CS 229, Autumn 2012
Problem Set #3 Solutions: Theory & Unsupervised learning

Due in class (9:00am) on Wednesday, November 14.

Notes: (1) These questions require thought, but do not require long answers. Please be as concise as possible. (2) If you have a question about this homework, we encourage you to post your question on our Piazza forum, at `https://piazza.com/class#fall2012/cs229`. (3) If you missed the first lecture or are unfamiliar with the collaboration or honor code policy, please read the policy on Handout #1 (available from the course website) before starting work. (4) For problems that require programming, please include in your submission a printout of your code (with comments) and any figures that you are asked to plot. (5) Please indicate the submission time and number of late dates clearly in your submission.

SCPD students: Please email your solutions to `cs229-qa@cs.stanford.edu` with the subject line "Problem Set 3 Submission". The first page of your submission should be the homework routing form, which can be found on the SCPD website. Your submission (including the routing form) must be a single pdf file, or we may not be able to grade it. If you are writing your solutions out by hand, please write clearly and in a reasonably large font using a dark pen to improve legibility.

1. **[23 points] Uniform convergence**

   You are hired by CNN to help design the sampling procedure for making their electoral predictions for the next presidential election in the (fictitious) country of Elbania.

   The country of Elbania is organized into states, and there are only two candidates running in this election: One from the Elbanian Democratic party, and another from the Labor Party of Elbania. The plan for making our electorial predictions is as follows: We'll sample m voters from each state, and ask whether they're voting democrat. We'll then publish, for each state, the estimated fraction of democrat voters. In this problem, we'll work out how many voters we need to sample in order to ensure that we get good predictions with high probability.

   One reasonable goal might be to set m large enough that, with high probability, we obtain uniformly accurate estimates of the fraction of democrat voters in every state. But this might require surveying very many people, which would be prohibitively expensive. So, we're instead going to demand only a slightly lower degree of accuracy.

   Specifically, we'll say that our prediction for a state is "highly inaccurate" if the estimated fraction of democrat voters differs from the actual fraction of democrat voters within that state by more than a tolerance factor γ. CNN knows that their viewers will tolerate some small number of states' estimates being highly inaccurate; however, their credibility would be damaged if they reported highly inaccurate estimates for too many states. So, rather than trying to ensure that all states' estimates are within γ of the true values (which would correspond to no state's estimate being highly inaccurate), we will instead try only to ensure that the number of states with highly inaccurate estimates is small.

To formalize the problem, let there be n states, and let m voters be drawn IID from each state. Let the actual fraction of voters in state i that voted democrat be ϕ_i. Also let X_{ij} $(1 \leq i \leq n, 1 \leq j \leq m)$ be a binary random variable indicating whether the j-th randomly chosen voter from state i voted democrat:

$$X_{ij} = \begin{cases} 1 & \text{if the } j^{th} \text{ example from the } i^{th} \text{ state voted democrat} \\ 0 & \text{otherwise} \end{cases}$$

We assume that the voters correctly disclose their vote during the survey. Thus, for each value of i, we have that X_{ij} are drawn IID from a Bernoulli(ϕ_i) distribution. Moreover, the X_{ij}'s (for all i, j) are all mutually independent.

After the survey, the fraction of democrat votes in state i is estimated as:

$$\hat{\phi}_i = \frac{1}{m} \sum_{j=1}^{m} X_{ij}$$

Also, let $Z_i = 1\{|\hat{\phi}_i - \phi_i| > \gamma\}$ be a binary random variable that indicates whether the prediction in state i was highly inaccurate.

(a) Let ψ_i be the probability that $Z_i = 1$. Using the Hoeffding inequality, find an upper bound on ψ_i.

Answer: A direct application of the Hoeffding inequality yields

$$\psi_i \leq 2e^{-2\gamma^2 m}$$

(b) In this part, we prove a general result which will be useful for this problem. Let V_i and W_i $(1 \leq i \leq k)$ be Bernoulli random variables, and suppose

$$\mathrm{E}[V_i] = P(V_i = 1) \leq P(W_i = 1) = \mathrm{E}[W_i] \qquad \forall i \in \{1, 2, \ldots k\}$$

Let the V_i's be mutually independent, and similarly let the W_i's also be mutually independent. Prove that, for any value of t, the following holds:

$$P\left(\sum_{i=1}^{k} V_i > t\right) \leq P\left(\sum_{i=1}^{k} W_i > t\right)$$

[Hint: One way to do this is via induction on k. If you use a proof by induction, for the base case ($k = 1$), you must show that the inequality holds for $t < 0$, $0 \leq t < 1$, and $t \geq 1$.]

Answer: Prove it by induction.

