvectors of the covariance matrix
[V,D] = eig(X'*X/n);

% Create mesh grid centered at theta
[A,B] = meshgrid((theta(1)-1.5):.1:(theta(1)+1.5), ...
                 (theta(2)-1.5):.1:(theta(2)+1.5));

% Compute the objective function at each grid point
```

```
J = zeros(size(A));
for p = 1:size(A,1)
    for q = 1:size(A,2)
        aux = y-X*[A(p,q); B(p,q)];
        J(p,q) = 0.5*(aux'*aux);
    end
end

% Plot contour map
figure; contour(A,B,J,15); hold on;

% Also show the optimal theta from part (a)
plot(theta(1), theta(2), 'kx', 'LineWidth', 3);
text(theta(1)+0.2, theta(2)+0.2, '\theta', 'FontSize', 16);

%% Part (c)

% Find the maximum eigenvalue
maxEig = max(diag(D));

% Run the LMS algorithm with various parameters
path1 = LMS(X, y, [0;0], 0.5/maxEig);
path2 = LMS(X, y, [0;0], 0.1/maxEig);
path3 = LMS(X, y, [0;0], 0.01/maxEig);

% Plot of the trajectory paths
h1 = line(path1(1,:), path1(2,:), 'Color', 'r');
h2 = line(path2(1,:), path2(2,:), 'Color', 'g');
h3 = line(path3(1,:), path3(2,:), 'Color', 'b');
legend([h1 h2 h3], '\rho = 0.5 / maxEig', ...
    '\rho = 0.1 / maxEig', '\rho = 0.01 / maxEig');
```

```
function path = LMS(X, y, init, rho)

% Maximum number of iterations
maxIter = 1000;

% Place to store the updated parameters
path = zeros(size(X,2), maxIter+1);
path(:,1) = init;

% Initialize seed for random number generator
RandStream.setDefaultStream(RandStream(...
    'mt19937ar','seed',sum(100*clock)));

% Random indices for the data point to use at each iteration
ind = randi(size(X,1), maxIter, 1);
```

