CPSC 540 Assignment 6 (due November 12)

Fenchel Dual, Ensemble Methods, Hidden Variables

Please put your name and student number on the assignment, staple the assignment together.

1 Fenchel Duality

Recall that the Fenchel dual for the primal problem

$$P(x) = f(Ax) + g(x),$$

is the dual problem

$$D(y) = -f^*(-y) - g^*(A^T y),$$

or if we re-parameterize in terms of $-y$:

$$D(y) = -f^*(y) - g^*(-A^T y), \tag{1}$$

1.1 Deriving Dual Problems

Convex conjugates are discussed in Section 3.3 of Boyd and Vandenberghe (`http://stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`). Read this, then derive the Fenchel dual for the following problems:

1. $P(x) = \frac{1}{2}\|Ax - b\|^2 + \frac{\lambda}{2}\|x\|^2$ (dual ridge regression)
2. $P(x) = \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1$ (quadratic w/ scaled norm-ball constraints)
3. $P(x) = \sum_{i=1}^N \log(1 + \exp(-b_i x^T a_i)) + \frac{\lambda}{2}\|x\|^2$ (regularized maximum entropy)

Hint: The structure of the primal and dual problem make this question much easier than taking a generic Lagrangian dual. A generic strategy to compute Fenchel duals is as follows:

- Determine A, f, and g to put the problem into the primal format.
- Determine the form $f^*(y)$ and $g^*(y)$ (note that A here is not relevant).
- Evaluate f^* at $-y$ and g^* at $A^T y$ to get the final result.

As an example, we'll do this step by step for the first one, $P(x) = \frac{1}{2}\|Ax - b\|^2 + \frac{\lambda}{2}\|x\|^2$:

- In this case, we get this simplest primal by just using the design matrix as A. With this choice, the primal can be written as
$$P(x) = f(Ax) + g(x),$$
where $f(z) = \frac{1}{2}\|z - b\|^2$ and $g(x) = \frac{\lambda}{2}g(x)$.

- We need to evaluate the two conjugate functions,

$$f^*(y) = \sup_z\{y^T z - \frac{1}{2}\|z-b\|^2\}, \quad g^*(y) = \sup_x\{y^T x - \frac{\lambda}{2}\|x\|^2\}.$$

  In both cases, we are maximizing a smooth concave function and we can compute the optimal value by taking the gradient and setting it to zero. For f^*, this gives us

$$0 = y - (z-b),$$

  or

$$z = y + b.$$

  To get the final form of $f^*(y)$, we plug in this value of z into the sup,

$$f^*(y) = y^T(y+b) - \frac{1}{2}\|(y+b)-b\|^2 = \|y\|^2 + y^T b - \frac{1}{2}\|y\|^2 = \frac{1}{2}\|y\|^2 + y^T b.$$

  You can get the form of $g^*(y)$ using the same method, or you could use that if $g(x) = ah(x)$ then $g^*(y) = ah^*(\frac{1}{a}y)$ (see Section 3.3.2) to give

$$g^*(y) = \frac{\lambda}{2}\|\frac{1}{\lambda}y\|^2 = \frac{1}{2\lambda}\|y\|^2.$$

- Now evaluate f^* at $-y$ and g^* and $A^T y$ to get the final result,

$$D(y) = -f^*(-y) - g^*(A^T y) = -(\frac{1}{2}\|(-y)\|^2 + (-y)^T b) - \frac{1}{2\lambda}\|A^T y\|^2 = -\frac{1}{2}\|y\|^2 + y^T b - \frac{1}{2\lambda}y^T A A^T y.$$

For part 2, you can use that if $f(x) = \|x\|_p$, then the convex conjugate is given by (see Example 3.26)

$$f^*(y) = \begin{cases} 0 & \|y\|_q \leq 1 \\ \infty & \text{otherwise} \end{cases},$$

where $\|\cdot\|_q$ is the dual norm of $\|\cdot\|_p$. For this problem, $p = 1$ so its dual norm uses $q = \infty$. Also, the dual-norm of $\lambda\|\cdot\|_p$ is $\frac{1}{\lambda}\|\cdot\|_q$.