Base case: Show

$$P\left(V_1 > t\right) \leq P\left(W_1 > t\right)$$

If $t < 0$, then both probalities are 1. If $t \geq 1$, then both probabilities are 0. Otherwise, the equation reduces to

$$P\left(V_1 = 1\right) \leq P\left(W_1 = 1\right)$$

which holds by our original assumptions.
Inductive step: Assume

$$P\left(\sum_{i=1}^{l} V_i > t\right) \leq P\left(\sum_{i=1}^{l} W_i > t\right), \forall t$$

Then,

$$\begin{aligned}
&P\left(\sum_{i=1}^{l+1} V_i > t\right) \\
&\quad = P\left(V_{l+1}=1\right) P\left(\sum_{i=1}^{l+1} V_i > t \middle| V_{l+1}=1\right) + P\left(V_{l+1}=0\right) P\left(\sum_{i=1}^{l+1} V_i > t \middle| V_{l+1}=0\right) \\
&\quad = P\left(V_{l+1}=1\right) P\left(\sum_{i=1}^{l} V_i > t-1 \middle| V_{l+1}=1\right) + P\left(V_{l+1}=0\right) P\left(\sum_{i=1}^{l} V_i > t \middle| V_{l+1}=0\right) \\
&\quad = P\left(V_{l+1}=1\right) P\left(\sum_{i=1}^{l} V_i > t-1\right) + P\left(V_{l+1}=0\right) P\left(\sum_{i=1}^{l} V_i > t\right) \\
&\quad = P\left(V_{l+1}=1\right) P\left(\sum_{i=1}^{l} V_i > t-1\right) + \left(1 - P\left(V_{l+1}=1\right)\right) P\left(\sum_{i=1}^{l} V_i > t\right) \\
&\quad = P\left(V_{l+1}=1\right) \left(P\left(\sum_{i=1}^{l} V_i > t-1\right) - P\left(\sum_{i=1}^{l} V_i > t\right)\right) + P\left(\sum_{i=1}^{l} V_i > t\right) \\
&\quad \leq P\left(W_{l+1}=1\right) \left(P\left(\sum_{i=1}^{l} V_i > t-1\right) - P\left(\sum_{i=1}^{l} V_i > t\right)\right) + P\left(\sum_{i=1}^{l} V_i > t\right) \\
&\quad = P\left(W_{l+1}=1\right) P\left(\sum_{i=1}^{l} V_i > t-1\right) + \left(1 - P\left(W_{l+1}=1\right) P\left(\sum_{i=1}^{l} V_i > t\right)\right. \\
&\quad = P\left(W_{l+1}=1\right) P\left(\sum_{i=1}^{l} V_i > t-1\right) + P\left(W_{l+1}=0\right) P\left(\sum_{i=1}^{l} V_i > t\right) \\
&\quad \leq P\left(W_{l+1}=1\right) P\left(\sum_{i=1}^{l} W_i > t-1\right) + P\left(W_{l+1}=0\right) P\left(\sum_{i=1}^{l} W_i > t\right) \\
&\quad = P\left(\sum_{i=1}^{l+1} W_i > t\right).
\end{aligned}$$

And the result is proved.

(c) The fraction of states on which our predictions are highly inaccurate is given by $\overline{Z} = \frac{1}{n}\sum_{i=1}^{n} Z_i$. Prove a reasonable closed form upper bound on the probability $P(\overline{Z} > \tau)$ of being highly inaccurate on more than a fraction τ of the states.

[Note: There are many possible answers, but to be considered reasonable, your bound must decrease to zero as $m \to \infty$ (for fixed n and $\tau > 0$). Also, your bound should either remain constant or decrease as $n \to \infty$ (for fixed m and $\tau > 0$). It is also fine

if, for some values of τ, m and n, your bound just tells us that $P(\overline{Z} > \tau) \leq 1$ (the trivial bound).]

Answer: There are multiple ways to do this problem. We list a couple of them below:

Using Chernoff's inequality

Let Y_i be new Bernoulli random variables with mean $\mu = 2e^{-2\gamma^2 m}$. Then we know from part (a) that $P(Z_i = 1) \leq \mu = P(Y_i = 1)$. Using the result from the previous part:

$$\begin{aligned} P(\overline{Z} > \tau) &\leq P\left(\frac{1}{n}\sum_{i=0}^{n} Y_i > \tau\right) \\ &= P\left(\frac{1}{n}\sum_{i=0}^{n} Y_i - \mu > \tau - \mu\right) \\ &\leq P\left(\left|\frac{1}{n}\sum_{i=0}^{n} Y_i - \mu\right| > \tau - \mu\right) \\ &\leq 2\exp\left(-2(\tau-\mu)^2 n\right), \end{aligned}$$

where the last step follows provided that $0 < \tau - \mu = \tau - 2e^{-2\gamma^2 m}$, or equivalently, $m > \frac{1}{2\gamma^2}\log\left(\frac{2}{\tau}\right)$. For fixed τ and m, this bound goes to zero as $n \to \infty$. Alternatively, we can also just compute the right side directly, as in

$$\begin{aligned} P(\overline{Z} > \tau) &\leq P\left(\frac{1}{n}\sum_{i=0}^{n} Y_i > \tau\right) \\ &= P\left(\sum_{i=0}^{n} Y_i > n\tau\right) \\ &= \sum_{j=k}^{n} P\left(\sum_{i=0}^{n} Y_i = j\right) \\ &= \sum_{j=k}^{n} \binom{n}{j} \mu^j (1-\mu)^{1-j} \\ &\leq \sum_{j=k}^{n} \binom{n}{j} \mu^j \end{aligned}$$

where k is the smallest integer such that $k > n\tau$. For fixed τ and n, observe that as $m \to \infty$, $\mu \to 0$, so this bound goes to zero. Therefore,

$$P(\overline{Z} > \tau) \leq \min\left\{1, 2e^{-2(\tau-\mu)^2 n}, \sum_{j=k}^{n} \binom{n}{j} \mu^j\right\}$$

has the properties we want.