```
16
17 % Iterate through the random indices
18 for i = 1:maxIter
19     path(:,i+1) = path(:,i) + ...
20         rho*(y(ind(i))-path(:,i)'*X(ind(i),:)')*X(ind(i),:)';
21
22     % Check for convergence
23     if norm(path(:,i+1)-path(:,i)) < 1e-14
24         path = path(:,1:i+1);
25         fprintf('Converges after %d iterations!\n', i);
26         break;
27     end
28 end
```

Problem 2.2

The course website contains a data set `classification2d.dat` of (x_i, y_i) pairs, where the x_i are 2-dimensional vectors and y_i is a binary label.

(a) Plot the data, using 0's and X's for the two classes. The plots in the following parts should be plotted on top of this plot.

Solution: All the plots for this problem are given in Figure 2, including the classification boundaries from parts (b) and (c), for both the train and test datasets.

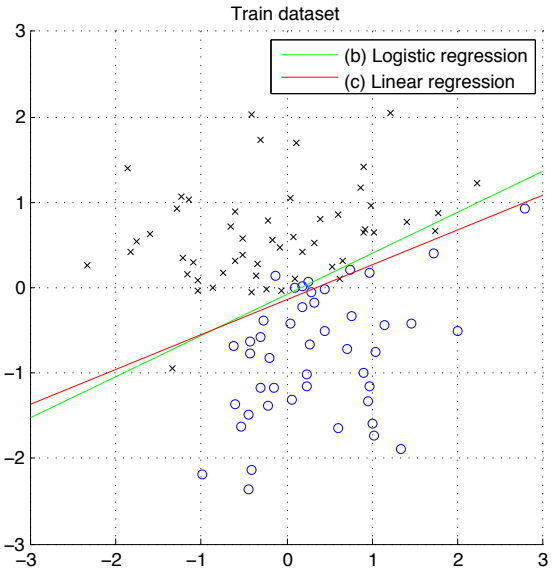
(b) Write a program to fit a logistic regression model using stochastic gradient ascent (or IRLS if you prefer). Plot the line where the logistic function is equal to 0.5.

Solution: The logistic regression model is

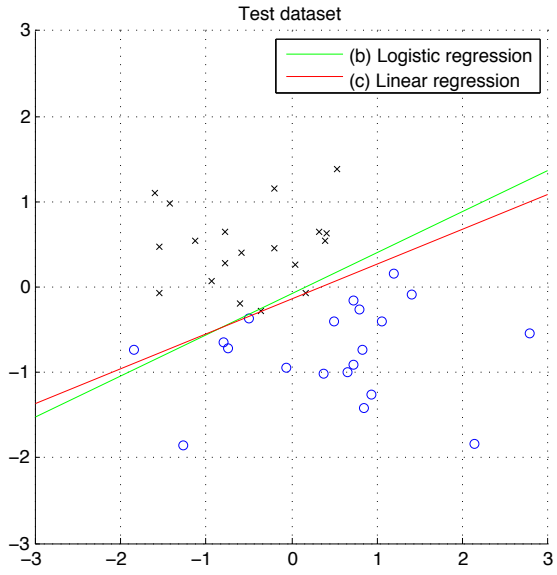
$$p(y = 1 \mid x) = \frac{1}{1 + \exp(-\theta^\top x - b)},$$

where θ is a parameter and b is a bias. To account for this bias term, we consider the modified matrix $\widetilde{X}$ consisting of the appended data points $\widetilde{x}_i = (x_i, 1)$, and the modified parameter $\widetilde{\theta} = (\theta, b)$. We estimate $\widetilde{\theta}$ using the stochastic gradient update algorithm given in Eq. (7.65) in Chapter 7, with initial value $\widetilde{\theta}^{(0)} = (0, 0, 0)$ and step size $\rho^{(t)} = 1/t$ (following the Robbins-Monro algorithm). We let the algorithm run for 50,000 iterations, where at each iteration we select a data point at random and update $\widetilde{\theta}$ using that data point, and in the end, we find the estimate for $\widetilde{\theta}$:

$$\widetilde{\theta} = \begin{pmatrix} -0.8123 \\ 1.6836 \\ 0.1398 \end{pmatrix}.$$



(a) Train dataset



(b) Test dataset

Figure 2: Plot of the train and test datasets, along with the classification boundaries.

The contour $p(y = 1 \mid x) = 1/2$ is given by the line $\theta^\top x + b = 0$, or equivalently,

$$-0.8123\, x_1 + 1.6836\, x_2 + 0.1398 = 0.$$

(c) Fit a linear regression to the problem, treating the class labels as real values 0 and 1. (You can solve the linear regression in any way you'd like, including solving the normal equations, using the LMS algorithm, or calling the built-in routines in Matlab or R). Plot the line where the linear regression function is equal to 0.5.

Solution: The linear regression model is

$$y = \theta_{\mathrm{LR}}^\top x + b,$$

where b is a bias term. As in part **(c)**, we consider the modified matrix $\widetilde{X}$ consisting of the appended data points $\widetilde{x}_i = (x_i, 1)$, and the modified parameter $\widetilde{\theta}_{\mathrm{LR}} = (\theta_{\mathrm{LR}}, b)$. Then $\widetilde{\theta}_{\mathrm{LR}}$ still satisfies the normal equation

$$\widetilde{X}^\top \widetilde{X} \widetilde{\theta}_{\mathrm{LR}} = \widetilde{X}^\top y \quad \Rightarrow \quad \widetilde{\theta}_{\mathrm{LR}} = (\widetilde{X}^\top \widetilde{X})^{-1} \widetilde{X}^\top y = \begin{pmatrix} -0.1534 \\ 0.3736 \\ 0.5527 \end{pmatrix}.$$

So the linear regression model is

$$y = -0.1534\, x_1 + 0.3736\, x_2 + 0.5527,$$

and the contour $y = 1/2$ is given by the line

$$-0.1534\, x_1 + 0.3736\, x_2 + 0.0527 = 0.$$

(d) The data set `testing.dat` is a separate data set generated from the same source. Test your fits from the previous parts on these data and compare the results.

Solution: For each of the classifiers from part (b) and (c), we compute the classification function $f(x)$ on the test dataset, and predict the label $y = 1$ if $f(x) \geq 0.5$, and $y = 0$ otherwise. We then compute the error of each classifier (i.e. the fraction of misclassified points). We find that logistic regression gives a 0.075 error rate, and linear regression gives a 0.050 error rate. Thus, linear regression yields a slightly better performance. Figure 2(b) shows the plot of the test dataset and the contour lines when the classification functions are equal to 1/2.

The code that implements this problem is as follows.

