Question 1

Let S be the symmetric matrix

$$\begin{bmatrix} -1 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix}.$$

(a) Run power iteration on S with initialization $\mathbf{a} = (1, 0, 0)$. What is the state $\mathbf{a}_5$ after five steps? What is the spectral norm estimate $\alpha_5 = \|S\mathbf{a}_5\|/\|\mathbf{a}_5\|$?

Solution: The first five iterates are $(0, 0, 0) \to (-.707, 0, .707) \to (.816, -.408, -.408) \to (-.707, 0, .707) \to (.816, -.408, -.408) \to \mathbf{a}_5 = (-.707, 0, .707)$ up to three digits of precision. The spectral norm estimate is $\alpha_5 \approx 1.732$.

(b) Repeat part (a) with initialization $\mathbf{b} = (0, 1, 0)$. How do $\mathbf{b}_5$ and β_5 compare with $\mathbf{a}_5$ and α_5 above? Explain the similarities and differences.

Solution: Now the iterates are $(0, 1, 0) \to (0, .707, -.707) \to (-.408, .816, -.408) \to (0, .707, -.707) \to (-.408, .816, -.408) \to \mathbf{b}_5 = (0, .707, -.707)$ and $\beta_5 \approx 1.732$.

The matrix S has eigenvalues $\sqrt{3}$, 0, and $-\sqrt{3}$. Let's call the corresponding eigenvectors $\mathbf{v}_+$, $\mathbf{v}_0$, and $\mathbf{v}_-$

Each iterate amplifies the component of $\mathbf{x}$ in the directions of $\mathbf{v}_+$ and $\mathbf{v}_-$ in equal measure. The component in the direction of $\mathbf{v}_0$ eventually vanishes. The state is evantually dominated by some linear combination $\alpha_+\sqrt{3}^t\mathbf{v}_+ + \alpha_-(-\sqrt{3})^t\mathbf{v}_-$ normalized suitably. The values of α_+ and α_- depend on the initialization, which explains the discrepancy in the states. The state itself keeps shifting between (multiples of) $\alpha_+\mathbf{v}_+ + \alpha_-\mathbf{v}_-$ and $\alpha_+\mathbf{v}_+ - \alpha_-\mathbf{v}_-$, which explains the periodic pattern. The absolute value of the dominant eigenvalues is $\sqrt{3}$, which explains the proximity of both α_5 and β_5 to $\sqrt{3}$.

(c) Now repeat parts (a) and (b) on input $S' = S + I$, where I is the identity matrix. Explain how the answers in this part relate to the ones you obtained in parts (a) and (b).

Solution: The new iterates are $\mathbf{a} = (1, 0, 0) \to (0, 0, 1) \to (.577, -.577, .577) \to (.229, -.688, .688) \to (.254, -.763, .594) \to \mathbf{c}_5 = (.218, -.777, .591)$ and $\mathbf{b} = (0, 1, 0) \to (0, .894, -.447) \to (-.169, .845, -.507) \to (-.186, .808, -.559) \to (-.205, .796, -.569) \to \mathbf{c}'_5 = (-.208, .791, -.575)$. The iterations now appear to converge to the same vector (up to sign). At step 10 they match up to three digits of precision: $-\mathbf{c}_{10} \approx \mathbf{c}'_{10} \approx (-.211, .789, -.577)$.

Adding the identity matrix so S has the effect of shifting all eigenvalues up by one while preserving the corresponding eigenvector. Thus S' now has the single dominant eigenvalue $1 + \sqrt{3}$ with eigenvector $\mathbf{v}_+$. Iterates will eventually converge to $\mathbf{v}_+$ (unless the initialization is exactly orthogonal to it). This is why the eventual state no longer depends on the initialization, and why the iterates stabilize.

(d) Finally, pick any vector $\mathbf{d}$ orthogonal to the output $\mathbf{c}_5$ from part (c) (when initialized with $(1, 0, 0)$). What happens when you run power iteration on S' with initialization $\mathbf{d}$? Explain.