Using Markov's inequality

Markov's inequality states that for any nonnegative random variable X and $\tau > 0$, then

$P(X > \tau) \leq \frac{E[X]}{\tau}$. From part (a), we have $E[Z_i] = P(Z_i = 1) \leq 2e^{-2\gamma^2 m}$, implying that

$$\begin{aligned} P(\overline{Z} > \tau) &= P\left(\frac{1}{n}\sum_{i=0}^{n} Z_i > \tau\right) \\ &\leq \frac{E\left[\frac{1}{n}\sum_{i=0}^{n} Z_i\right]}{\tau} \\ &\leq \frac{2}{\tau} e^{-2\gamma^2 m}. \end{aligned}$$

This bound satisfies the given requirements: as $m \to \infty$, the bound goes to zero; if $n \to \infty$, the bound stays constant.

Using Chebyshev's inequality

Chebyshev's inequality states that for any random variable X with expected value μ and finite variance σ^2, then for any constant $\tau > 0$, $P(|X - \mu| > \tau) \leq \frac{\sigma^2}{\tau}$. Let Y_i be new Bernoulli random variables with mean $\mu = 2e^{-2\gamma^2 m}$. Then we know from part (a) that $P(Z_i = 1) \leq \mu = P(Y_i = 1)$. Using the result from the previous part:

$$\begin{aligned} P(\overline{Z} > \tau) &\leq P\left(\frac{1}{n}\sum_{i=0}^{n} Y_i > \tau\right) \\ &= P\left(\frac{1}{n}\sum_{i=0}^{n} Y_i - \mu > \tau - \mu\right) \\ &\leq P\left(\left|\frac{1}{n}\sum_{i=0}^{n} Y_i - \mu\right| > \tau - \mu\right) \\ &\leq \frac{\mathsf{Var}\left[\frac{1}{n}\sum_{i=0}^{n} Y_i\right]}{(\tau - \mu)^2} \\ &= \frac{\frac{1}{n^2}\sum_{i=0}^{n} \mathsf{Var}\left[Y_i\right]}{(\tau - \mu)^2} \\ &= \frac{2e^{-2\gamma^2 m}(1 - 2e^{-2\gamma^2 m})}{n(\tau - \mu)^2} \\ &\leq \frac{2e^{-2\gamma^2 m}}{n(\tau - \mu)^2}, \end{aligned}$$

where we again require that $m > \frac{1}{2\gamma^2}\log\left(\frac{2}{\tau}\right)$. This version of the bound goes to zero both when $m \to \infty$ and when $n \to \infty$.

2. **[15 points] More VC dimension**

Let the domain of the inputs for a learning problem be $\mathcal{X} = \mathbb{R}$. Consider using hypotheses of the following form:

$$h_\theta(x) = 1\{\theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_d x^d \geq 0\},$$

and let $\mathcal{H} = \{h_\theta : \theta \in \mathbb{R}^{d+1}\}$ be the corresponding hypothesis class. What is the VC dimension of $\mathcal{H}$? Justify your answer.

[Hint: You may use the fact that a polynomial of degree d has at most d real roots. When doing this problem, you should not assume any other non-trivial result (such as that the VC dimension of linear classifiers in d-dimensions is $d+1$) that was not formally proved in class.]

Answer: The key insight is that if the polynomial does not cross the x-axis (i.e. have a root) between two points, then it must give the two points the same label.

First, we need to show that there is a set of size $d+1$ which $\mathcal{H}$ can shatter. We consider polynomials with d real roots. A subset of the polynomials in $\mathcal{H}$ can be written as

$$\pm \prod_{i=1}^{d} (x - r_i)$$

where r_i is the i^{th} real root. Consider any set of size $d+1$ which does not contain any duplicate points. For any labelling of these points, construct a function as follows: If two consecutive points are labelled differently, set one of the r_i to the average of those points. If two consecutive points are labelled the same, don't put a root between them. If we haven't used up all of our d roots, place them beyond the last point. Finally, choose $\pm$ to get the desired labelling.

A more constructive proof of the above is the following: consider any set of distinct points $x^{(1)}, \ldots, x^{(d+1)}$, and let $y^{(1)}, \ldots, y^{(d+1)} \in \{-1, 1\}$ be any labeling of these points (where we have used -1 for points which would normally be labeled zero). Then, consider the following polynomial:

$$p(x) = \sum_{k=1}^{d+1} y^{(k)} \prod_{j \neq k} \left(\frac{x^{(j)} - x}{x^{(j)} - x^{(k)}} \right).$$

Here, observe that in the above expression, each term of the summation is a polynomial (in x) of degree d, and hence the overall expression is a polynomial of degree d. Furthermore, observe that when $x = x^{(i)}$, then the ith term of the summation evaluates to $y^{(i)}$, and all other terms of the summation evaluate to 0 (since all other terms have a factor $(x^{(i)} - x)$). Therefore, $p(x^{(i)}) = y^{(i)}$ for $i = 1, \ldots, d+1$. This construction is known as a "Lagrange interpolating polynomial." Therefore, any labeling of $d+1$ points can be realized using a degree d polynomial.

