CSCI 5254 Homework 6

Tuguluke Abulitibu

November 17, 2020

Chapter 7, Estimation

7.3

Since v is a zero mean unit variance Gaussian variable, we have the CDF (from sum to integral, since integral is convex):

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt$$

in our case:

$$\begin{cases} \mathbf{prob}(x|y=1) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-z^2}{2}} dz \\ \mathbf{prob}(x|y=0) = 1 - \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-z^2}{2}} dz \end{cases}$$

hence the likely function[1] is

$$\prod_{i=1}^{q} P_i(a^T u_i + b|y=1) \prod_{i=q+1}^{m} (1 - P_i(a^T u_i + b|y=0))$$

taking the log:

$$l(a,b) = \sum_{i=1}^{q} log(P_i(a^T u_i + b|y=1)) + \sum_{i=q+1}^{m} \log(1 - P_i(a^T u_i + b|y=0))$$

$$\boxed{l(a,b) = \sum_{y_i=1} \log P_i(a^T u_i + b) + \sum_{y_i=1} log(1 - P_i(a^T u_i + b))}$$

objective is concave, hence the problem is convex.

7.4 (a)

$$-\frac{N}{2} n \log(2\pi) - \frac{N}{2} \log(\det R) - \frac{1}{2} R^{-1} \sum_{k=1}^{N} (y_k - a)(y_k - a)^T$$

$$= -\frac{N}{2} n \log(2\pi) - \frac{N}{2} \log(\det R) - \frac{1}{2} R^{-1} (\sum_{k=1}^{N} y_k y_k^T - \sum_{k=1}^{N} a y_k^T - \sum_{k=1}^{N} y_k a^T + N a a^T)$$

plug in sample mean $\mu = \frac{1}{N} \sum_{k=1}^{n} y_k$, we have

$$= -\frac{N}{2} n \log(2\pi) - \frac{N}{2} \log(\det R) - \frac{1}{2} R^{-1} (\sum_{k=1}^{N} y_k y_k^T - N a \mu^T - N \mu a^T + N a a^T)$$

[1] $\prod_{i=1}^{q} p_i \prod_{i=q+1}^{m} (1 - p_i)$

$$= -\frac{N}{2}n\log(2\pi) - \frac{N}{2}\log(\det R) - R^{-1}\sum_{k=1}^{N}(y_k-\mu)(y_k-\mu)^T - R^{-1}N(a-\mu)(a-\mu)^T$$

plug in covariance $Y = \frac{1}{N}\sum_{k=1}^{N}(y_k-\mu)(y_k-\mu)^T$, we have

$$-\frac{N}{2}n\log(2\pi) - \frac{N}{2}\log(\det R) - \frac{1}{2}(NR^{-1}Y + R^{-1}N(a-\mu)(a-\mu)^T)$$

that is the equivalent

$$= \frac{N}{2}(-n\log(2\pi) - \frac{N}{2}\log(\det R) - \mathbf{tr}(R^{-1}Y) - (a-\mu)^T R^{-1}(a-\mu))$$

Next, by letting

$$\begin{cases} \frac{\partial}{\partial a}l(R,a) = -2R^{-1}(a-\mu) = 0 \\ \frac{\partial}{\partial R}l(R,a) = -R^{-1} + R^{-1}(Y-(a-\mu)(a-\mu)^T)R^{-1} = 0 \end{cases} \Rightarrow \begin{cases} a_{ml} = \mu \\ R_{ml} = Y \end{cases}$$

7.8

We order values with $y > 1$ followed by $y < 0$, hence

$$\prod_{i=1}^{k}\mathbf{prob}(a_i^T x + b_i + v_i > 0)\prod_{i=k+1}^{m}\mathbf{prob}(a_i^T x + b_i + v_i < 0)$$

Since the relative variance is the noise term v_i[2], we introduce CDF F for **prob**:

$$\prod_{i=1}^{k}F(-a_i^T x - b_i)\prod_{i=k+1}^{m}1 - F(-a_i^T x - b_i)$$

from 7.3, we know that the log-likeihood function is concave

$$l(x) = \sum_{i=1}^{k}\log(F(-a_i^T x - b_i)) + \sum_{i=k+1}^{m}\log(1 - F(-a_i^T x - b_i))$$

therefore maximize problem will be concex[3].