For part 3, life gets easier if you define $\tilde{A}$ as a matrix where each row is given by $(-b_i a_i)$. You can then write the problem as

$$P(x) = f(\tilde{A}x) + g(x),$$

where $f(z) = \sum_{i=1}^N \log(1+\exp(z_i))$. Here z is a vector and note that f is a separable function. Because it is separable, the conjugate simplifies:

$$\begin{aligned} f^*(y) &= \sup_z\{y^T z - \sum_{i=1}^N \log(1+\exp(z_i))\} \\ &= \sup_z\{\sum_{i=1}^N y_i z_i - \sum_{i=1}^N \log(1+\exp(z_i))\} \\ &= \sum_{i=1}^N \sup_{z_i}\{y_i z_i - \sum_{i=1}^N \log(1+\exp(z_i)\} \quad \text{(we can optimize the } z_i \text{ independently)} \\ &= \sum_{i=1}^N h^*(y_i), \end{aligned}$$

where $h(z_i) = \log(1+\exp(z_i))$. (In fact, the conjugate will always simplify into a sum of conjugates over all training examples and the ℓ_2-regularization guarantees the dual will be smooth, and this is what has made dual coordinate ascent a popular solver.) For part 3, it will be a bit nicer if you use (1) as the final form of the dual.

1.2 Dual SVM Solver

The dual of the SVM problem,

$$P(x) = \sum_{i=1}^{N} \max\{0, 1 - b_i x^T a_i\} + \frac{\lambda}{2}\|x\|^2,$$

is

$$D(y) = e^T y - \frac{1}{2\lambda} y^T \tilde{A}\tilde{A}^T y, \quad \text{s.t. } 0 \leq y_i \leq 1, \forall_i.$$

where e is a vector of ones and row i of $\tilde{A}$ is $b_i a_i$ and we have $x^* = \frac{1}{\lambda}\tilde{A}^T y^*$.

Starting from *dualSVMDemo.m*, implement a dual coordinate ascent strategy to optimize the SVM objective. Hand in your code and report the optimal value of the dual objective with $\lambda = 1$.

Hint: the objective function is quadratic, so you can derive a closed-form solution for the optimal update of each variable. However, this solution may not satisfy the constraints so you will need to project that solution onto the constraint set. To help debugging, you can explicitly compute the dual objective after each update (it should never go down).

Note that in the dual problem, the 'support vectors' correspond to the non-zero entries of y.

Hint: As in the code, let's use the notation $G = \tilde{A}\tilde{A}^T$. With this notation the dual objective is

$$D(y) = e^T y - \frac{1}{2\lambda} y^T G y,$$

and its gradient is

$$\nabla D(y) = e - \frac{1}{\lambda} G y.$$

Normally we could just solve for y but this doesn't work because of the constraints. But each constraint only affects one variable so we can optimize one variable at a time and modify the update to maintain the constraints. The gradient with respect to one variable i is

$$\nabla_i D(y) = 1 - \frac{1}{\lambda} y^T g_i$$

where g_i is row i of G (which we've written as a column-vector). If we use the definition of the inner product we get that the update to y_i needs to solve

$$0 = 1 - \frac{1}{\lambda} \sum_{j=1}^{N} y_j g_{ij}.$$

To get the update for this coordinate, you solve this for y_i, and then project this value onto the constraint set $0 \leq y_i \leq 1$.

2 Ensemble Methods

In a previous assignment, we can the function *findMin* an anonymous function that computes the objective function and gradient for a given parameter vector. We can use the idea of anonymous functions to implement generic ensemble methods, where the anonymous function trains a model given a (possibly-weighted) dataset. In this question you will modify a generic bagging and a generic boosting code.