Second, we need to prove that $\mathcal{H}$ can't shatter a set of size $d+2$. If two points are identical, we can't realize any labelling that labels them differently. If all points are unique, we can't achieve an alternating labelling because we would need $d+1$ roots.

3. **[15 points] LOOCV and SVM**

   (a) **Linear Case.** Consider training an SVM using a linear Kernel $K(x, z) = x^T z$ on a training set $\{(x^{(i)}, y^{(i)}) : i = 1, \ldots, m\}$ that is linearly separable, and suppose we do not use ℓ_1 regularization. Let $|SV|$ be the number of support vectors obtained when training on the entire training set. (Recall $x^{(i)}$ is a support vector if and only if $\alpha_i > 0$.) Let $\hat{\varepsilon}_{\text{LOOCV}}$ denote the leave one out cross validation error of our SVM. Prove that

   $$\hat{\varepsilon}_{\text{LOOCV}} \leq \frac{|SV|}{m}.$$

   Answer: At a high level, the result is a consequence of the following claim (to be proven below): if $x^{(i)}$ is not a support vector when training on the entire training set,

then the optimal w and b does not change when leaving $x^{(i)}$ out of the training set. Since the original data are linearly separable and since we are using a hard-margin classifier, the hypothesis given by the original w and b will not make an error on $x^{(i)}$, and hence, no error will be made in the ith step of the LOOCV. Equivalently, the only possible errors in the LOOCV procedure are made on $x^{(i)}$'s which are support vectors when training on the entire training set, and hence $\hat{\varepsilon}_{\text{LOOCV}} \leq 1 - \frac{|non-SV|}{m} = \frac{|SV|}{m}$, and we are done.

To show the claim, let $S = \{(x^{(i)}, y^{(i)}) : i = 1, \ldots, m\}$. Let (w_S^*, b_S^*) and α_S^* denote the optimal primal and dual solutions for the SVM when training on S. Also, let $S_i = S \setminus \{(x^{(i)}, y^{(i)})\}$ be the set of training examples when omitting the ith example, and let (w_{S_i}, b_{S_i}), and α_{S_i} be the primal and dual variables of the optimization problem when training on S_i. Observe that α_{S_i} consists of only $m-1$ variables, which we'll denote as $\alpha_{S_i,1}, \ldots, \alpha_{S_i,i-1}, \alpha_{S_i,i+1}, \ldots, \alpha_{S_i.m}$.

If $x^{(i)}$ is not a support vector when training on S, then $\alpha_{S,i} = 0$. To show that w and b do not change when leaving out $(x^{(i)}, y^{(i)})$, consider the setting of dual variables $\alpha_{S_i,j}^* = \alpha_{S,j}^*$ for each $j \neq i$. Observe (w^*, b^*) and α_{S_i} satisfy the KKT conditions for the SVM optimization problem for training on S_i. In particular, the fact that the derivatives of the Lagrangian with respect to the primal variables hold is guaranteed by our construction of the dual problem. The remaining conditions (KKT dual complentarity, primal feasibility, and dual feasibility) follow from the KKT conditions for verifying that (w_S^*, b_S^*) and (α_S^*) is optimal when training on the entire set. From this (and the fact that w^* and b^* are unique since the objective function is strictly convex), we can conclude that w^* and b^* do not change when omitting $(x^{(i)}, y^{(i)})$, as desired.

(b) **General Case.** Consider a setting similar to in part (a), except that we now run an SVM using a general (Mercer) kernel. Assume that the data is linearly separable in the high dimensional feature space corresponding to the kernel. Does the bound in part (a) on $\hat{\varepsilon}_{\text{LOOCV}}$ still hold? Justify your answer.

Answer: Yes. The above argument only uses the facts that the optimum of a convex optimization problem is not affected by leaving out non-active constraints, and that the training data can be perfectly classified by the obtained hypothesis based on training on the full dataset. The choice of kernel has no influence.

4. **[12 points] MAP estimates and weight decay**

Consider using a logistic regression model $h_\theta(x) = g(\theta^T x)$ where g is the sigmoid function, and let a training set $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ be given as usual. The maximum likelihood estimate of the parameters θ is given by

$$\theta_{\text{ML}} = \arg\max_\theta \prod_{i=1}^m p(y^{(i)}|x^{(i)}; \theta).$$

If we wanted to regularize logistic regression, then we might put a Bayesian prior on the parameters. Suppose we chose the prior $\theta \sim \mathcal{N}(0, \tau^2 I)$ (here, $\tau > 0$, and I is the $n+1$-by-$n+1$ identity matrix), and then found the MAP estimate of θ as:

$$\theta_{\text{MAP}} = \arg\max_\theta p(\theta) \prod_{i=1}^m p(y^{(i)}|x^{(i)}, \theta)$$

Prove that

$$||\theta_{\text{MAP}}||_2 \leq ||\theta_{\text{ML}}||_2$$

[Hint: Consider using a proof by contradiction.]
Remark. For this reason, this form of regularization is sometimes also called **weight decay**, since it encourages the weights (meaning parameters) to take on generally smaller values.