```
%% CS 281A/Stat241A Homework 2.2 demo code

%% Part (a)

% Load data
load classification2d.dat;
X = classification2d(:, 1:2);
y = classification2d(:, 3);

% Plot data
figure;
scatter(X(y==0,1), X(y==0,2), 'bo');
hold on;
scatter(X(y==1,1), X(y==1,2), 'kx');
axis square; grid;
title('Train dataset ');

%% Part (b)

% Append a constant term in the data to account for bias term
X_app = [X ones(size(X,1),1)];

% Compute theta using stochastic gradient algorithm
theta_log = SGA(X_app, y, [0;0;0]);

% Plot the line where logistic function is equal to 0.5
xplot = [-3 3];
hb = line(xplot, ...
    -theta_log(3)/theta_log(2)-(theta_log(1)/theta_log(2))*xplot, ...
    'Color', 'g');

%% Part (c)

% Solve normal equation for linear regression
theta_LR = (X_app'*X_app)\(X_app'*y);

% Plot the line where linear regression is equal to 0.5
hc = line(xplot, ...
    (0.5-theta_LR(3))/theta_LR(2)-(theta_LR(1)/theta_LR(2))*xplot, ...
    'Color', 'r');
axis([-3 3 -3 3]);
legend([hb hc], ...
    '(b) Logistic regression', '(c) Linear regression');

%% Part (d)

```

```
% Load test data
load testing.dat;
X_test = testing(:, 1:2);
y_test = testing(:, 3);
X_test_app = [X_test ones(size(X_test,1),1)];

% Plot test data
figure;
scatter(X_test(y_test==0,1), X_test(y_test==0,2), 'bo');
hold on;
scatter(X_test(y_test==1,1), X_test(y_test==1,2), 'kx');
axis square; grid;
title('Test dataset ');

% Plot contour lines for each classifier on the test dataset
hb = line(xplot, ...
    -theta_log(3)/theta_log(2)-(theta_log(1)/theta_log(2))*xplot, ...
    'Color', 'g');
hc = line(xplot, ...
    (0.5-theta_LR(3))/theta_LR(2)-(theta_LR(1)/theta_LR(2))*xplot, ...
    'Color', 'r');
axis([-3 3 -3 3]);
legend([hb hc], ...
    '(b) Logistic regression', '(c) Linear regression');

% Compute classification function for each classifier
aux_b = 1./(1+exp(-X_test_app*theta_log));
aux_c = X_test_app*theta_LR;

% Then compute classification prediction for each classifier
pred_b = (aux_b ≥ 0.5);
pred_c = (aux_c ≥ 0.5);

% Finally, compute the misclassification error
err_b = sum(pred_b ≠ y_test)/length(y_test);
err_c = sum(pred_c ≠ y_test)/length(y_test);
```

```
function theta = SGA(X, y, init)

% Maximum number of iterations
maxIter = 50000;

% Current theta parameter
theta = init;

% Initialize seed for random number generator
RandStream.setDefaultStream(...
```

```
    RandStream('mt19937ar','seed',sum(100*clock)));

% Random indices for the data point to use at each iteration
ind = randi(size(X,1), maxIter, 1);

