Practice Questions for CS 181, Midterm 2 (Spring 2017)

April 23, 2017

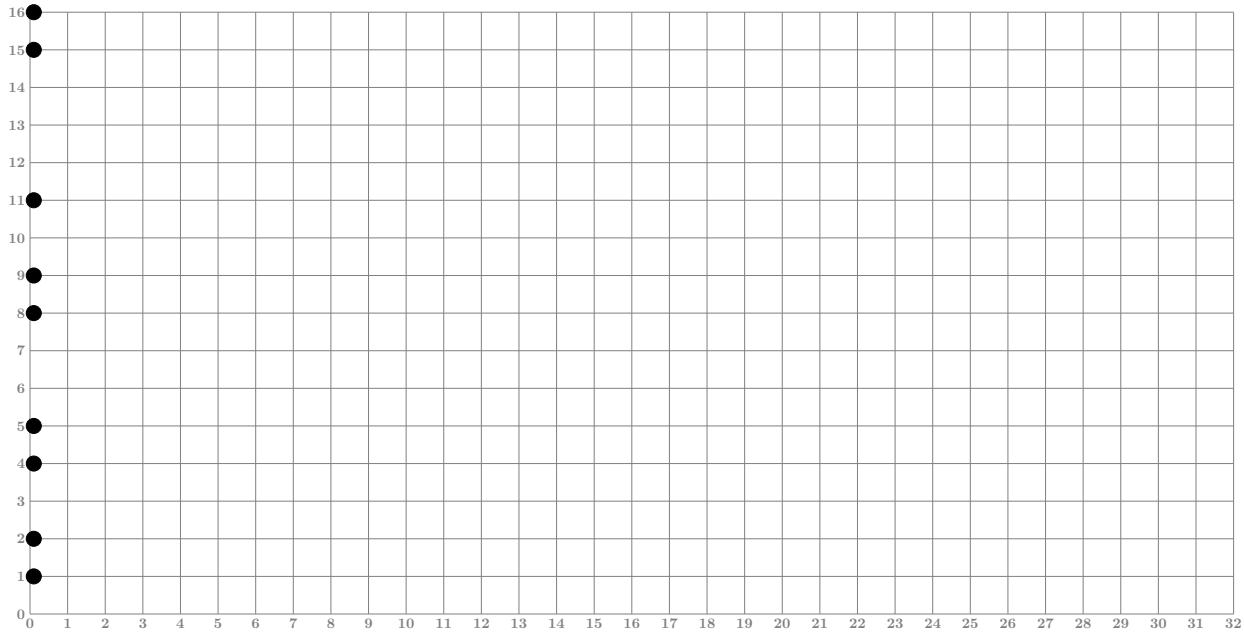
These practice questions are illustrative of the kinds of understanding that you should expect to be tested on the midterm. If anything they are slightly more difficult than the questions on the exam.

1. **Hierarchical Agglomerative Clustering**

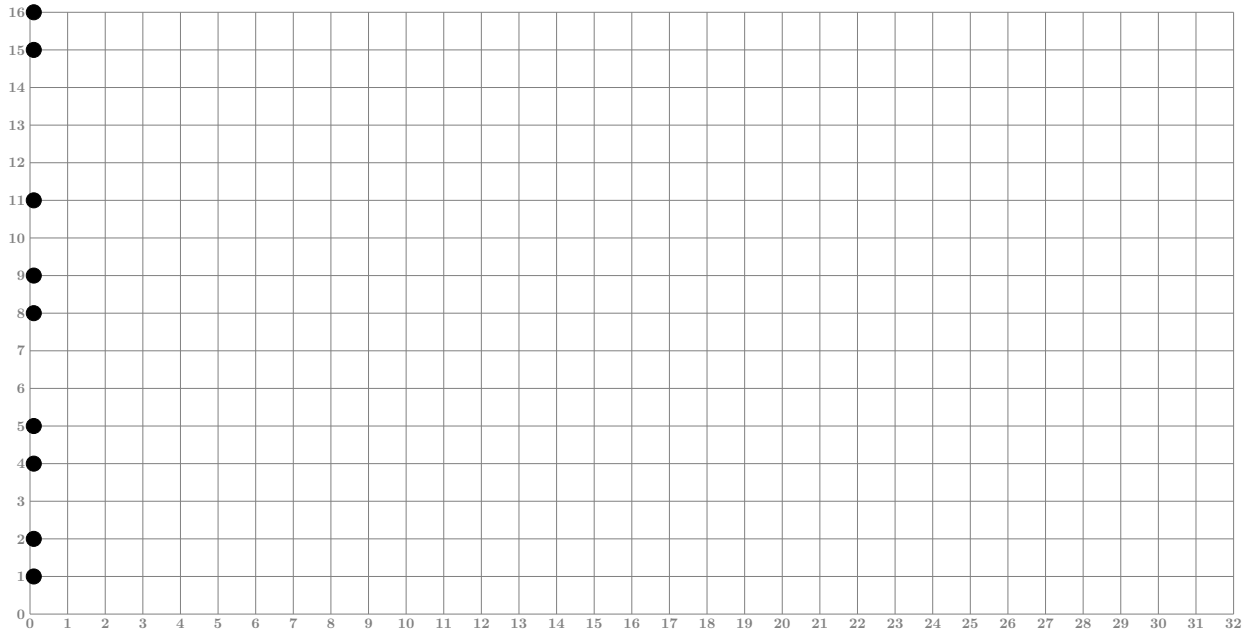
Consider nine points $x_1, \ldots, x_9$ in $\mathbb{R}$ shown below, where the y-axis provides their values. We define $d(x, x') = |x - x'|$, and consider two different cluster distances.

Draw the dendrogram for the data. Join together clusters one per step (on the horizontal-axis), breaking ties towards joining lower x values first. In the top figure, use the min-linkage distance and in the bottom figure use the max-linkage distance.