Answer: Assume that

$$||\theta_{\text{MAP}}||_2 > ||\theta_{\text{ML}}||_2$$

Then, we have that

$$\begin{aligned} p(\theta_{\text{MAP}}) &= \frac{1}{(2\pi)^{\frac{n+1}{2}}|\tau^2 I|^{\frac{1}{2}}} e^{-\frac{1}{2\tau^2}(||\theta_{\text{MAP}}||_2)^2} \\ &< \frac{1}{(2\pi)^{\frac{n+1}{2}}|\tau^2 I|^{\frac{1}{2}}} e^{-\frac{1}{2\tau^2}(||\theta_{\text{ML}}||_2)^2} \\ &= p(\theta_{\text{ML}}) \end{aligned}$$

This yields

$$\begin{aligned} p(\theta_{\text{MAP}}) \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta_{\text{MAP}}) &< p(\theta_{\text{ML}}) \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta_{\text{MAP}}) \\ &\leq p(\theta_{\text{ML}}) \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta_{\text{ML}}) \end{aligned}$$

where the last inequality holds since θ_{ML} was chosen to maximize $\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta)$. However, this result gives us a contradiction, since θ_{MAP} was chosen to maximize $\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}, \theta)p(\theta)$

5. **[15 points] KL divergence and Maximum Likelihood**

   The Kullback-Leibler (KL) divergence between two discrete-valued distributions $P(X), Q(X)$ is defined as follows:[1]

$$KL(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

   For notational convenience, we assume $P(x) > 0, \forall x$. (Otherwise, one standard thing to do is to adopt the convention that "$0 \log 0 = 0$.") Sometimes, we also write the KL divergence as $KL(P||Q) = KL(P(X)||Q(X))$.

   The KL divergence is an assymmetric measure of the distance between 2 probability distributions. In this problem we will prove some basic properties of KL divergence, and work out a relationship between minimizing KL divergence and the maximum likelihood estimation that we're familiar with.

[1] If P and Q are densities for continuous-valued random variables, then the sum is replaced by an integral, and everything stated in this problem works fine as well. But for the sake of simplicity, in this problem we'll just work with this form of KL divergence for probability mass functions/discrete-valued distributions.

(a) Nonnegativity. Prove the following:

$$\forall P, Q \quad KL(P\|Q) \geq 0$$

and

$$KL(P\|Q) = 0 \quad \text{if and only if } P = Q.$$

[Hint: You may use the following result, called **Jensen's inequality**. If f is a convex function, and X is a random variable, then $E[f(X)] \geq f(E[X])$. Moreover, if f is strictly convex (f is convex if its Hessian satisfies $H \geq 0$; it is *strictly* convex if $H > 0$; for instance $f(x) = -\log x$ is strictly convex), then $E[f(X)] = f(E[X])$ implies that $X = E[X]$ with probability 1; i.e., X is actually a constant.]

Answer:

$$\begin{aligned}
-KL(P\|Q) &= -\sum_x P(x) \log \frac{P(x)}{Q(x)} && (1)\\
&= \sum_x P(x) \log \frac{Q(x)}{P(x)} && (2)\\
&\leq \log \sum_x P(x) \frac{Q(x)}{P(x)} && (3)\\
&= \log \sum_x Q(x) && (4)\\
&= \log 1 && (5)\\
&= 0 && (6)
\end{aligned}$$

Where all equalities follow from straight forward algebraic manipulation. The inequality follows from Jensen's inequality.

To show the second part of the claim, note that $\log t$ is a strictly concave function of t. Using the form of Jensen's inequality given in the lecture notes, we have equality if and only if $\frac{Q(x)}{P(x)} = E[\frac{Q(x)}{P(x)}]$ for all x. But since $E[\frac{Q(x)}{P(x)}] = \sum_x P(x)\frac{Q(x)}{P(x)} = \sum_x Q(x) = 1$, it follows that $P(x) = Q(x)$. Hence we have $KL(P\|Q) = 0$ if and only if $P(x) = Q(x)$ for all x.

(b) **Chain rule for KL divergence.** The KL divergence between 2 conditional distributions $P(X|Y), Q(X|Y)$ is defined as follows:

$$KL(P(X|Y)\|Q(X|Y)) = \sum_y P(y) \left(\sum_x P(x|y) \log \frac{P(x|y)}{Q(x|y)} \right)$$

This can be thought of as the expected KL divergence between the corresponding conditional distributions on x (that is, between $P(X|Y = y)$ and $Q(X|Y = y)$), where the expectation is taken over the random y.

Prove the following chain rule for KL divergence:

$$KL(P(X,Y)\|Q(X,Y)) = KL(P(X)\|Q(X)) + KL(P(Y|X)\|Q(Y|X)).$$

Answer:

$$
\begin{aligned}
KL(P(X,Y)\|Q(X,Y)) &= \sum_{x,y} P(x,y)\log\frac{P(x,y)}{Q(x,y)} && (7)\\
&= \sum_{x,y} P(x,y)\log\frac{P(x)P(y|x)}{Q(x)Q(y|x)} && (8)\\
&= \sum_{x,y} P(x,y)\log\frac{P(x)}{Q(x)} + P(x,y)\log\frac{P(y|x)}{Q(y|x)} && (9)\\
&= \sum_{x,y} P(x,y)\log\frac{P(x)}{Q(x)} + \sum_{x,y} P(x)P(y|x)\log\frac{P(y|x)}{Q(y|x)} && (10)\\
&= \sum_{x} P(x)\log\frac{P(x)}{Q(x)} + \sum_{x} P(x)\sum_{y} P(y|x)\log\frac{P(y|x)}{Q(y|x)} && (11)\\
&= KL(P(X)\|Q(X)) && (12)\\
&\quad + KL(P(Y|X)\|Q(Y|X)). && (13)
\end{aligned}
$$

Where we applied (in order): definition of KL, definition of conditional probability, log of product is sum of logs, splitting the summation, $\sum_y P(x,y) = P(x)$, definition of KL.