2.1 Bagging

If you run the script *baggingDemo.m*, it generates m copies of a dataset and fits a polynomial-basis linear regression model to each dataset, then uses the average of these models as the final result. This script then shows a plot of the individual models and their average (which are all identical). Modify the demo so that instead of generating m identical copies of the original data, it uses m boostrap samples. Hand in the updated plot after making this modification.

2.2 Boosting

If you run the script *boostingDemo.m* (after adding the 'minFunc' directory to the path using *addpath minFunc* in Matlab), it runs a logistic regression model on a dataset with two features and plots the results, and then it applies a boosted version of logistic regression (which involves fitting weighted logistic regression models). As you can see, boosting doesn't seem to help much. Write a model function that implements the decision stump that minimizes the (weighted) classification error (and be sure to modify the *predict* function to classify points based on the threshold). Note that infogain will not work with boosting since it doesn't guarantee that the weighted accuracy is above 50%. Hand in the new model function, the plot when using the decision stump directly (with equal weights to all points), and the the plot with boosted decision stumps.

Be sure to check the threshold in both the '>' and '<' cases!

3 Expectation Maximizaiton

This question explores two of the most common applications of the EM algorithm: fitting mixture models and semi-supervised learning.

3.1 Gaussian Mixture Models

The script *mixtureDemo.m* fits a Gaussian distribution to a dataset that is not uni-modal. It thus gives a very bad fit. Use the EM algorithm to fit mixture of 3 Gaussians to this dataset (see Section 11.4.2 of the textbook). Hand in your code and the updated plot.

Hint: The Gaussian mixture model uses a PDF of the form

$$p(x_i) = \sum_{k=1}^{K} p(z_i = k|\pi)p(x_i|\mu_k, \Sigma_k),$$

where each mixture component is a multivariate Gaussian,

$$p(x_i|\mu, \Sigma_k) = (2\pi)^{-\frac{d}{2}}|\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x_i - \mu)^T\Sigma^{-1}(x_i - \mu)\right).$$

and z_i follows a multinoulli,

$$p(z_i = k) = \frac{\pi_k}{\sum_j \pi_j}.$$

One way to interpret this is, to generate each x_i, we first generate a discrete z_i and then given this z_i we generate x_i from the corresponding Gaussian. A standard way to train this model that leads to simple/intuitive closed-form updates is by treating the z_i as hidden variables and applying expectation maximization.

Given the full set of parameters $\{\pi^t, \mu^t, \Sigma^t\}$ at iteration t, the E-step computes the weights (called 'responsibilities' in this context)

$$r_{ik} = p(z_i = k|x_i, \pi^t, \mu^t, \Sigma^t) = \frac{p(z_i = k|\pi_k)p(x_i|\mu_k, \Sigma_k)}{\sum_{k'=1}^{K} p(z_i = k'|\pi_k)p(x_i|\mu_{k'}, \Sigma_{k'})}.$$

Using this in the EM machinery leads to the following updates, which are weighted MLE estimates:

$$\pi_k^{t+1} = \frac{1}{N}\sum_{i=1}^{N} r_{ik},$$

$$\mu_k^{t+1} = \frac{\sum_{i=1}^{N} r_{ik}x_i}{\sum_{i=1}^{N} r_{ik}},$$

$$\Sigma_k^{t+1} = \frac{\sum_{i=1}^{N} r_{ik}(x_i - \mu_k^t)(x_i - \mu_k^t)^T}{\sum_{i=1}^{N} r_{ik}}.$$

Notice that if we just have one cluster, $r_{ik} = 1$ so μ_1 and Σ_1 would be the standard MLE estimates. It is possible that Σ_k^{t+1} may not be positive-definite, in such cases a MAP estimate for Σ would add small multiple of the identity matrix to the above estimate.

3.2 Semi-Supervised Naive Bayes

Consider a scenario where we have a small labeled data set $\{X_L, y_L\}$ and a large unlabelled data set X_U. Assume that all variables are binary, and we are use a naive Bayes classifier. Derive the EM algorithm that results from training on the full data set combining $\{X_L, y_L\}$ and X_U and treating the unknown lables y_U as hidden variables.