7.9

$f'(t) > 0$, then it is a monotone function, hence, f is invertible, then from

$$y_i = f(a_i^T x + b_i + v_i), i = 1, \ldots, m$$

we can get

$$v_i = f^{-1}(y_i) - a_i^T x - b_i$$

hence the probability of y_i is

$$\prod_{i=1}^{m}\mathbf{prob}(f^{-1}(y_i) - a_i^T x - b_i)$$

[2] a_i, b_i are known

[3] Textbook page 358:Therefore, for any maximum likelihood estimation problem with concave log likelihood function, we can add a prior density for x that is log-concave, and the esulting MAP estimation problem will be convex.

the log-likelihood function will be

$$l(x,f) = \sum_{i=1}^{m} \log(\mathbf{prob}(f^{-1}(y_i) - a_i^T x - b_i))$$

since $f' \in [l, u]$, we know that $f^{-1} \in [1/u, 1/l]$, we get to the convex optimization of ml:

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{m} \log(\mathbf{prob}(f^{-1}(y_i) - a_i^T x - b_i)) \\ \text{subject to} & \dfrac{\|y_i - y_j\|}{u} \leq \|f_i^{-1} - f_h^{-1}\| \leq \dfrac{\|y_i - y_j\|}{l} \end{array}$$

introducing $z = f^{-1}$

$$\boxed{\begin{array}{ll} \text{maximize} & \sum_{i=1}^{m} \log(\mathbf{prob}(z_i - a_i^T x - b_i)) \\ \text{subject to} & \dfrac{\|y_i - y_j\|}{u} \leq \|z_i - z_j\| \leq \dfrac{\|y_i - y_j\|}{l} \end{array}}$$

Chapter 8, Extremal volume ellipsoids

8.16

first of all, we know

$$v = \prod_{i=1}^{n}(u_i - l_i)$$

then maximizing the volume can be maximizing

$$\prod_{i=1}^{n}(u_i - l_i)^{\frac{1}{n}}$$

this is geometric means, hence concave.
Now we can express x_i as u_i, l_i, hence

$$\sum_{i=1}^{n} a_{ij}(u_j - l_j) \leq b_i$$

by introducing

$$\begin{cases} a_{ij}^{+} = \max\{a_{ij}, 0\} \\ a_{ij}^{-} = \max\{-a_{ij}, 0\} \end{cases}$$

we can rewrite the system as

$$\sum_{i=1}^{n}(a_{ij}^{+} u_j - a_{ij}^{-} l_j) \leq b_i$$

the max volume problem then become

$$\begin{array}{ll} \text{maximize} & (\prod_{i=1}^{n}(u_i - l_i))^{1/n} \\ \text{subject to} & \sum_{i=1}^{n}(a_{ij}^{+} u_j - a_{ij}^{-} l_j) \leq b_i, \ \forall i \end{array}$$

taking the log, we have

$$\boxed{\begin{array}{ll} \text{maximize} & \log(u_i - l_i))^{1/n} \\ \text{subject to} & \sum_{i=1}^{n}(a_{ij}^{+} u_j - a_{ij}^{-} l_j) \leq b_i, \ \forall i \\ & u_i \succeq l_i \ \forall i \end{array}}$$

Chapter 8, Classification

8.24

Since

$$(a+u)^T x_i \geq b \Leftrightarrow a^T x_i + ||u||_2 |x_i||_2 \geq b$$

and $\|u\|_2 \leq \rho$, we have

$$a^T x_i + \rho \|x_i\|_2 \geq b \Leftrightarrow \rho \leq \frac{a^T x_i - b}{\|x_i\|_2}$$

same for

$$(a+u)^T y_j \leq b \Leftrightarrow a^T y_j - b \leq -\rho \|y_j\|_2 \Leftrightarrow \rho \leq \frac{b - a^T y_i}{\|y_i\|_2}$$

this is to say that weight error can be

$$\min\{\frac{a^T x_i - b}{\|x_i\|_2}, \frac{b - a^T y_i}{\|y_i\|_2}\}$$

now, introducing auxiliary variable t, the weight error margin problem(maximizing the margin) can be written as

$$\boxed{\begin{array}{ll} \text{maximize} & t \\ \text{subject to} & a^T x_i - b \geq t\|x_i\|_2 \; i = i, \cdots, N \\ & b - a^T y_i \geq t\|y_i\|_2 \; j = i, \cdots, M \\ & \|a\|_2 \leq 1 \end{array}}$$

Additional Exercises

5.12 Least-squares with some permuted measurements

Estimate an initial $\hat{x}$ using the huber penalty function. We then use that $\hat{x}$ to calculate a $\hat{P}$ by aligning the indices of Ax and y to find a permutation matrix. Repeat until the euclidean norm of the distance between the $\hat{x}_\tau$ and $\hat{x}_{\tau-1}$ is below tolerance, then stop.