(c) **KL and maximum likelihood.**

Consider a density estimation problem, and suppose we are given a training set $\{x^{(i)}; i = 1,\ldots,m\}$. Let the empirical distribution be $\hat{P}(x) = \frac{1}{m}\sum_{i=1}^{m} 1\{x^{(i)} = x\}$. ($\hat{P}$ is just the uniform distribution over the training set; i.e., sampling from the empirical distribution is the same as picking a random example from the training set.)

Suppose we have some family of distributions P_θ parameterized by θ. (If you like, think of $P_\theta(x)$ as an alternative notation for $P(x;\theta)$.) Prove that finding the maximum likelihood estimate for the parameter θ is equivalent to finding P_θ with minimal KL divergence from $\hat{P}$. I.e. prove:

$$\arg\min_\theta KL(\hat{P}\|P_\theta) = \arg\max_\theta \sum_{i=1}^{m} \log P_\theta(x^{(i)})$$

Remark. Consider the relationship between parts (b-c) and multi-variate Bernoulli Naive Bayes parameter estimation. In the Naive Bayes model we assumed P_θ is of the following form: $P_\theta(x,y) = p(y)\prod_{i=1}^{n} p(x_i|y)$. By the chain rule for KL divergence, we therefore have:

$$KL(\hat{P}\|P_\theta) = KL(\hat{P}(y)\|p(y)) + \sum_{i=1}^{n} KL(\hat{P}(x_i|y)\|p(x_i|y)).$$

This shows that finding the maximum likelihood/minimum KL-divergence estimate of the parameters decomposes into $2n+1$ independent optimization problems: One for the class priors $p(y)$, and one for each of the conditional distributions $p(x_i|y)$ for each feature x_i given each of the two possible labels for y. Specifically, finding the maximum likelihood estimates for each of these problems individually results in also maximizing the likelihood of the joint distribution. (If you know what Bayesian networks are, a similar remark applies to parameter estimation for them.)

Answer:

$$\begin{aligned}
\arg\min_\theta KL(\hat{P}\|P_\theta) &= \arg\min_\theta \sum_x \hat{P}(x)\log\hat{P}(x) - \hat{P}(x)\log P_\theta(x) && (14)\\
&= \arg\min_\theta \sum_x -\hat{P}(x)\log P_\theta(x) && (15)\\
&= \arg\max_\theta \sum_x \hat{P}(x)\log P_\theta(x) && (16)\\
&= \arg\max_\theta \sum_x \frac{1}{m}\sum_{i=1}^m 1\{x^{(i)}=x\}\log P_\theta(x) && (17)\\
&= \arg\max_\theta \frac{1}{m}\sum_{i=1}^m\sum_x 1\{x^{(i)}=x\}\log P_\theta(x) && (18)\\
&= \arg\max_\theta \frac{1}{m}\sum_{i=1}^m \log P_\theta(x^{(i)}) && (19)\\
&= \arg\max_\theta \sum_{i=1}^m \log P_\theta(x^{(i)}) && (20)
\end{aligned}$$

where we used in order: definition of KL, leaving out terms independent of θ, flip sign and correspondingly flip min-max, definition of $\hat{P}$, switching order of summation, definition of the indicator and simplification.

6. **[20 points] K-means for compression**

   In this problem, we will apply the K-means algorithm to lossy image compression, by reducing the number of colors used in an image.

   The directory `/afs/ir.stanford.edu/class/cs229/ps/ps3/` contains a 512x512 image of a mandrill represented in 24-bit color. This means that, for each of the 262144 pixels in the image, there are three 8-bit numbers (each ranging from 0 to 255) that represent the red, green, and blue intensity values for that pixel. The straightforward representation of this image therefore takes about $262144 \times 3 = 786432$ bytes (a byte being 8 bits). To compress the image, we will use K-means to reduce the image to $k = 16$ colors. More specifically, each pixel in the image is considered a point in the three-dimensional (r, g, b)-space. To compress the image, we will cluster these points in color-space into 16 clusters, and replace each pixel with the closest cluster centroid.

   Follow the instructions below. Be warned that some of these operations can take a while (several minutes even on a fast computer)![2]

   (a) Copy mandrill-large.tiff from `/afs/ir.stanford.edu/class/cs229/ps/ps3` on the leland system. Start up MATLAB, and type `A = double(imread('mandrill-large.tiff'));` to read in the image. Now, `A` is a "three dimensional matrix," and `A(:,:,1)`, `A(:,:,2)` and `A(:,:,3)` are 512x512 arrays that respectively contain the red, green, and blue values for each pixel. Enter `imshow(uint8(round(A)));` to display the image.