Solution: We start with $\mathbf{d} = (.777, .218, 0)$ and normalize it to length 1. This is orthogonal to $\mathbf{c}_5$ at least up to 3 digits of precision. The first ten iterates $\mathbf{d}_t$ and the corresponding values $r_t = \|d_t\|/\|d_{t-1}\|$ are

```
t  d_t                      r_t
------------------------------
0 [ 0.777  0.218  0    ]
1 [ 0.     0.615  0.789]   .709
```

```
 2 [ 0.857  0.48   0.189]  .920
 3 [ 0.193  0.791  0.58 ]  .975
 4 [ 0.501  0.865 -0.015] 1.158
 5 [-0.008  0.977 -0.212] 1.786
 6 [-0.085  0.872 -0.482] 2.485
 7 [-0.179  0.826 -0.534] 2.694
 8 [-0.196  0.802 -0.564] 2.727
 9 [-0.207  0.794 -0.572] 2.731
10 [-0.209  0.79  -0.576] 2.732
```

If $\mathbf{d}$ was perfectly orthogonal to $\mathbf{v}_+$, all iterates would belong to the subspace spanned by $\mathbf{v}_0$ and $\mathbf{v}_-$. Eventually $\mathbf{v}_0$ should dominate as it is associated to the larger eigenvalue 1. At step 3 the norm ratio is around one, indicating that $\mathbf{d}_3$ might be zeroing in on $\mathbf{v}_0$. This effect, is however, short-lived; $\mathbf{d}_10$ is already quite close to $\mathbf{c}_{10}$.

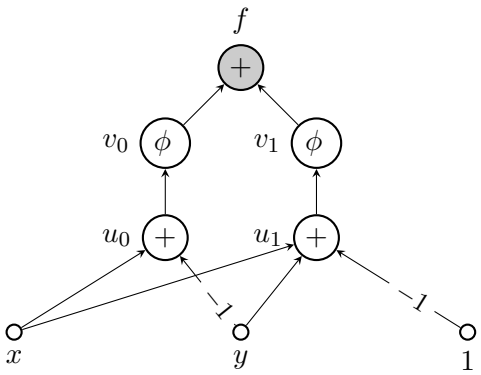
The reason is that, owing to convergence and precision errors, $\mathbf{d}$ is not perfectly orthogonal to $\mathbf{v}_+$. It has some small component in the direction of $\mathbf{v}_+$: $\mathbf{d}\cdot\mathbf{v}_+ \approx \mathbf{d}\cdot\mathbf{c}_{10} \approx .008$. Although initially small this component is amplified by the largest eigenvalue of S' and eventually dominates $\mathbf{d}_t$.

Question 2

Let $f(x, y) = \phi(x - y) + \phi(x + y - 1)$, where ϕ is some "activation function" with one input and one output.

(a) Draw a circuit for f with $+$, $\times$, and ϕ gates. You may use edge weights for scaling.

Solution:

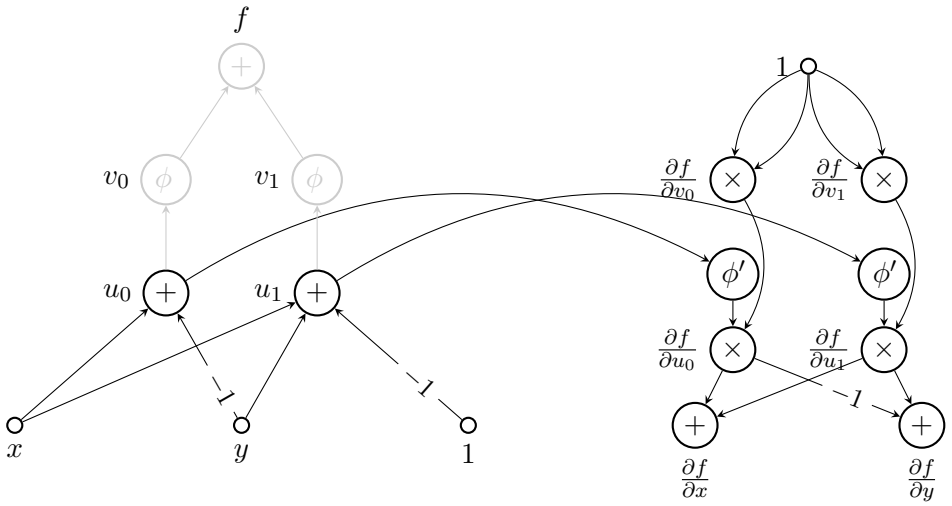


(b) Draw the circuit ∇f obtained by applying Backpropagation to f. The gates in your circuits should be $+$, $\times$, ϕ, and ϕ'. What is the size of this circuit? What is the depth?