Algorithm 1: Least-squares with some permuted measurements

initialization
$x(0) \leftarrow \arg_x \|Ax - y\|$;
repeat
  $P(t) \leftarrow \arg max \|Ax(t) - P^T y\|_2$
  $x(t+1) \leftarrow \arg max \|Ax - P(t)^T y\|_2$
  Stop if $\|x(t-1) - x(t)\|_2 \leq \epsilon$;
until $P(t) = P(t-1)$;

$$\begin{cases} \|x_{true} - x_{naive}\| = 2.2683660401079058 \\ \|x_{true} - x_{final}\| = 0.08421494480703208 \end{cases}$$

```python
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)
m=100
k=40 # max # permuted measurements
n=20
A=10*np.random.randn(m,n)
x_true=np.random.randn(n) # true x value
y_true = A.dot(x_true) + np.random.randn(m)
# build permuted indices
perm_idxs=np.random.permutation(m)
perm_idxs=np.sort(perm_idxs[:k])
```

```python
temp_perm=np.random.permutation(k)
new_pos=np.zeros(k)
for i in range(k):
    new_pos[i] = perm_idxs[temp_perm[i]]
new_pos = new_pos.astype(int)
# true permutation matrix
P=np.identity(m)

P[perm_idxs]=P[new_pos,:]
true_perm=[]
for i in range(k):
    if perm_idxs[i] != new_pos[i]:
        true_perm = np.append(true_perm, perm_idxs[i])
y=P.dot(y_true)
new_pos = None
# naive estimator (P=I)
x_naive = np.linalg.lstsq(A,y)[0]
# robust estimator
x_hub = cp.Variable(n)
obj = cp.sum(cp.huber(A@x_hub-y))  #TODO dimention bugs
cp.Problem(cp.Minimize(obj)).solve()
plt.figure(1)
plt.plot(np.arange(m), np.abs(A.dot(x_hub.value)-y), '.')
plt.ylabel('residual')
plt.xlabel('idx')
plt.savefig('prob_152.png')

# remove k largest residuals
cand_idxs = np.zeros(m)
cand_idxs[:] = np.flip(np.argsort(np.abs(A.dot(x_hub.value)-y)))
cand_idxs = np.sort(cand_idxs[:k])
cand_idxs = cand_idxs.astype(int)
keep_idxs = np.zeros(m)
keep_idxs[:] = np.argsort(np.abs(A.dot(x_hub.value)-y).T)
keep_idxs = np.sort(keep_idxs[:(m-k)])
keep_idxs = keep_idxs.astype(int)
# print(np.shape(A))
# print(np.shape(y))

A_hat = A[keep_idxs,:]
# print(np.shape(A_hat))
y_hat = y[keep_idxs]
# print(np.shape(y_hat))

# ls estimate with candidate idxs removed
x_ls = np.linalg.lstsq(A_hat,y_hat)[0]
# match predicted outputs with measurements
b = np.zeros(k)
c = np.zeros(k)
b[:] = np.argsort(A[cand_idxs,:].dot(x_ls).T)
b = b.astype(int)
c[:] = np.argsort(y[cand_idxs].T)
c = c.astype(int)
# reorder A matrix
cand_perms = np.zeros(len(cand_idxs))
cand_perms[:]=cand_idxs[:]
cand_perms[b]=cand_perms[c]
cand_perms = cand_perms.astype(int)
A[cand_perms,:]=A[cand_idxs,:]
x_final = np.linalg.lstsq(A,y)[0]
# final estimate of permuted indices
perm_estimate = []
for i in range(k):
    if cand_perms[i] != cand_idxs[i]:
        perm_estimate = np.append(perm_estimate, cand_idxs[i])
naive_error = np.linalg.norm(x_naive-x_true)
final_error = np.linalg.norm(x_final-x_true)
```

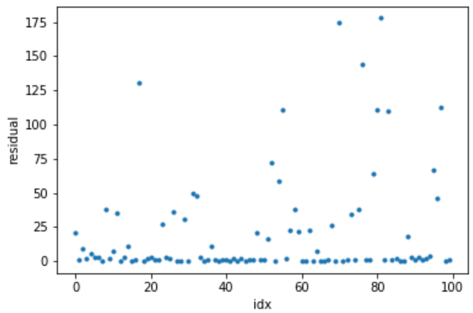

Figure 1: Estimator residual.