   (b) Since the large image has 262144 pixels and would take a while to cluster, we will instead run vector quantization on a smaller image. Repeat (a) with `mandrill-small.tiff`.

[2] In order to use the `imread` and `imshow` commands in octave, you have to install the Image package from octave-forge. This package and installation instructions are available at: `http://octave.sourceforge.net`

Treating each pixel's (r, g, b) values as an element of $\mathbb{R}^3$, run K-means[3] with 16 clusters on the pixel data from this smaller image, iterating (preferably) to convergence, but in no case for less than 30 iterations. For initialization, set each cluster centroid to the (r, g, b)-values of a randomly chosen pixel in the image.

(c) Take the matrix `A` from `mandrill-large.tiff`, and replace each pixel's (r, g, b) values with the value of the closest cluster centroid. Display the new image, and compare it visually to the original image. Hand in all your code and a printout of your compressed image (printing on a black-and-white printer is fine).

(d) If we represent the image with these reduced (16) colors, by (approximately) what factor have we compressed the image?

Answer: Figure **??** shows the original image of the mandrill. Figure **??** shows the image compressed into 16 colors using K-means run to convergence, and shows the 16 colors used in the compressed image. (These solutions are given in a color PostScript file. To see the colors without a color printer, view them with a program that can display color PostScript, such as ghostview.) The original image used 24 bits per pixel. To represent one of 16 colors requires $log_2 16 = 4$ bits per pixel. We have therefore achieved a compression factor of about $24/4 = 6$ of the image. MATLAB code for this problem is given below.

```
A = double(imread('mandrill-small.tiff'));
imshow(uint8(round(A)));

% K-means initialization
k = 16;
initmu = zeros(k,3);
for l=1:k,
    i = random('unid', size(A, 1), 1, 1);
    j = random('unid', size(A, 2), 1, 1);
    initmu(l,:) = double(permute(A(i,j,:), [3 2 1])');
end;

% Run K-means
mu = initmu;
for iter = 1:200, % usually converges long before 200 iterations
    newmu = zeros(k,3);
    nassign = zeros(k,1);
    for i=1:size(A,1),
        for j=1:size(A,2),
            dist = zeros(k,1);
            for l=1:k,
                d = mu(l,:)'-permute(A(i,j,:), [3 2 1]);
                dist(l) = d'*d;
            end;
            [value, assignment] = min(dist);
            nassign(assignment) = nassign(assignment) + 1;
            newmu(assignment,:) = newmu(assignment,:) + ...
                permute(A(i,j,:), [3 2 1])';
```

[3]Please implement K-means yourself, rather than using built-in functions from, e.g., MATLAB or octave.

```
        end; end;
    for l=1:k,
        if (nassign(l) > 0)
            newmu(l,:) = newmu(l,:) / nassign(l);
        end;
    end;
    mu = newmu;
end;

% Assign new colors to large image
bigimage = double(imread('mandrill-large.tiff'));
imshow(uint8(round(bigimage)));
qimage = bigimage;
for i=1:size(bigimage,1), for j=1:size(bigimage,2),
        dist = zeros(k,1);
        for l=1:k,
            d = mu(l,:)'-permute(bigimage(i,j,:), [3 2 1]);
            dist(l) = d'*d;
        end;
        [value, assignment] = min(dist);
        qimage(i,j,:) = ipermute(mu(assignment,:), [3 2 1]);
    end; end;
imshow(uint8(round(qimage)));
```