Hint:

The naive Bayes model assumes $p(y_i, x_i|\theta) = p(y_i|\theta)\prod_{j=1}^{d} p((x_i)_j|y_i, \theta)$, and we can use the notation

$$y_i|\theta \sim \text{Ber}(\theta_1), \quad (x_i)_j|y_i, \theta \sim \text{Ber}(\theta_{y_i j}),$$

with $\theta = \{\theta_1, \theta_{01}, \theta_{11}, \theta_{02}, \theta_{12}, \ldots, \theta_{0d}, \theta_{1d}\}$.

Let's use N as the number of labeled examples and T as the number of unlabeled examples. If we were given $\{X_L, y_L, X_U, y_U\}$ the likelihood would be given by

$$p(y_L, X_L, X_U, y_U|\theta) = \prod_{i=1}^{N}[p(y_i, x_i|\theta)]\prod_{i=1}^{T}[p(y_i, x_i|\theta)],$$

the first product is over the labeled examples and the second product is over the unlabeled examples, and if we had these labels we could fit this using the closed-form solution as in Assignment 1 (Question 6). Since we do not actually have the labels of the unlabelled examples, we marginalize over all of the possible values of the hidden labels. This gives a likelihood for $\{X_L, y_L, X_U, y_U\}$ of

$$\begin{aligned} p(y_L, X_L, X_U|\theta) &= \sum_{y_1\in\{0,1\}}\sum_{y_2\in\{0,1\}}\cdots\sum_{y_T\in\{0,1\}}\left[\prod_{i=1}^{N}[p(y_i, x_i|\theta)]\prod_{i=1}^{T}[p(y_i, x_i|\theta)]\right] \\ &= \prod_{i=1}^{N}[p(y_i, x_i|\theta)]\sum_{y_1\in\{0,1\}}\sum_{y_2\in\{0,1\}}\cdots\sum_{y_T\in\{0,1\}}\left[\prod_{i=1}^{T}[p(y_i, x_i|\theta)]\right] \\ &= \prod_{i=1}^{N}[p(y_i, x_i|\theta)]\prod_{i=1}^{T}[\sum_{y_i\in\{0,1\}} p(y_i, x_i|\theta)]. \end{aligned}$$

The first line above sums over all 2^T possible values of the T hidden labels y_U. The second line takes the likelihood for the labeled examples outside the sum (the labels of the unlabeled examples do not occur in these factors). The third line uses the distributive law to push the individual sums inside the corresponding products (e.g., $\sum_i \prod_j a_{ij} = \prod_j \sum_i a_{ij}$).

This gives a log-likelihood of

$$\log p(y_L, X_L, X_U) = \sum_{i=1}^{N} \log p(y_i, x_i|\theta) + \sum_{i=1}^{T} \log \left(\sum_{y_i \in \{0,1\}} p(y_i, x_i|\theta) \right).$$

This is no longer nice/convex because of the second term, but the EM iterations will be able to take advantage of the fact that if we knew y_U the problem would be easy.

E-step: The E-step involes computing $p(y_i|x_i, \theta^t)$ for the unlabeled examples. You will have to use the definitions of the naive Bayes model and the Bernoulli distribution to derive an expression for these values.