(a) Min Linkage:

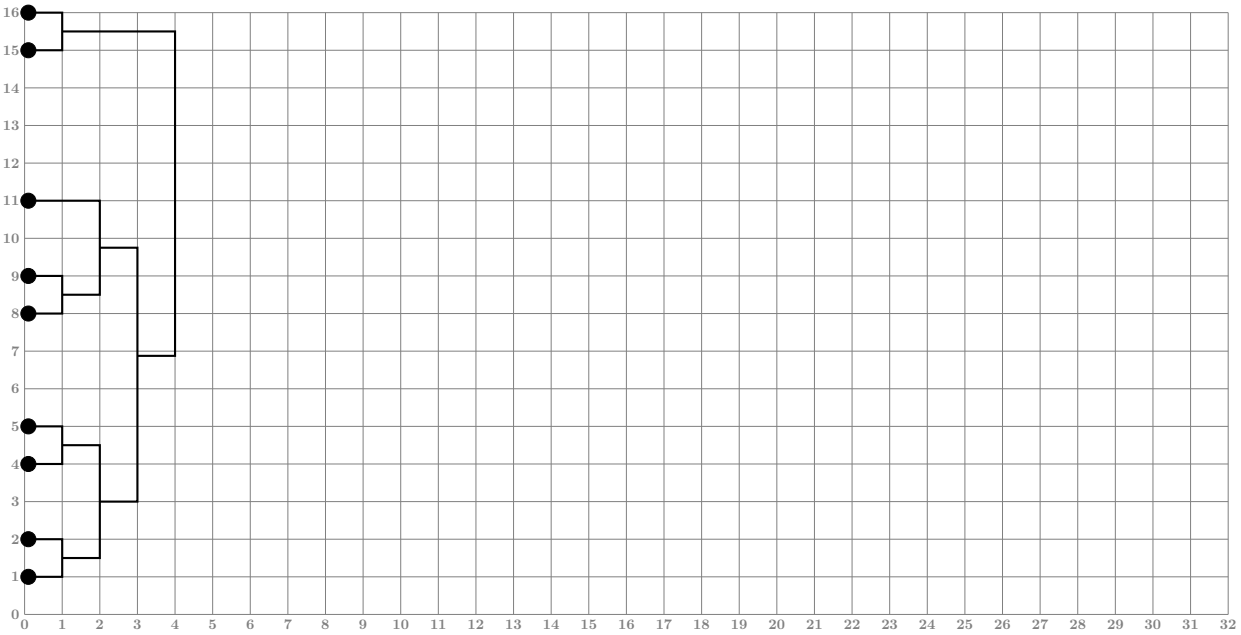


(b) Max Linkage:

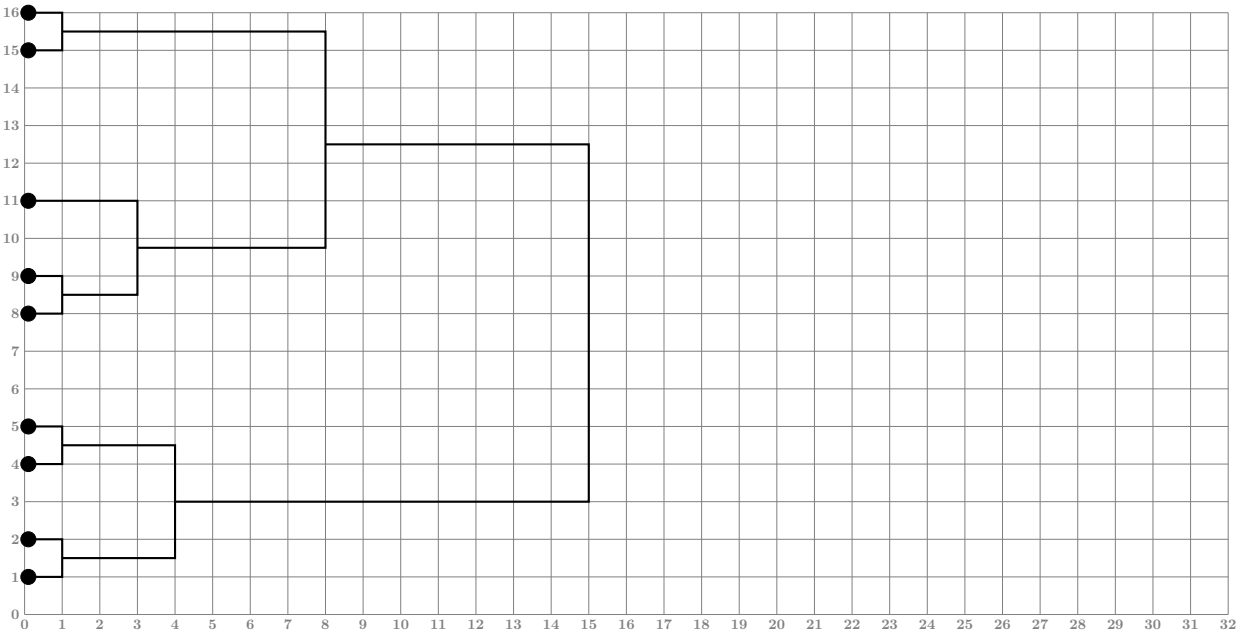


Solution:

(a) Min Linkage:

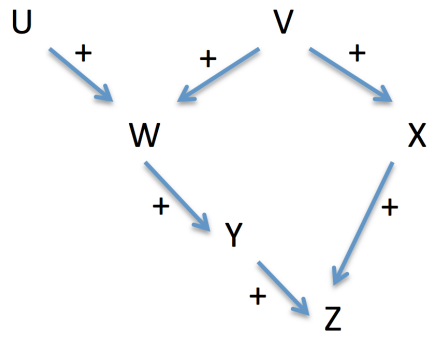


(b) Max Linkage:



2. **Bayesian networks**

Consider the following Bayesian network, where the variables are all Boolean.



The '+' annotations indicate the direction of the local effect; e.g., the '+' from U to W means that for each value v of V,

$$p(W = true \,|\, U = true, V = v) > p(W = true \,|\, U = false, V = v).$$

For each of the following questions, select one of the following, and also state which (if any) paths are blocked:

$=$ if the two probabilities are necessarily equal;
$<$ if the first probability is necessarily smaller;
$>$ if the first probability is necessarily larger;
? if none of these cases hold.

(a) $p(V = true \mid Y = false)$ $p(V = true \mid Y = true)$

(b) $p(V = true \mid Z = false)$ $p(V = true \mid Z = true)$

(c) $p(U = true \mid W = true, Y = false)$ $p(U = true \mid W = true, Y = true)$

(d) $p(Y = true \mid Z = true, X = false)$ $p(Y = true \mid Z = true, X = true)$

(e) $p(U = true \mid Y = true, Z = false)$ $p(U = true \mid Y = true, Z = true)$

Solution.

(a) $p(V = true \mid Y = false) \; < \; p(V = true \mid Y = true)$
Path V-W-Y not blocked, and positive effect. path V-X-Z-Y blocked at Z.

(b) $p(V = true \mid Z = false) \; < \; p(V = true \mid Z = true)$
Paths V-W-Y-Z and V-X-Z not blocked, and positive effect.