5.18 Multi-label support vector machine

(a)

Since

$$L(A,b) = \sum_{i=1}^{m}(1 + \sum_{k \neq y_i} f_k(x_i) - f_{y_i}(x_i))$$

we minimize $L(A,b) + \mu||A||_F^2$ which is

$$\boxed{\begin{array}{ll} \text{minimize} & \sum_i z_i + \mu||A||_F^2 \\ \text{subject to} & 1 + f_k(x_i) - f_{y_i}(x_i) \leq z_i,\ \forall k \neq y_i \\ & z_i \geq 0 \end{array}}$$

(b)

```
% data file for multi-label SVM problem
clear all;
randn('state', 0);
mTrain = 1000; % size of training data
mTest = 100; % size of test data
K = 10; % number of categories
n = 20; % number of features
A_true = randn(K, n);
b_true = randn(K, 1);
v = 0.2*randn(K, mTrain + mTest); % noise
data = randn(n, mTrain + mTest);
[~, label] = max(A_true * data + b_true * ones(1, mTrain + mTest) + v, [], 1);
% training data
x = data(:, 1:mTrain);
y = label(1:mTrain);
% test data
xtest = data(:, (mTrain+1):end);
ytest = label((mTrain+1):end);

%%
up = 10^2
lo = 10^(-2)

U = [];
E = [];

U = [0.01 0.05 0.1 0.2 0.5 1 2 5 10 20 50 75 100]
% This loop generates a new u value.
for u = 1:size(U,2)
```

```
    cvx_begin
        variable z(mTrain, 1)
        variable A(K, n)
        variable b(K, 1)
        minimize(sum(z) + U(u)*square_pos(norm(A,'fro')))
        subject to
        for i=1:mTrain
            for k=[1:y(i)-1 y(i)+1:K]
                1+(A(k,:)*x(:,i)+b(k))-(A(y(i),:)*x(:,i)+b(y(i))) <= z(i);
            end
            z(i) >= 0;
        end
        sum(b) == 0;

    cvx_end

    correct = 0
    y_pred = zeros(1,mTest);

    for i=1:mTest
        [~, y_pred(i)] = max(A*xtest(:,i) + b);
        if (y_pred(i) == ytest(i))
            correct = correct + 1;
        end
    end
    percent_correct = correct/mTest
    E = [E ; percent_correct]
end

plot(U,E)
```

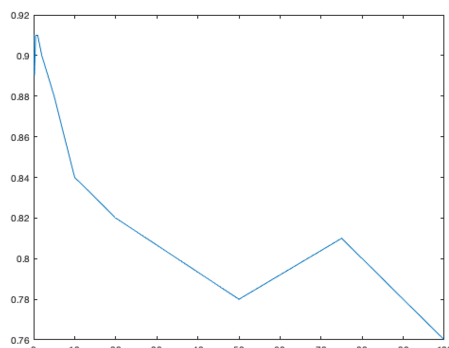

Figure 2: Estimator residual.

6.4 Maximum likelihood prediction of team ability

(a)

By the CDF def $\Phi(\frac{x-u}{\sigma})$, where $x = y_i(a_{i,j} - a_{i,k})$, we can get the total prob

$$p(y|a) = \prod_{i=1}^{n} \Phi(\frac{y_i(a_i - a_j)}{\sigma})$$

hence the log-likeihood function is

$$l(a) = \sum_{i}^{n} log(\Phi(\frac{y_i(a_i - a_j)}{\sigma}))$$

from here, the problem of finding the maximum likelihood estimate of team abilities is:

$$\boxed{\begin{array}{ll} \text{maximize} & \sum_i^n \log(\Phi(\frac{y_i(a_i - a_j)}{\sigma})) \\ \text{subject to} & 0 \preceq a \preceq 1 \end{array}}$$