% Iterate through the random indices
for i = 1:maxIter
    Ui = 1/(1+exp(-theta'*X(ind(i),:)'));
    theta = theta + (1/i) * (y(ind(i))-Ui) * X(ind(i),:)';
end
```

Problem 2.3

The ridge regression estimate is defined as

$$\widehat{\theta} \in \arg\min_{\theta \in \mathbb{R}^d} \left\{ \|y - X\theta\|_2^2 + \lambda_n \|\theta\|_2^2 \right\}$$

where $\lambda_n > 0$ is a positive regularization weight.

(a) Can the ridge regression problem have multiple optimal solutions? Why or why not? Justify your answer.

Solution: No, ridge regression has a unique optimal solution because the objective function is *strictly* convex due to the presence of the term $\lambda_n \|\theta\|_2^2$. Note that merely stating that the objective function is convex is not enough, as a convex function can have multiple minimizers (e.g. a constant function).

(b) In a Bayesian model, the parameter θ is viewed as random, and equipped with a prior distribution π. The maximum a posteriori (MAP) estimate is obtained by maximizing the function $f(\theta) := \mathbb{P}(y \mid X, \theta)\, \pi(\theta)$. Explain how the ridge regression estimate can be recovered as a MAP estimate.

Solution: Let $\theta \sim \mathcal{N}(0, (1/\lambda_n)I)$, and conditioned on X and θ, let $y_1, \ldots, y_n$ be independent with $y_i \mid X_i, \theta \sim \mathcal{N}(X_i^\top \theta, 1)$. Then the

MAP estimate of θ given X and y is

$$\begin{aligned}
\theta_{\text{MAP}} &= \arg\max_{\theta \in \mathbb{R}^d} \; p(\theta \mid X, y) \\
&= \arg\max_{\theta \in \mathbb{R}^d} \; p(y \mid X, \theta)\, \pi(\theta) \\
&= \arg\max_{\theta \in \mathbb{R}^d} \; \left[\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\big(y_i - X_i^\top \theta\big)^2 \Big) \right] \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{\lambda_n}{2} \|\theta\|_2^2 \right) \\
&= \arg\max_{\theta \in \mathbb{R}^d} \; \frac{1}{(2\pi)^{\frac{n+d}{2}}} \exp\left(-\frac{1}{2}\|y - X\theta\|_2^2 - \frac{\lambda_n}{2}\|\theta\|_2^2 \right) \\
&= \arg\min_{\theta \in \mathbb{R}^d} \; \|y - X\theta\|_2^2 + \lambda_n \|\theta\|_2^2 \\
&= \widehat{\theta}.
\end{aligned}$$

(c) Suppose that the matrix X is orthonormal. Give an explicit and easily computed expression for the ridge regression solution as a function (y, X, λ_n).

Solution: Taking derivative of the ridge regression objective function and setting is to zero gives us

$$(X^\top X + \lambda_n I)\widehat{\theta} = X^\top y,$$

so $\widehat{\theta} = (X^\top X + \lambda_n I)^{-1} X^\top y$. When X is orthonormal, i.e. $X^\top X = I$, this reduces to

$$\widehat{\theta} = \frac{1}{1 + \lambda_n} X^\top y.$$

As an aside, note that the matrix $X^\top X + \lambda_n I$ is strictly positive definite, so it is invertible. This is because $X^\top X$ is positive semidefinite, so it has all nonnegative eigenvalues, and adding $\lambda_n I$ simply shifts all the eigenvalues up by λ_n.

(d) If we replace the quantity $\|\theta\|_2^2$ with the ℓ_1-norm $\|\theta\|_1 = \sum_{j=1}^d |\theta_j|$, the resulting estimator is known as the Lasso. Assuming that X is orthonormal, give an explicit and easily computed expression for the Lasso solution as a function of (y, X, λ_n).

Solution: When X is orthonormal, we can write the Lasso objective

function as

$$\begin{aligned}\mathcal{L}(\theta) &= \|y - X\theta\|_2^2 + \lambda_n\|\theta\|_1 \\ &= \|y\|_2^2 - 2(X^\top y)^\top\theta + \|\theta\|_2^2 + \lambda_n\|\theta\|_1 \\ &= \sum_{i=1}^{n} \big(y_i^2 - 2(X^\top y)_i\theta_i + \theta_i^2 + \lambda_n|\theta_i|\big).\end{aligned}$$

Thus, minimizing $\mathcal{L}(\theta)$ is equivalent to minimizing each function