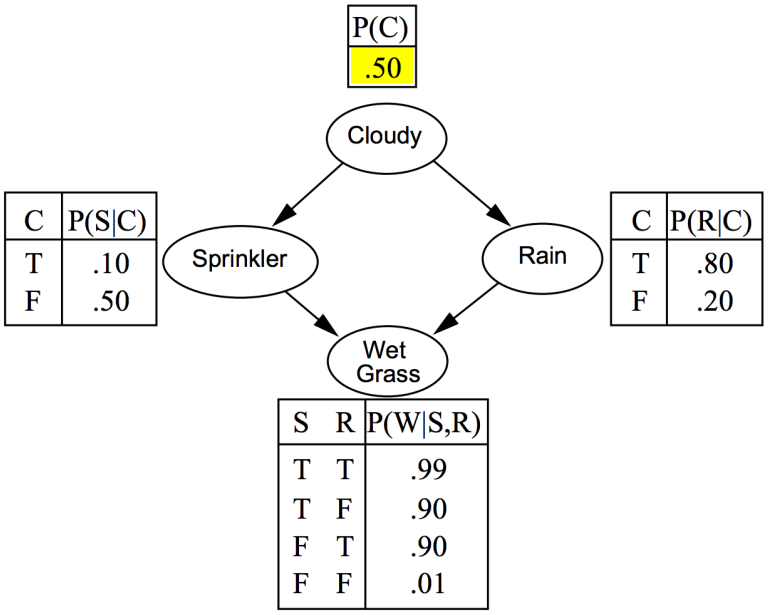
(c) $p(U = true \mid W = true, Y = false) \; = \; p(U = true \mid W = true, Y = true)$
W in evidence (since its value is fixed). Is $I(U, Y \mid W)$? yes. Path $U - W - Y$ blocked at W, path $U - W - V - X - Z - Y$ blocked at Z (converging arrows!).

(d) $p(Y = true \mid Z = true, X = false)$? $p(Y = true \mid Z = true, X = true)$
Z in evidence. Is $I(Y, X \mid Z)$? path $Y - Z - X$ not blocked. path $Y - W - V - X$ not blocked. effect could go either way, because explaining away through Z but positive effect through V.

(e) $p(U = true \mid Y = true, Z = false) \; > \; p(U = true \mid Y = true, Z = true)$
Y in evidence. Is $I(U, Z \mid Y)$? No! path $U - W - Y - Z$ blocked at Y. path $U - W - V - X - Z$ is NOT blocked at W because Y is in the evidence. Thus, we have the explaining away pattern.
[Intuition: conditioned on $Y = true$, then if we also know Z is true this provides information that reduces our belief that $U = true$ (since both are competing explanations of why $Y = true$).]

3. **Bayesian networks**

Consider this example of a Bayesian network with binary variables. It models a garden lawn and whether or not the grass is wet.



(a) Consider an alternate variable ordering, S, C, R, W. Use the given Bayesian network to determine which conditional indpendence properties (if any) hold amongst preceding variables in this alternate ordering, and construct the Bayesnet with a min number of edges for this alternate ordering. (Don't worry about specifying the conditional probability tables.)

(b) Is this new Bayesian network a correct model of the distribution? Which network is preferable?

(c) Going back to the original network, what is the probability that it is not cloudy, rains, sprinkler doesn't run, and grass is wet?

(d) In the original network: write down the first two steps of variable elimination for $p(W)$, eliminating C and then S. Perform the numerical calculations!

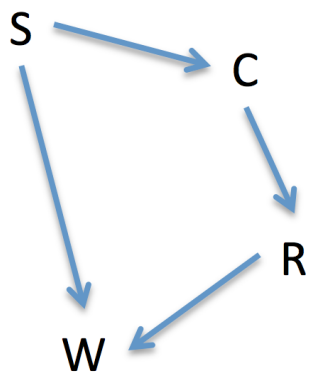
(e) Explain how Gibbs sampling works for the inference problem $p(C \mid W = true)$.

Solution.

(a) Construct Bayesnet for ordering S, C, R, W. [I'm choosing to write this out in full detail so that you understand the steps. The conditional independence checks use the original network.]

- Variable S: just add the node
- Variable C: check the following
  - can we add no in-edges? Is $I(C, S)$? no!

  Conclude that we need edge $S \to C$.
- Variable R: check the following
  - can we add no in-edges? Is $I(R, C)$? no!
  - can we add just one in-edge? Suppose we add C to R. Is $I(R, S \mid C)$? yes! (in original network: R-C-S blocked at C, R-W-S blocked at W.) Conclude can just add C to R.
- Variable W: check the following
  - can we add no in-edges? Is $I(W, R)$? No!
  - can we add just one in-edge?
    what if we just had the R-W edge? Is $I(W, C \mid R)$? no, path $C - S - W$ not blocked in original network.
    what if we just had the C-W edge? Is $I(R, W \mid C)$? no!
    what if we just had the S-W edge? Is $I(R, W \mid S)$? no!
  - can we add just two in-edges? Consider edges S-W and R-W. Is $I(C, W \mid S, R)$? checking original network, yes! paths CRW and CSW are both blocked. Conclude that it is sufficient to just add these two edges.



(b) yes, it is correct (networks are correct for any ordering, but this can affect compactness and interpretability.) It turns out that both networks have the same number of parameters ($1 + 2 + 2 + 4 = 9$) and thus one is not preferred over the other for reasons of compactness (compact networks are generally prefered because they have fewer parameters, need less data to learn, and are less likely to overfit if data is noisy). But we might still prefer the first network because it is more interpretable since it is constructed in a causal order (the $S \to C$ edge in the new network is a bit hard to interpret!).