(b) and (c)

```
Status: Solved
Optimal value (cvx_optval): +11.4487


a_hat =

    1.0000    0.0000    0.6829    0.3696    0.7946    0.5779    0.3795    0.0895    0.6736    0.5779


Pml =

  -20.6444
```

```
global n m m_test sigma train test;

A1 = sparse(1:m,train(:,1),train(:,3),m,n);
A2 = sparse(1:m,train(:,2),-train(:,3),m,n);
A = A1+A2;

cvx_begin
    variable a_hat(n)
    minimize(-sum(log_normcdf(A*a_hat/sigma)))
    subject to
    a_hat >= 0
    a_hat <= 1
cvx_end

a_hat = a_hat'
res = sign(a_hat(test(:,1))-a_hat(test(:,2)));
Pml = 1-length(find(res-test(:,3)))/m_test
```

6.6 Maximum likelihood estimation of an increasing nonnegative signal

(a)

$$\boxed{\begin{array}{ll} \text{minimize} & \sum_{t=2}^{N+2}(y(t) - \sum_{\tau=1}^{k} h(\tau)x(t-\tau))^2 \\ \text{subject to} & x(N) \geq x(N-1) \geq ... \geq x(1) \geq 0 \\ & x(t) = 0, t \leq 0 \end{array}}$$

(b)

```
import numpy as np
import cvxpy as cp
import matplotlib.pyplot as plt
# create problem data
N = 100;

# create an increasing input signal
xtrue = np.zeros((N,1))
xtrue[1:40] = 0.1
```

```
xtrue[50] = 2
xtrue[70:80] = 0.15;
xtrue[80] = 1
xtrue = np.cumsum(xtrue)

# pass the increasing input through a moving-average filter
# and add Gaussian noise
h = np.array([1, -0.85 ,0.7 ,-0.3])
k = h.shape[0]
yhat = np.convolve(h,xtrue)
y = yhat[:-3] + np.random.randn(N)
x = cp.Variable((100,),nonneg = True)
z = y[:,None] - cp.conv(h,x)[:-3]
objective = cp.Minimize(cp.sum_squares(z))
constraints = [cp.diff(x) >= 0]
prob=cp.Problem(objective,constraints=constraints)
prob.solve()

#plot
t = list(range(0,xtrue.size))
plt.plot(t,list(xtrue), color='red',label='x_true')
plt.plot(t,list(x.value), color='blue',label='x_hat')
plt.legend(loc="upper left")
plt.savefig('prob_66.png')
plt.show()
```

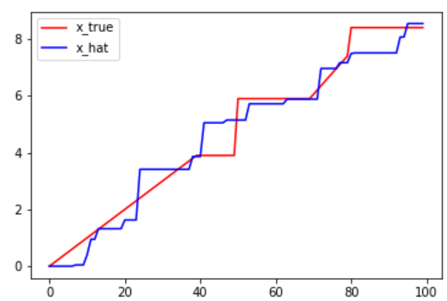

Figure 3: Maximum likelihood estimate x_{ml}, along with the true signal.

15.3 Utility versus latency trade-off in a network

(a)

Maximize the given log (concave) function:

$$\boxed{\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} log(f_j) \\ \text{subject to} & Rf \preceq c, \\ & f \succeq 0 \end{array}}$$

(b)

Latency is the sum of link delays when the link traffic t_i is zero. $d_i = \dfrac{1}{c_i}$ resulting in zero flow. The link delay vector can be written as:

$$(\frac{1}{c_1}, \dots, \frac{1}{c_m})$$

multply by R^T and find the maximum element to get L^{min}, we have

$$L^{min} = \max(R^T(\frac{1}{c_1}, \dots, \frac{1}{c_m})$$

This is to say minimum latency is the maximum of the flow latency.

(c)

Same as part (a), we maximimize the log function, while make sure the latency is min:

$$\boxed{\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} log(f_j) \\ \text{subject to} & Rf \preceq c, f \succeq 0 \\ & \sum_{i=1}^{m} \dfrac{R_{ij}}{c_i - r_i^T f} \leq L, j = 1, \dots, n \end{array}}$$

(d)

lstinputlisting[language=matlab]prob157.m