$$\mathcal{L}_i(\theta_i) = y_i^2 - 2(X^\top y)_i\theta_i + \theta_i^2 + \lambda_n|\theta_i|.$$

Note that we can also write

$$\mathcal{L}_i(\theta_i) = \max\{\mathcal{L}_i^+(\theta_i), \mathcal{L}_i^-(\theta_i)\} = \mathcal{L}_i^+(\theta_i)\,\mathbf{1}_{\{\theta_i \geq 0\}} + \mathcal{L}_i^-(\theta_i)\,\mathbf{1}_{\{\theta_i < 0\}},$$

where

$$\mathcal{L}_i^+(\theta_i) = y_i^2 - 2(X^\top y)_i\theta_i + \theta_i^2 + \lambda_n\theta_i$$

and

$$\mathcal{L}_i^-(\theta_i) = y_i^2 - 2(X^\top y)_i\theta_i + \theta_i^2 - \lambda_n\theta_i.$$

The function $\mathcal{L}_i^+$ is minimized by $\theta_i^+ = (X^\top y)_i - \lambda_n/2$, which is nonnegative when $(X^\top y)_i \geq \lambda_n/2$, in which case θ_i^+ also minimizes $\mathcal{L}_i(\theta_i)$. Similarly, $\mathcal{L}_i^-$ is minimized by $\theta_i^- = (X^\top y)_i + \lambda_n/2$, which is nonpositive when $(X^\top y)_i \leq -\lambda_n/2$, in which case θ_i^- also minimizes $\mathcal{L}_i(\theta_i)$. When $|(X^\top y)_i| < \lambda_n/2$, the function $\mathcal{L}_i$ is increasing for $\theta \geq 0$ and decreasing for $\theta \leq 0$, so the minimum is achieved at $\theta_i = 0$. Combining all the cases above, we can write the optimal θ_i as

$$\theta_i^{\text{Lasso}} = \text{sign}((X^\top y)_i)\,\Big(\big|(X^\top y)_i\big| - \frac{\lambda_n}{2}\Big)_+.$$

(e) Based on parts (c) and (d), which estimator (ridge or Lasso) is likely to lead to a sparser solution? Explain. (*Note:* A vector is sparse if it has a relatively small number $s \ll d$ of non-zero components.)

Solution: From part (c) we see that for the ridge regression estimator, $\widehat{\theta}_i = 0$ if and only if $(X^\top y)_i = 0$. From part (d) we see that for the Lasso estimator, $\theta_i^{\text{Lasso}} = 0$ if and only if $|(X^\top y)_i| \leq \lambda_n/2$. Thus the Lasso estimator is sparser.

Problem 2.4
Recall that a probability distribution in the exponential family takes the form

$$p(x;\eta) = h(x)\exp\{\eta^T T(x) - A(\eta)\}$$

for a parameter vector η, often referred to as the *natural parameter*, and for given functions T, A, and h.

(a) Determine which of the following distributions are in the exponential family, exhibiting the T, A, and h functions for those that are.

(i) $N(\mu, I)$—multivariate Gaussian with mean vector μ and identity covariance matrix.

Solution: The density for a d-dimensional Gaussian with mean μ and covariance matrix I is

$$\begin{aligned} p(x) &= \frac{1}{(2\pi)^{d/2}}\exp\Big(-\frac{1}{2}\|x-\mu\|_2^2\Big) \\ &= \frac{1}{(2\pi)^{d/2}}\exp\Big(-\frac{1}{2}x^\top x + \mu^\top x - \frac{1}{2}\mu^\top\mu\Big), \end{aligned}$$

so we have an exponential family with parameters

$$\begin{aligned} h(x) &= \frac{1}{(2\pi)^{d/2}}\exp\Big(-\frac{1}{2}x^\top x\Big), \\ T(x) &= x, \\ \eta &= \mu, \\ A(\eta) &= \frac{1}{2}\eta^T\eta. \end{aligned}$$

(ii) $\mathrm{Dir}(\alpha)$—Dirichlet with parameter vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$.
Solution: The Dirichlet density for $\theta \in \mathbb{R}^K$ is

$$p(\theta) \;=\; \frac{1}{B(\alpha)}\prod_{i=1}^K \theta_i^{\alpha_i - 1} \;=\; \prod_{i=1}^K \frac{1}{\theta_i}\exp\left(\sum_{i=1}^K \alpha_i \log\theta_i - \log B(\alpha)\right),$$

where $B(\alpha) = \frac{\prod_{i=1}^K \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^K \alpha_i)}$. Hence, we have an exponential family

distribution with parameters

$$\begin{aligned} h(\theta) &= \prod_{i=1}^{K} \frac{1}{\theta_i}, \\ T(\theta) &= \begin{pmatrix} \log(\theta_1) & \cdots & \log(\theta_K) \end{pmatrix}^\top, \\ \eta &= \alpha, \\ A(\eta) &= \log B(\eta). \end{aligned}$$