(c)

$$\begin{aligned} p(\neg C, R, \neg S, W) &= p(\neg C)p(\neg S \mid \neg C)p(R \mid \neg C)p(W \mid \neg S, R) \\ &= 0.5 \cdot 0.5 \cdot 0.2 \cdot 0.9 = 0.045 \end{aligned}$$

(d)

$$\begin{aligned} p(W) &= \sum_{c,s,r} p(C)p(S \mid C)p(R \mid C)p(W \mid S, R) \\ &= \sum_{s,r} p(W \mid S, R) \sum_{c} p(C)p(S \mid C)p(R \mid C) \\ &= \sum_{s,r} p(W \mid S, R)\psi_1(S, R) \\ &= \sum_{r} \sum_{s} p(W \mid S, R)\psi_1(S, R) \\ &= \sum_{r} \psi_2(W, R) \end{aligned}$$

Calculations:

ψ_1		$C = false$				$C = true$				
S	R	$p(C)$	$p(S \mid C)$	$p(R \mid C)$	$\times$	$p(C)$	$p(S \mid C)$	$p(R \mid C)$	$\times$	$\sum$
T	T	0.5	0.5	0.2	0.05	0.5	0.1	0.8	0.04	0.09
T	F	0.5	0.5	0.8	0.2	0.5	0.1	0.2	0.01	0.21
F	T	0.5	0.5	0.2	0.05	0.5	0.9	0.8	0.36	0.41
F	F	0.5	0.5	0.8	0.2	0.5	0.9	0.2	0.09	0.29

ψ_2		$S = false$			$S = true$			
W	R	$p(W \mid S, R)$	$\psi_1(S, R)$	$\times$	$p(W \mid S, R)$	$\psi_1(S, R)$	$\times$	$\sum$
T	T	0.9	0.41	0.369	0.99	0.09	0.0891	0.4581
T	F	0.01	0.29	0.0029	0.9	0.21	0.189	0.1919
F	T	0.1	0.41	0.041	0.01	0.09	0.0009	0.0419
F	F	0.99	0.29	0.2871	0.1	0.21	0.021	0.3081

(e) Gibbs sampling for $p(C \mid W = true)$ works by randomly initializing the variables C, S, R, holding $W = true$, and then repeating:

- pick one of C, S, R uniformly at random. call this X_j
- sample the value of X_j from the conditional distribution $p(X_j \mid X_{-j})$ where X_{-j} is the current value of the other random variables.

After long enough, the empirical marginal distribution on C from this Markov chain will converge to the correct $p(C \mid W = true)$.

4. **Markov Decision Process (modeling).**

   You are asked to develop a Markov Decision Process (MDP) to be used for the control of a single elevator. There are three floors, three buttons inside the car, and a single call button outside on each floor. The door of the elevator opens and closes. The agent here is the elevator itself, and the aim of the system is to get passengers to their appropriate floors.

   Describe in words the states, actions, reward function, and transition model for a suitable MDP model. There is no single correct answer here, but a proper response will specify the size of each set and make the reward function clear. An optimal policy over this MDP should lead to an intuitively efficient system.

Solution.

State $s = (F, R, C)$ is factored:

- (floor) $F \in \{1, 2, 3\}$
- (requests from inside car) $R \subseteq \{1, 2, 3\}$
- (call from outside elevator) $C \subseteq \{1, 2, 3\}$

initial state: $F = 1, R = \emptyset, C = \emptyset$. Note: we choose not to model whether the door is open or closed to keep things simple, nor where the elevator was in the past, nor how long someone has been waiting.

Actions are:

- open-close (modeled as a single action, for simplicity)
- up (available if $F < 3$)
- down (available if $F > 1$)
- nothing

Note: 'nothing' action is to stop the elevator continually doing something

Reward

$$r(s, \text{open-close}) = \begin{cases} 1 & \text{if } (F \in C) \vee (F \in R) \\ -0.01 & \text{o.w.} \end{cases}$$

$$r(s, up) = r(s, down) = -0.01$$

$$r(s, nothing) = 0$$

Note: only need reward when the open-close action is taken and the elevator is at a floor where it has been requested to go, or called from. Other rewards negative to dissuade it from doing things without need.

Transition

Define random variables $X_j \sim [\{j\} : 0.5, \emptyset : 0.5]$ that take on set value $\{j\}$ w.p. 0.5, and emptyset otherwise.

Define random variable $X(C, F)$ that is $\emptyset$ if $F \notin C$ (was not called to this floor), or uniformly sampled from $\{1, 2, 3\} \setminus F$ otherwise (the floor the rider wants to go to.)

Can now define transition model for $s' = (F', R', C')$ reached after action a in state s. Do this in factored way. First, in regard to floor:

- if $a = up$: then $F' = F + 1$
- if $a = down$: then $F' = F - 1$
- if $a = nothing$ or $a =$ open-close then $F' = F$

Second, in regard to calls:

- if $a \in \{up, down, nothing\}$ then $C' := C \cup X_1 \cup X_2 \cup X_3$
- if $a =$ open-close then $C' := (C \setminus F) \cup X_1 \cup X_2 \cup X_3$

Note: this models the effect of open-close being to drop any call button at that floor. But still, someone else can call before the next period.

Third, in regard to requests:

- if $a \in \{up, down, nothing\}$ then $R' := R$
- if $a =$ open-close then $R' := (R \setminus F) \cup X(C, F)$

Note: this models the effect of open-close being to drop any request that had been at that floor, and add a request only if there had been a call at the floor, and only going to a different floor.

5. **Alternate Reward Function for MDPs**

We have been assuming that the reward function has the form $r(s, a)$, i.e., it only depends on the current state and action. In the discounted infinite-horizon case, we use this when computing the optimal value function via:

$$V'(s) \leftarrow \max_a \left[r(s,a) + \gamma \sum_{s'} p(s' \mid s, a) V(s') \right]$$

Now, imagine that we have a reward function that depends on both the current state *and* the next state, i.e., $r(s, a, s')$.

(a) **Write an expression for the value iteration step that incorporates this alternative type of reward.**

(b) **Argue that this approach is strictly more general, by showing that any standard MDP (with $r(s, a)$) can be written in this form.**

Solution.

(a) Value iteration step:

$$V'(s) \leftarrow \max_a \left[\sum_{s'} p(s' \mid s, a)(r(s, a, s') + \gamma V(s')) \right]$$

(b) More general, because we can always define $r(s, a, s') = r(s, a)$, for all values of s'.

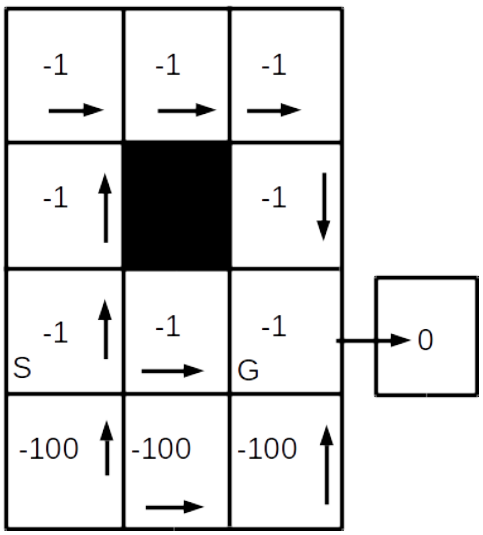
6. **Planning in MDPs**

Consider a gridworld with the layout below. From each square, the agent may move into an adjoining square (up, down, left, right) or stay in place. Actions are deterministic, that is, they always have their intended effect. We use an infinite horizon with discount γ. We always start in state marked with an S. Upon reaching the state marked G the agent transitions into an absorbing state where it stays forever. The rewards associated with a state are the reward for taking any action in that state. Assume a discount factor $\gamma = 1$.