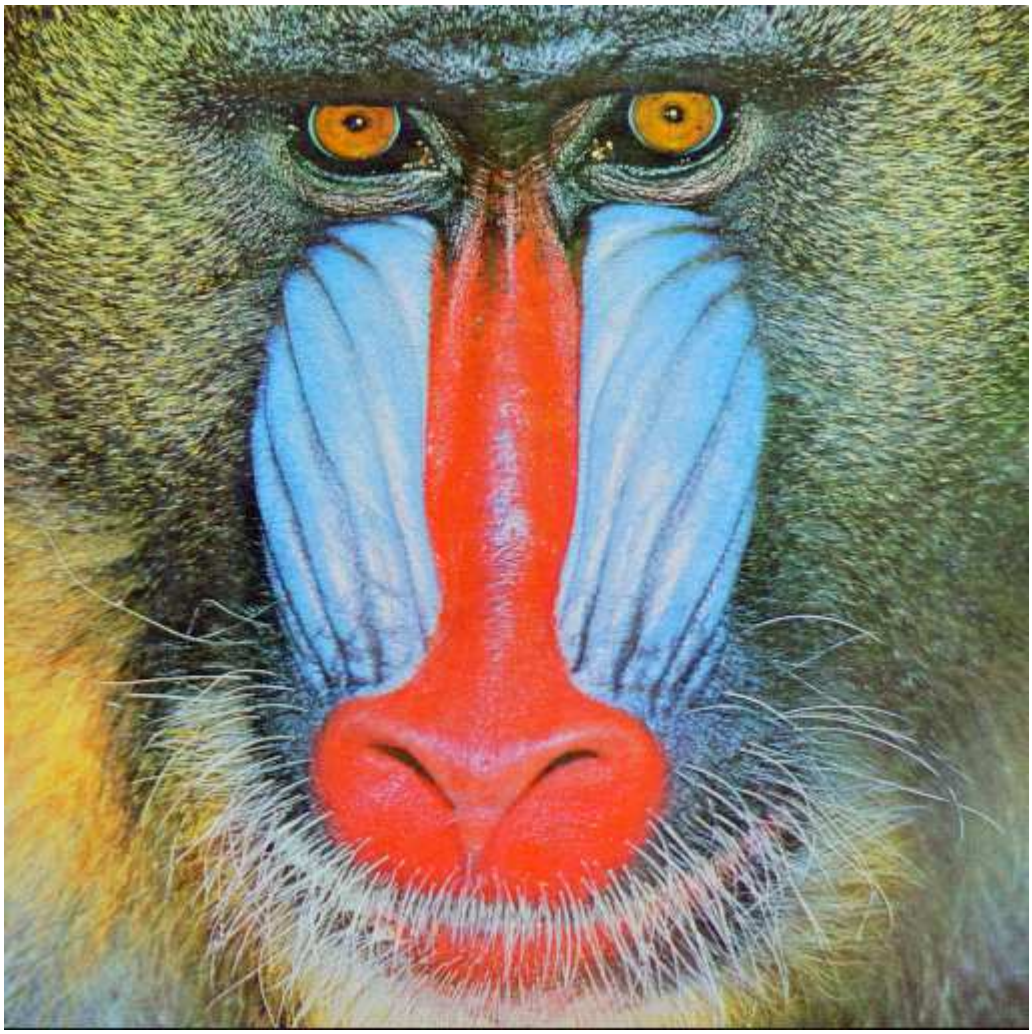

Figure 1: The original image of the mandrill.

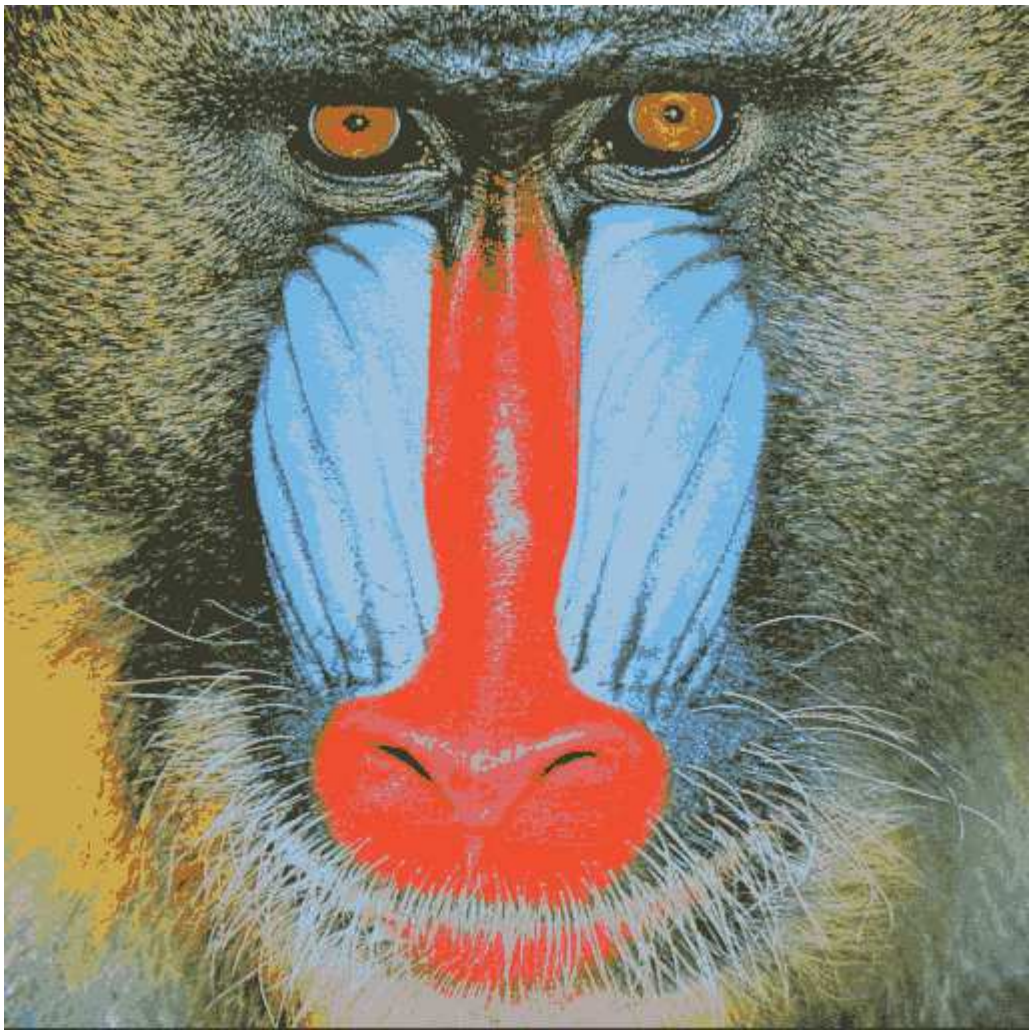

Figure 2: The compressed image of the mandrill.