(iii) Mult(θ)—multinomial with parameter vector $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$. Use the fact that $\theta_K = 1 - \sum_{k=1}^{K-1} \theta_k$ and express the distribution using a $(K-1)$-dimensional parameter η.

Solution: We assume the number of trials is fixed at n. When $\sum_{i=1}^{K} x_i = n$, the density is

$$\begin{aligned} p(x) &= \binom{n}{x_1, x_2, \ldots, x_K} \prod_{i=1}^{K} \theta_i^{x_i} \\ &= \binom{n}{x_1, x_2, \ldots, x_K} \exp\left(\sum_{i=1}^{K} x_i \log \theta_i \right) \\ &= \binom{n}{x_1, x_2, \ldots, x_K} \exp\left(\sum_{i=1}^{K-1} x_i \log \theta_i + \Big(n - \sum_{i=1}^{K-1} x_i\Big) \log \theta_K \right) \\ &= \binom{n}{x_1, x_2, \ldots, x_K} \exp\left(\sum_{i=1}^{K-1} x_i (\log \theta_i - \log \theta_K) + n \log \theta_K \right). \end{aligned}$$

For $i = 1, \ldots, K$, take

$$\eta_i = \log \theta_i - \log \theta_K = \log \frac{\theta_i}{\theta_K}.$$

Note that

$$1 = \sum_{i=1}^{K} \theta_i = \theta_K \sum_{i=1}^{K} e^{\eta_i},$$

so

$$\theta_K = \left(\sum_{i=1}^{K} e^{\eta_i} \right)^{-1} = \left(1 + \sum_{i=1}^{K-1} e^{\eta_i} \right)^{-1}.$$

Hence, we have an exponential family with parameters

$$\begin{aligned} h(x) &= \mathbf{1}_{\{\sum_{i=1}^K x_i = n\}} \binom{n}{x_1, x_2, \ldots, x_K}, \\ T(x) &= \begin{pmatrix} x_1 & \cdots & x_{K-1} \end{pmatrix}^\top, \\ \eta &= \begin{pmatrix} \eta_1 & \cdots & \eta_{K-1} \end{pmatrix}^\top, \\ A(\eta) &= -n \log \theta_K = n \log \left(1 + \sum_{i=1}^{K-1} e^{\eta_i} \right). \end{aligned}$$

(iv) the uniform distribution over the interval $[0, \eta]$.
Solution: This is not in the exponential family, since the support of this distribution depends on the parameter η.

(v) the log normal distribution: the distribution of $Y = \exp(X)$, where $X \sim N(0, \sigma^2)$.
Solution: The log normal density has the form

$$\begin{aligned} p(y) &= \frac{1}{y\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log y)^2}{2\sigma^2} \right) \\ &= \frac{1}{y\sqrt{2\pi}} \exp\left(\frac{-1}{2\sigma^2} (\log y)^2 - \frac{1}{2} \log\left(\sigma^2\right) \right). \end{aligned}$$

Hence, the log normal distribution is in the exponential family with

$$\begin{aligned} h(y) &= \frac{1}{y\sqrt{2\pi}}, \\ T(y) &= (\log y)^2, \\ \eta &= -\frac{1}{2\sigma^2}, \\ A(\eta) &= -\frac{1}{2} \log(-2\eta). \end{aligned}$$

(b) Recall that the function $A(\eta)$ has moment-generating properties—in particular, $\nabla_\eta A(\eta) = \mathbb{E}[T(X)]$. Demonstrate that this relationship holds for those examples that are in the exponential family in part (a).

(i) Normal:

$$\nabla_\eta A(\eta) = \nabla_\eta \left(\frac{1}{2} \eta^\top \eta \right) = \eta = \mu = \mathbb{E}[X] = \mathbb{E}[T(X)].$$

(ii) Dirichlet:

For this distribution, we employ a general exponential family argument to derive the desired result:

$$1 = \int h(x) \exp\Big(\eta^\top T(x) - A(\eta)\Big),$$

so

$$\begin{aligned} 0 &= \nabla_\eta \int h(x) \exp\Big(\eta^\top T(x) - A(\eta)\Big)\, dx \\ &= \int \nabla_\eta \big\{ h(x) \exp\Big(\eta^\top T(x) - A(\eta)\Big) \big\} dx \\ &= \int \big(T(x) - \nabla_\eta A(\eta)\big)\, h(x)\, \exp\Big(\eta^\top T(x) - A(\eta)\Big)\, dx \\ &= \mathbb{E}[T(X)] - \nabla_\eta A(\eta), \end{aligned}$$

implying that

$$\mathbb{E}[T(X)] = \nabla_\eta A(\eta).$$

Note that we exchange the order of integration and differentiation. There are cases in which this exchange is not valid. See Appendix A.9 in "Probability: Theory and Examples" by Durrett for sufficient conditions under which differentiation and integration can be exchanged.