M-step: Let's use the notation $r_{i1}^t = p(y_i = 1|x_i, \theta^t)$ and $r_{i0}^t = p(y_i = 0|x_i, \theta^t)$ to denote the weights that we get from the E-step. The objective in the M-step is:

$$\begin{aligned}
&Q(\theta|\theta^t) = \mathbb{E}_{y_U|X_L,y_L,X_U}[\log p(y_L, X_L, X_U, y_U)] \\
&= \sum_{y_1 \in \{0,1\}} \sum_{y_2 \in \{0,1\}} \cdots \sum_{y_T \in \{0,1\}} p(y_U|X_L, y_L, X_U, \theta^t)[\log p(y_L, X_L, X_U, y_U)] \\
&= \sum_{y_U} p(y_U|X_L, y_L, X_U, \theta^t)[\log p(y_L, X_L, X_U, y_U)] && \text{(notation for the summations)} \\
&= \sum_{y_U} p(y_U|X_L, y_L, X_U, \theta^t)[\sum_{i=1}^{N} \log p(y_i, x_i|\theta) + \sum_{i=1}^{T} \log p(y_i, x_i|\theta)] && \text{(log-likelihood definition)} \\
&= \sum_{i=1}^{N} \log p(y_i, x_i|\theta) \sum_{y_U} p(y_U|X_L, y_L, X_U, \theta^t) + \sum_{i=1}^{T} \sum_{y_U} p(y_U|X_L, y_L, X_U, \theta^t) \log p(y_i, x_i|\theta) && \text{(switch order of sums)} \\
&= \sum_{i=1}^{N} \log p(y_i, x_i|\theta) + \sum_{i=1}^{T} \sum_{y_i \in \{0,1\}} p(y_i|x_i, \theta^t) \log p(y_i, x_i|\theta) && (*) \\
&= \sum_{i=1}^{N} \log p(y_i, x_i|\theta) + \sum_{i=1}^{T} \sum_{y_i \in \{0,1\}} r_{iy_i}^t \log p(y_i, x_i|\theta).
\end{aligned}$$

This is the log-likelihood for a weighted naive Bayes model. You will have to use the definitions of the naive Bayes and the Bernoulli distribution to derive the elements of θ that maximizes $Q(\theta|\theta^t)$, you may want to review the notes on naive Bayes on the course webpage for hints on how the MLE in naive Bayes is derived.

In the line marked $(*)$, for the first term we use that $\sum_{y_U} p(y_U|X_L, y_L, X_U, \theta^t) = 1$ because it is a probability.

For the second line we use that

$$\sum_{i=1}^{T}\sum_{y_U} p(y_U|X_L, y_L, X_U, \theta^t)\log p(y_i, x_i|\theta)$$
$$= \sum_{i=1}^{T}\sum_{y_U}\prod_{j=1}^{T}[p(y_j|x_j, \theta^t)]\log p(y_i, x_i|\theta)$$
$$= \sum_{i=1}^{T}\sum_{y_1\in\{0,1\}}\sum_{y_2\in\{0,1\}}\cdots\sum_{y_T\in\{0,1\}}\prod_{j=1}^{T}[p(y_j|x_j, \theta^t)]\log p(y_i, x_i|\theta)$$
$$= \sum_{i=1}^{T}\sum_{y_i\in\{0,1\}}\sum_{y_{k_1}\in\{0,1\}}\sum_{y_{k_2}\in\{0,1\}}\cdots\sum_{y_{k_{T-1}}\in\{0,1\}}\prod_{j=1}^{T}[p(y_j|x_j, \theta^t)]\log p(y_i, x_i|\theta)$$

(the k_j range over 1 to T but exclude i)

$$= \sum_{i=1}^{T}\sum_{y_i\in\{0,1\}} p(y_i|x_i, \theta^t)\log p(y_i, x_i|\theta)\sum_{y_{k_1}\in\{0,1\}} p(y_{k_1}|x_{k_1}, \theta^t)\sum_{y_{k_2}\in\{0,1\}} p(y_{k_2}|x_{k_2}, \theta^t)\cdots\sum_{y_{k_{T-1}}\in\{0,1\}} p(y_{k_{T-1}}|x_{k_{T-1}}, \theta^t)$$
$$= \sum_{i=1}^{T}\sum_{y_i\in\{0,1\}} p(y_i|x_i, \theta^t)\log p(y_i, x_i|\theta),$$

where in the last line we observe that all the inner sums are equal to one because they are probabilities.