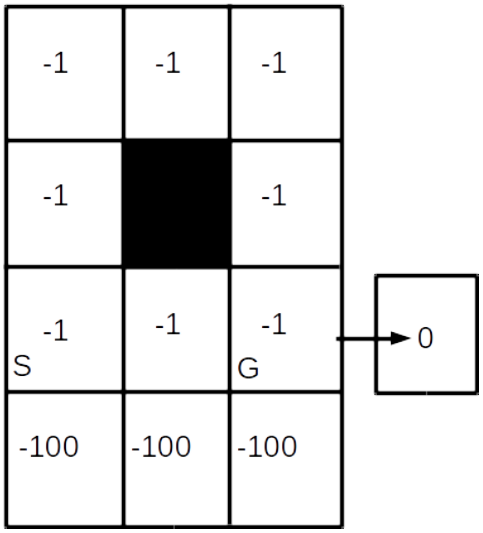
Recall the improvement step in policy iteration:

$$\pi'(s) \leftarrow \arg\max_{a \in A} \left[r(s,a) + \gamma \sum_{s' \in S} p(s' \mid s, a) V^{\pi}(s') \right], \quad \forall s$$

(a) Suppose that we follow the policy given by the arrows. What is the MDP value of each state under this policy? You may draw the values directly on the figure.

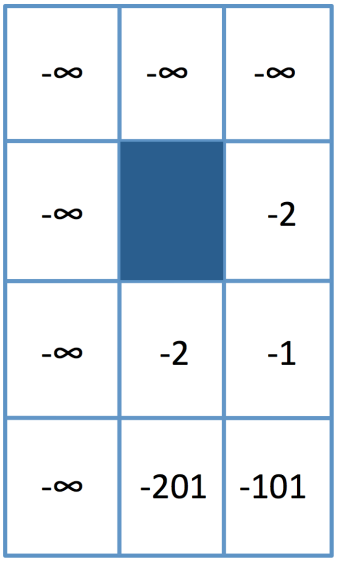


(b) Can this policy be improved? Draw the adjusted policy based on one round of policy iteration and compute the new value function in each state. Is the new policy optimal?

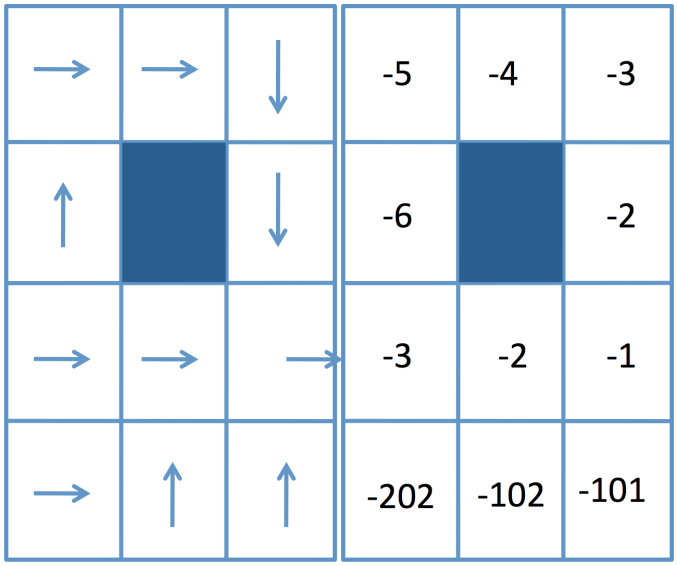


Solution.

(a) See the following figure for the MDP value of each state under this policy. [Note: it is $-\infty$ in states for which the policy does not escape to the goal state because $\gamma = 1$ and thus there is no discount.]



(b) See the following figure for the adjusted policy based on one round of policy iteration, as well as the new value function in each state. [Note: we left the action unchanged if there was no better action relative to the last round of value iteration. See how the policy is better, but not optimal. In particular, the policy is moving right in the bottom-right state, whereas it would be optimal to move up. Similarly, it is moving up in the state to the right of the missing position, when it should move down. Finally,becaue this is value iteration, we do policy evaluation to compute the new values in each state for this new policy.]



7. **Reinforcement learning**

The update rule for **SARSA** learning is:

$$w_{s,a} \leftarrow Q(s, a; \mathbf{w}) - \eta(Q(s, a; \mathbf{w}) - [r + \gamma Q(s', \pi(s'); \mathbf{w})])$$

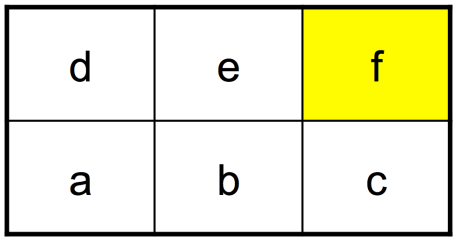
(a) **Describe** the aim of the temporal difference update. If π is a greedy policy, what property will hold at convergence?

(b) **How** do we select state s, action a, reward r, state s' and action $\pi(s')$ in SARSA learning?

(c) What is meant by 'on-policy' and 'off-policy', and is SARSA an on-policy or off-policy method?

(d) What does it mean to **exploit** in the context of reinforcement learning?

(e) Consider this simple MDP world, where the reward is 100 for any action in state f and 0 in all other states, and actions are deterministic (thus 'up' always moves 'up').



Assume the Q-values are initialized to 0, and the agent is initially in state c. What are the updates made by SARSA following each action (for $\eta = 0.9, \gamma = 0.9$). Assume that no update is possible until the values of s, a, r, s', a' are all well-defined.

i. up (to state f)

ii. left (to state e)

iii. right (to state f)

iv. down (to state c)

Solution.

(a) The temporal difference update is derived from the Bellman equations for the Q-function. If we use a greedy policy for π, i.e. $\pi(s) = \arg\max_a Q(s,a)$, upon convergence the Bellman equations will hold, which implies the policy is optimal.

(b) s is current state, a is selected via ϵ-greedy, reward is given by the MDP for $r(s,a)$, state s' is given by the MDP for $p(s' \mid s,a)$, action $\pi(s')$ is the one selected via ϵ-greedy.