One may also solve this by using the definition that $A(\eta)$ is the function which makes the density integrate to 1, that is,

$$A(\eta) = \log \int h(x) \exp(\eta^T T(x)).$$

(iii) Multinomial:

For each $1 \le i \le K-1$,

$$\frac{\partial}{\partial \eta_i} \Big\{ n \log\Big(1 + \sum_{i=1}^{K-1} e^{\eta_i}\Big) \Big\} = n \frac{e^{\eta_i}}{1 + \sum_{i=1}^{K-1} e^{\eta_i}} = n\theta_i = \mathbb{E}[X_i].$$

(v) Log normal:

Note that $\mathbb{E}[(\log Y)^2] = \mathbb{E}[X^2] = \sigma^2$, since $Y = \exp(X)$ for $X \sim N(0, \sigma^2)$. Furthermore,

$$\frac{d}{d\eta} A(\eta) = \frac{d}{d\eta} \Big\{ -\frac{1}{2} \log(-2\eta) \Big\} = \frac{-1}{2\eta} = \sigma^2.$$

Problem 2.5

(ML/entropy, conjugacy and duality): Given a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the dual function is a new function $f^* : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, defined as follows:

$$f^*(v) \quad = \quad \sup_{u \in \mathbb{R}^n} \big\{ v^T u - f(u) \big\}. \tag{1}$$

(Note that the supremum can be $+\infty$ for some $v \in \mathbb{R}^n$.)

(a) Given the cumulant generating function $A(\theta) = \log[1 + \exp(\theta)]$, for a Bernoulli variable, compute the dual function A^*. What is the link between this computation and maximum likelihood estimation? How is A^* related to Bernoulli entropy? Compute the double dual A^{**}, and verify that $A^{**} = A$. How is computing A^{**} related to maximum entropy?

Solution: The dual function is $A^*(v) = \sup_{u \in \mathbb{R}} \{uv - \log(1 + \exp(u)\}$. The inner expression is a concave function of u, so we can maximize it by setting its derivative to zero:

$$v - \frac{\exp(u^*)}{1 + \exp(u^*)} = 0,$$

from which we get $u^* = \log(\frac{v}{1-v})$. Note that this is valid only for $0 < v < 1$. Plugging this value back to the definition, we obtain

$$A^*(v) = u^* v - \log(1 + \exp(u^*)) = v \log v + (1 - v) \log(1 - v)$$

for $0 < v < 1$. It is easy to see that $A^*(v) = \infty$ when $v < 0$ (by taking $u \to -\infty$) or $v > 1$ (by taking $u \to \infty$). Moreover, we can also show that $A^*(v) = 0$ for $v \in \{0, 1\}$.

The computation of $A^*(v)$ is precisely the process of computing the MLE of u when v is a sample drawn from the Bernoulli(u) distribution. Moreover, for $0 < v < 1$, $A^*(v)$ is equal to (-1) times the entropy of the Bernoulli(v) distribution.

To compute the double dual $A^{**}(u) = \sup_{v \in \mathbb{R}} \{vu - A^*(v)\}$, it suffices to restrict the domain of v to $[0, 1]$, for otherwise we have $-A^*(v) = -\infty$. By taking derivative of the inner expression and setting it to zero, we obtain

$$u = \log v^* - \log(1 - v^*),$$

which gives us

$$v^* = \frac{\exp(u)}{1+\exp(u)}.$$

Plugging this value back to the definition of A^{**}, we get

$$A^{**}(u) = v^*u - A^*(v^*) = \log(1+\exp(u)) = A(u).$$

In particular, $A^{**}(0)$ is computing the maximum entropy of the Bernoulli distribution.

(b) Using the definition (1), prove that the dual function f^* is always convex: i.e., for all $\lambda \in [0,1]$, $v, v' \in \mathbb{R}^n$, $f^*\big(\lambda v + (1-\lambda)v'\big) \le \lambda f^*(v) + (1-\lambda)f^*(v')$.

Solution: We have

$$\begin{aligned}
f^*(\lambda v + (1-\lambda)v') &= \sup_{u\in\mathbb{R}^n} \big\{ \big(\lambda v + (1-\lambda)v'\big)^\top u - f(u)\big\} \\
&= \sup_{u\in\mathbb{R}^n} \big\{\lambda(v^\top u - f(u)) + (1-\lambda)(v'^\top u - f(u))\big\} \\
&\le \lambda \sup_{u\in\mathbb{R}^n} \{v^\top u - f(u)\} + (1-\lambda) \sup_{u\in\mathbb{R}^n} \{v'^\top u - f(u)\} \\
&= \lambda f^*(v) + (1-\lambda) f^*(v').
\end{aligned}$$