(c) On-policy: the Q-values learned will correspond to those of the policy we follow while learning. off-policy: the Q-values learned will update towards those that correspond to the optimal policy. SARSA is on-policy.

(d) Exploit: this means to follow $\max_a Q(s,a)$ in state s, rather than also explore (e.g., by taking some other action with small probability $\epsilon > 0$).

(e) SARSA updates:

- up (to state f): have $s = c, a = up, r = 0, s' = f, a' =$??, and cannot do an update yet
- left (to state e): now have $s = c, a = up, r = 0, s' = f, a' = \textit{left}$, and thus can do update to $Q(c, up)$:

$$Q'(c, up) \leftarrow Q(c, up) - 0.9(Q(c, up) - (0 + 0.9Q(f, \textit{left}))) = 0 - 0.9(0 - (0 + 0.9(0))) = 0$$

- right (to state f): now have $s = f, a = \textit{left}, r = 100, s' = e, a' = right$, and thus can do update to $Q(f, \textit{left})$:

$$\begin{aligned} Q'(f, \textit{left}) &\leftarrow Q(f, \textit{left}) - 0.9(Q(f, \textit{left}) - (100 + 0.9Q(e, right))) \\ &= 0 - 0.9(0 - (100 + 0.9(0))) = 90 \end{aligned}$$

- down (to state c): now have $s = e, a = right, r = 0, s' = f, a' = down$, and thus can do update to $Q(e, right)$:

$$\begin{aligned} Q'(e, right) &\leftarrow Q(e, right) - 0.9(Q(e, right) - (0 + 0.9Q(f, down))) \\ &= 0 - 0.9(0 - (0 + 0.9(0))) = 0 \end{aligned}$$

8. **K-Means**

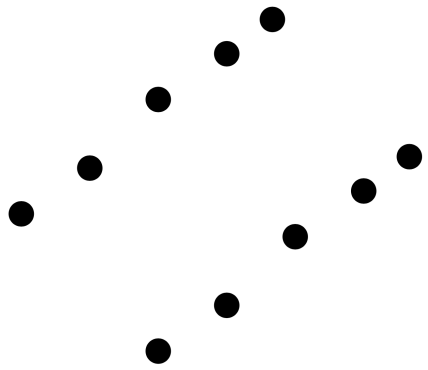
In K-Means, we are given a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and a fixed number of clusters K. Our aim is to find cluster centers $\mathbf{z}_1, \ldots, \mathbf{z}_K$ that represent the data.

(a) Formally define the K-Means loss function.

(b) What two steps does Lloyd's algorithm repeat in order to find a good clustering?

(c) What is the asympotic run-time of each step of Lloyd's algorithm, as a function of the number of examples n and the number of clusters K?

(d) Given data that falls on two parallel diagonal lines as shown below, can Lloyd's algorithm find two clusters, such that each line is in one of the clusters?



Solution.

(a) Objective is to find prototypes and an assignment to minimize

$$\sum_{i=1}^{n}\sum_{k=1}^{K} r_{ik}||\mathbf{x}_i - \boldsymbol{\mu}_k||$$

were $\mathbf{z} = \sqrt{\mathbf{z}^\top \mathbf{z}}$.

(b) Step 1: assign each example to the closest prototype ; step 2: for each cluster k, set $\boldsymbol{\mu}_k$ to the centroid of the assigned examples

$$\boldsymbol{\mu}_k := \frac{1}{n_k}\sum_i r_{ik}\mathbf{x}_i$$

where $n_k = \sum_i r_{ik}$ and $r_{ik} = 1$ if i assigned to k, and 0 otherwise.

(c) Step 1 takes time nK because each example is checked for the closest of K prototypes. Step 2 takes time n because each example is in exactly one cluster, and this is used once in the averaging step. Thus, complexity is $n + nK$ each time step 1 and 2 are run, or $O(nK)$. [Note: we chose to ignore the dependence on number of attributes m, and if writing as a function of number of iterations, this would be come $O(nKT)$.]

(d) It's a bit hard to tell precisely, but I'd say 'yes.' Think about the two assignments, one to the top line and one to the bottom line. We need to check this is a stable assignment. In particular, is each point in the cluster closest to its centroid than the centroid of the other cluster. This looks to be true and thus K-means would converge on such a clustering, and thus it can find this clustering (for an appropriate initialization! e.g., the one that gets lucky and makes this initialization of prototypes.)

9. **Hidden Markov Models**

Consider a weather domain, with outputs $\mathbf{x}_t \in \{D, R\}$ (no rain, rain) and state $\mathbf{s}_t \in \{C, S\}$ (cloud, sun). Assume the following parameters:

- initial prob (θ): $p(\mathbf{s}_1 = C; \theta) = 0.7$
- transition ($\mathbf{T}$)

$p(\mathbf{s}_{t+1} \mid \mathbf{s}_t; \mathbf{T})$		Next State C	Next State S
State	C	0.8	0.2
	S	0.1	0.9

- output ($\boldsymbol{\pi}$)

$p(\mathbf{x}_t \mid \mathbf{s}_t; \boldsymbol{\pi})$		Output D	Output R
State	C	0.25	.75
	S	0.6	0.4

(a) For a general HMM, if the total number of timesteps is n and $t < n$ is a timestep in the middle of the sequence, why is $p(\mathbf{s}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) \neq p(\mathbf{s}_t \mid \mathbf{x}_1, \ldots, \mathbf{x}_t)$? (An informal answer is fine.)

(b) (Forward-backward algorithm). Now suppose we have a single sequence of observed data with $\mathbf{x}_1 = R$, $\mathbf{x}_2 = R$ in the weather domain.

We can calculate:

$$\alpha_1(\mathbf{s}_1) = \begin{cases} 0.525 & \text{, if } \mathbf{s}_1 = C \\ 0.12 & \text{, if } \mathbf{s}_1 = S \end{cases}$$

Use

$$\alpha_2(\mathbf{s}_2) = p(\mathbf{x}_2 \mid \mathbf{s}_2) \sum_{\mathbf{s}_1} p(\mathbf{s}_2 \mid \mathbf{s}_1) \alpha_1(\mathbf{s}_1)$$

to compute the α_2-values.

(c) We have $\beta_2(\mathbf{s}_2) = 1$. In addition, we can calculate:

$$\beta_1(\mathbf{s}_1) = \begin{cases} 0.68 & \text{, if } \mathbf{s}_1 = C \\ 0.435 & \text{, if } \mathbf{s}_1 = S \end{cases}$$

Use these quantities, and

$$p(\mathbf{s}_t \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_t) \propto \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)$$

to infer the values of $p(\mathbf{s}_1 \,|\, \mathbf{x}_1, \mathbf{x}_2)$ and $p(\mathbf{s}_2 \,|\, \mathbf{x}_1, \mathbf{x}_2)$.