(c) Given a function f, assume that it is differentiable on $\mathbb{R}^n$, and that it satisfies the duality relation $f^{**} = f$. Use definition (1) for f^* and $f^{**} = f$ to prove that $f(u) \ge f(w) + \nabla f(w)^T(u-w)$ for all $u, w \in \mathbb{R}^n$.

Solution: Since f is differentiable, we can take gradient and set it to zero in computing $f^*(v)$, giving us the first-order optimality condition $v = \nabla f(u^*)$. Thus, for each $u \in \mathbb{R}^n$ we have

$$f^*(\nabla f(u)) = u^\top \nabla f(u) - f(u).$$

Now for any $u, w \in \mathbb{R}^n$, since $f^{**} = f$,

$$\begin{aligned}
f(u) = f^{**}(u) &= \sup_{v\in\mathbb{R}^n} \{u^\top v - f^*(v)\} \\
&\ge u^\top \nabla f(w) - f^*(\nabla f(w)) \\
&= u^\top \nabla f(w) - w^\top \nabla f(w) + f(w) \\
&= f(w) + \nabla f(w)^\top (u-w).
\end{aligned}$$

Note: Some students wrote that the desired inequality is equivalent to the convexity of f, which follows from part (b) and the assumption

that $f^{**} = f$. While this is true, we want you to prove the inequality from first principles.

Hint: Each of parts (b) and (c) require proofs, but the arguments need not be very long.

Problem 2.6

Maximum entropy and exponential families For a discrete random variable $X \in \mathcal{X}$ with distribution $p(\cdot)$, the (Boltzmann-Shannon) entropy is given by $H(p) := -\sum_{x \in \mathcal{X}} p(x) \log p(x)$. (We assume that $0 \log 0 = 0$ in this expression). The entropy is a measure of the uncertainty associated with X. Although entropy can be defined more generally, for this problem assume that $|\mathcal{X}|$ is finite.

(a) Suppose that we are given a set of expectation constraints on $p(\cdot)$, say of the form $\sum_{x \in \mathcal{X}} p(x) T_\alpha(x) = \mu_\alpha$ for a collection of functions $\{T_1, T_2, \ldots, T_D\}$. (In practice, these constraints would be imposed by making observations.) Consider the maximum entropy problem of maximizing $H(p)$ subject to these expectation constraints, the non-negativity condition $p(x) \geq 0$ for all $x \in \mathcal{X}$, and the normalization constraint $\sum_{x \in \mathcal{X}} p(x) = 1$. Write out the Lagrangian associated with this constrained optimization problem.

Solution: The optimization variables are $p = \{p(x) \colon x \in \mathcal{X}\}$. Let λ_0 denote the Lagrange multiplier for the normalization constraint, and for $1 \leq \alpha \leq D$, let λ_α denote the Lagrange multiplier for the expectation constraint for T_α. Then the Lagrangian associated with this optimization problem is

$$\begin{aligned}\Lambda(p, \lambda) = &- \sum_{x \in \mathcal{X}} p(x) \log p(x) + \lambda_0 \Big(\sum_{x \in \mathcal{X}} p(x) - 1 \Big) \\ &+ \sum_{\alpha=1}^{D} \lambda_\alpha \Big(\sum_{x \in \mathcal{X}} p(x) T_\alpha(x) - \mu_\alpha \Big),\end{aligned}$$

where $p \in \mathbb{R}^{|\mathcal{X}|}_{\geq 0}$ and $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_D) \in \mathbb{R}^{D+1}$.

(b) By computing stationary points of the Lagrangian, show that the optimal solution $\widehat{p}$ takes the form of an exponential family.

Solution: The stationary points of the Lagrangian satisfy $\nabla_{p,\lambda} \Lambda = 0$. The partial derivative of Λ with respect to each $p(x)$ is

$$\frac{\partial \Lambda}{\partial p(x)} = -\log p(x) - 1 + \lambda_0 + \sum_{\alpha=1}^{D} \lambda_\alpha T_\alpha(x),$$

and upon setting this equal to zero, we obtain

$$\widehat{p}(x) = \exp\left(-1 + \lambda_0 + \sum_{\alpha=1}^{D} \lambda_\alpha T_\alpha(x)\right).$$

Hence we see that the optimal solution $\widehat{p}$ must be in the exponential family, where the parameters λ are chosen to satisfy the expectation constraints.