(d) Use $p(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{s}_t} \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)$ to calculate the likelihood of the data.

Solution.

(a) This is because the additional observations from $t+1$ to n may also be informative as to the hidden state at period t. For example, the observation at $\mathbf{x}_{t+1}$ might only be possible if $\mathbf{s}_t$ has a value that allows for $\mathbf{s}_{t+1}$ to have a value that can generate $\mathbf{x}_{t+1}$.

(b) Calculations for $\alpha_2(\mathbf{s}_2)$ values are:

$\mathbf{s}_1$	$\mathbf{s}_2$	$p(\mathbf{s}_2 \mid \mathbf{s}_1)$	$p(R \mid \mathbf{s}_2)$	$\alpha_1(\mathbf{s}_1)$	$\times$
C	C	0.8	0.75	0.525	0.315
C	S	0.2	0.4	0.525	0.042
S	C	0.1	0.75	0.12	0.009
S	S	0.9	0.4	0.12	0.0432

and then summing out $\mathbf{s}_1$, we obtain

$\mathbf{s}_2$	$\alpha_2(\mathbf{s}_2)$
C	0.315+0.009=0.324
S	0.042+0.0432=0.0852

(c) For $t=1$, we have

$$p(\mathbf{s}_1 \mid R, R) \propto \alpha_1(\mathbf{s}_1)\beta_1(\mathbf{s}_1)$$

For $\mathbf{s}_1 = C$:

$$\alpha_1(C)\beta_1(C) = (0.525)(0.68) = 0.357$$

For $\mathbf{s}_1 = S$:

$$\alpha_1(S)\beta_1(S) = (0.12)(0.435) = 0.0522$$

We conclude that $p(\mathbf{s}_1 = C \mid R, R) \approx 0.872$ and $p(\mathbf{s}_1 = S \mid R, R) \approx 0.128$.

For $t=2$, we have

$$p(\mathbf{s}_2 \mid R, R) \propto \alpha_2(\mathbf{s}_2)\beta_2(\mathbf{s}_2)$$

For $\mathbf{s}_2 = C$:

$$\alpha_2(C)\beta_2(C) = (0.324)(1) = 0.324$$

For $\mathbf{s}_2 = S$:

$$\alpha_2(S)\beta_2(S) = (0.0852)(1) = 0.0852$$

We conclude that $p(\mathbf{s}_2 = C \mid R, R) \approx 0.792$ and $p(\mathbf{s}_2 = S \mid R, R) \approx 0.208$.

(d) The likelihod of the data is

$$p(R, R) = \sum_{\mathbf{s}_1} \alpha_1(\mathbf{s}_1)\beta_1(\mathbf{s}_1) = (0.525)(0.68) + (0.12)(0.435) = 0.4092.$$

Equivalently, we could calculate using $t=2$ alpha and beta values, and get

$$p(R, R) = \sum_{\mathbf{s}_2} \alpha_2(\mathbf{s}_2)\beta_2(\mathbf{s}_2) = (0.324)(1) + (0.0852)(1) = 0.4092$$

10. **Mean of a Mixture Model**

We are given a mixture model of the form,

$$p(\mathbf{x} \mid \{\boldsymbol{\pi}_k\}_{k=1}^{c}, \boldsymbol{\theta}) = \sum_{k=1}^{c} \theta_k \, p(\mathbf{x} \mid \mathbf{z} = C_k, \boldsymbol{\pi}_k)$$

where $\mathbf{x} \in \mathbb{R}^m$.

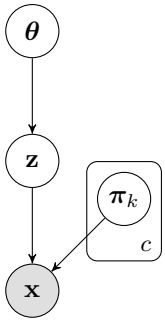
(a) **Draw** a graphical model with plates to show the form of the mixture distribution.

(b) The mean of the c'th component distribution with parameters $\boldsymbol{\pi}_k$ is given by $\boldsymbol{\mu}_k$. Show that the mean of the overall mixture is given by

$$\mathbb{E}[\mathbf{x}] = \sum_{k=1}^{c} \theta_k \boldsymbol{\mu}_k.$$

Solution:

(a) We will draw it using a single $\mathbf{x}$ variable.



(b) Let's explicitly introduce the latent variable $\mathbf{z}$,

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\{\boldsymbol{\pi}_k\}_{k=1}^c,\boldsymbol{\theta})}[\mathbf{x}] = \sum_{k=1}^{c} p(\mathbf{z}=C_k|\boldsymbol{\theta})\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z}=C_k,\boldsymbol{\pi}_k)}[\mathbf{x}] = \sum_{k=1}^{c} \theta_k \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\mathbf{z}=C_k,\boldsymbol{\pi}_k)}[\mathbf{x}] = \sum_{k=1}^{c} \theta_k \boldsymbol{\mu}_k$$

Note that step two comes from basic properties of conditional expectations, or could be found explicitly from summing out over possible $\mathbf{z}$ values. It does not however require knowing anything about the form of $p(\mathbf{x}|\mathbf{z}=C_k,\boldsymbol{\pi}_k)$, which could be an arbitrary distribution with mean $\boldsymbol{\mu}_k$.

11. **Expectation Maximization**

(This problem is a bit harder than we would ask on the exam, but it is a good review of the EM algorithm.)

Imagine that we have a collection of n binary images $\mathbf{x}_1, \ldots, \mathbf{x}_n$, each of which is 5×5. We treat each image as a 25-dimensional binary vector where image i is denoted as a vector $\mathbf{x}_i$ and the jth pixel is $x_{i,j}$.

We wish to estimate the parameters of a mixture model, and use a product of Bernoulli distributions for each component in the mixture:

$$p(\mathbf{x}_i \mid \boldsymbol{\mu}_k) = \prod_{j=1}^{25} \mu_{k,j}^{x_{i,j}} (1 - \mu_{k,j})^{1 - x_{i,j}} \qquad \mathbf{x}_i \in \{0,1\}^{25} \qquad \boldsymbol{\mu}_k \in (0,1)^{25} .$$

Each of the C mixture components has a parameter $\boldsymbol{\mu}_k$ and each dimension $\mu_{k,j}$ is a Bernoulli distribution parameter that specifies the probability of pixel j being black. The (known) mixture weights are $\{\theta_k\}_{k=1}^{c}$. You want to learn the parameters $\{\boldsymbol{\mu}_k\}_{k=1}^{c}$, so you decide to use the expectation-maximization algorithm. (There are four parts to this, make sure you do parts (c) and (d) on the next page.)

(a) Write down the probability of generating a single image $\mathbf{x}$, i.e.,

$$p(\mathbf{x} \mid \{\boldsymbol{\mu}_k\}_{k=1}^{c}, \boldsymbol{\theta})$$

.

(b) This is an example of a latent variable model. What is the form of the latent variables? Draw the plate diagram for this problem indicating what is known and unknown.

(c) In the E-step, you improve the estimate of the latent variables, fixing the parameters $\{\boldsymbol{\mu}_k\}_{k=1}^c$. Derive the update for this approximation. In particular compute the $\mathbf{q}$ vectors.

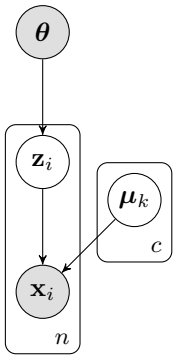
(d) In the M-step, you update the parameters $\{\boldsymbol{\mu}_k\}_{k=1}^c$. Derive the update for the parameters. Assuming that you have the $\mathbf{q}$ vectors.

Solutions:

(a) Sum out over possible values of the latent variables $\mathbf{z}$ and then just use the properties of the problem.

$$p(\mathbf{x} \,|\, \{\boldsymbol{\mu}_k\}_{k=1}^c, \boldsymbol{\theta}) = \sum_{k=1}^{c} p(\mathbf{z} = C_k| \, \boldsymbol{\theta}) p(\mathbf{x} \,|\, \mathbf{z} = C_k, \boldsymbol{\mu}_k) = \sum_{k=1}^{c} \theta_k \prod_{j=1}^{25} \mu_{k,j}^{x_j} (1 - \mu_{k,j})^{1-x_j}$$

(b) Plate diagram. Note that the mixture weights for this problem are known (so grey).



(c) We will use very similar methodology to the topic modeling homework problem. We need to compute "expected counts"

$$q_{i,k} = p(\mathbf{z}_i = C_k \,|\, \mathbf{x}_i, \{\boldsymbol{\mu}_k\}_{k=1}^c, \boldsymbol{\theta})$$

We do this by applying Bayes' rule.

$$q_{i,k} = p(\mathbf{z}_i = C_k \,|\, \mathbf{x}_i, \{\boldsymbol{\mu}_k\}_{k=1}^c, \boldsymbol{\theta}) \quad = \quad \frac{1}{Z} p(\mathbf{z}_i = C_k \,|\, \boldsymbol{\theta}) p(\mathbf{x}_i \,|\, \mathbf{z}_i = C_k, \boldsymbol{\mu}_k)$$

The first term is known since $\boldsymbol{\theta}$ is given. The second term can be computed using the formula in the problem assuming current $\boldsymbol{\mu}_k$ values. Finally the normalization term can be computing as $Z = \sum_\ell q_{i,\ell}$.

(d) Assuming we know the values of $q_{i,k}$ for all i, k, we want to reestimate the Bernoulli parameters $\{\boldsymbol{\mu}_k\}_{k=1}^c$.
To provide some intuition first consider the supervised case, where we have true one-hot vectors $\mathbf{y}_i$ for the class of each image. The MLE update for the Bernoulli parameters would be (intuitively percentage of times pixel j was on for a class k),

$$\mu_{k,j} = \frac{\sum_{i=1}^n y_{i,k} x_{i,j}}{\sum_{i=1}^n y_{i,k}}$$

For EM we instead utilize the expected values that we computed in the previous step.

$$\mu_{k,j} = \frac{\sum_{i=1}^n q_{i,k} x_{i,j}}{\sum_{i=1}^n q_{i,k}}$$

(You derived this for the multinomial case in the homework, the derivation for mixture of Bernoullis is quite similar. We won't ask you to do this again on the exam, but you should be able to use this property).

12. **PCA**

Consider a data set of four points $\mathbf{x}_1 = (1, 0)$, $\mathbf{x}_2 = (-1, 0)$, $\mathbf{x}_3 = (0, -2)$, $\mathbf{x}_4 = (0, 2)$.

(a) Compute the feature covariance matrix $\mathbf{S}$ for this dataset.

(b) Draw a (rough) sketch of the distribution $\mathcal{N}(0, \mathbf{S})$ formed with this covariance matrix.

(c) If we were to run PCA on this data, what would be the first and second principal components?

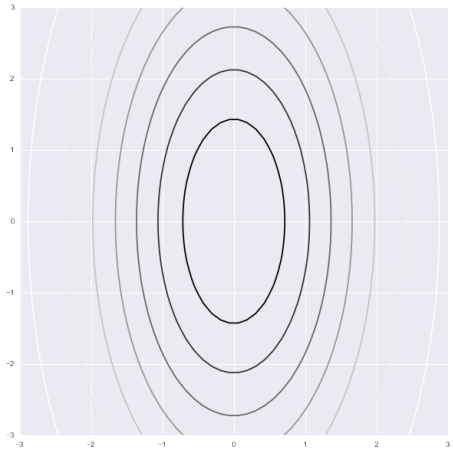
(d) Graph the four points after running PCA to project down to a single dimension. What is lost in this transformation?

Solutions:

(a) The normalized feature covariance matrix is $\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^\top$.

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$$

(b) Here's what it looks like. What we would like is a rough sketch of this form, i.e. that it is an axis-aligned ellipse.



(c) These will correspond to the eigenvectors with the first and second highest eigenvalues. In this case they have a simple form $\mathbf{u}_1 = [0, 1]$ with $\mathbf{S}\mathbf{u}_1 = 8\mathbf{u}_1$ and $\mathbf{u}_2 = [1, 0]$ with $\mathbf{S}\mathbf{u}_2 = 2\mathbf{u}_2$. This can also be seen visually from the data.

(d) Projecting to one dimension corresponds to projecting onto the first principal component,

$$\mathbf{z}_i = (\mathbf{x}_i^\top \mathbf{u}_1)\mathbf{u}_1$$

This takes a simple form for this problem, we just drop the x_1 component of each data point.

$$\mathbf{z}_1 = (0, 0), \mathbf{z}_2 = (0, 0), \mathbf{x}_3 = (0, -2), \mathbf{x}_4 = (0, 2)$$

This has the effect of collapsing $\mathbf{x}_1$ and $\mathbf{x}_2$ to the same point, but keeps the distinction between $\mathbf{x}_3$ and $\mathbf{x}_4$.