Solution: In this circuit there are gates $\partial f/\partial z$ for every gate z in the circuit in part (a). The circuit is constructed by applying the chain rule backwards starting from $\partial f/\partial f = 1$:

$$\frac{\partial f}{\partial v_0} = \frac{\partial f}{\partial f}\cdot\partial_{v_0}[f] = \frac{\partial f}{\partial f}\times 1$$
$$\frac{\partial f}{\partial u_0} = \frac{\partial f}{\partial v_i}\cdot\partial_{u_0}[v_i] = \frac{\partial f}{\partial v_0}\times\phi'(u_0)$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u_0}\cdot\partial_x[u_0] + \frac{\partial f}{\partial u_1}\cdot\partial_x[u_1] = \frac{\partial f}{\partial u_0}\times 1 + \frac{\partial f}{\partial u_1}\times 1$$

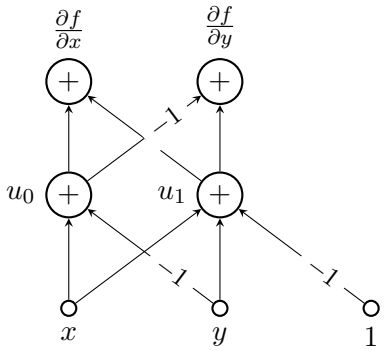
and similarly for v_1, u_1, y. The resulting circuit is



This circuit has size 10 (after eliminating gates v_0, v_1, f) and depth 4. If we did not allow scaling by constants, there would be an extra 3 multiplication gates that implement these scalings.

(c) Assume now that ϕ is the function $\phi(t) = \frac{1}{2}t^2$. Then $\phi'(t) = t$ is the identity function. Simplify the circuit by evaluating all gates that depend on constants only (e.g., replace $1 \times (-3)$ by the constant -3) and short-circuiting all ϕ' gates. What is the size and depth of the simplified circuit?

Solution: The ϕ' gates become through connections. The multiplication gates take two arguments one of which is the constant 1 so they also disappear. The resulting circuit has size 4 and depth 2.



(d) Explain how the circuit in part (c) computes the function $\nabla f(\mathbf{x}) = A^T(A\mathbf{x} + b)$, where $A\mathbf{x} = b$ is the linear system $x - y = 0, x + y = 1$.

Solution: The layer connecting $x, y, 1$ to u_0, u_1 is inherited from the circuit in part (a). It computes $A\mathbf{x}+b$. Without the constant 1 it computes the function $\mathbf{u} = A\mathbf{x}$. The next layer is its mirror image. This is the function $\nabla f = A^T\mathbf{u}$.

(e) In general, given a linear system with m equations and n unknowns, what is the size, depth, and number of wires of the output of backpropagation given a circuit for $f(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - b\|^2$ as input? Provide an answer in big-theta notation and justify it.

Solution: Applying backpropagation to the circuit representation of f, after simplifying, results in a circuit whose first layer has m plus gates u_1 up to u_m. Gate u_j is connected to input x_i by a wire scaled with a_{ij}. The second layer has n plus gates $\partial f/\partial x_i$. The wire from u_j to $\partial f/\partial x_i$ is also scaled by a_{ij}. The size of this circuit is $m + n$ and its depth is two ($O(1)$ in asymptotic notation).

The number of wires is at most $m(n + 1)$ in the lower and at most mn in the upper one as there can be a potential wire between every pair of nodes, for a total of $O(mn)$ wires. If $\mathbf{A}$ was a sparse matrix with s nonzero entries, the number of wires would be at most $2s + m$.

If we did not allow scaling of the edges, the size of the circuit would grow to $O(mn)$ because each edge would have to be augmented by a gate that implements multiplication by the resulting constant.

This is why applying Theorem 3 in Lecture 5 gives a circuit bound of $O(mn)$, i.e., the number of wires. The non-simplified circuit produced by backpropagation contains a multiplication by $\partial_{x_i}[u_j]$ for every addition gate u_j in f. This is a total of mn multiplications. In our instance, u_j are additions, $\partial_{x_i}[u_j]$ is the constant a_{ij}, and all these multiplications can be replaced by wire scalings, reducing the effective circuit size.

Question 3

A *decision tree* is a nested if-then-else program that branches over variables and outputs variables or their negations. For example, the decision tree in Figure 1 evaluates to $+1$ when the input is $x = +1, y = +1, z = +1$.

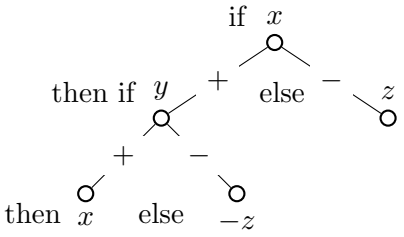


Figure 1: A function $f(x, y, z)$ specified by a decision tree.

(a) Write the list representation of the function f specified by the decision tree in Figure 1.

Solution: With $+$ and $-$ standing for $+1$ and -1, respectively, the list representation is

xyz	$+++$	$++-$	$+-+$	$+--$	$-++$	$-+-$	$--+$	$---$
$f(x,y,z)$	$+$	$+$	$-$	$+$	$+$	$-$	$+$	$-$

(b) Write the polynomial representation of f. (**Hint:** For every root-to-leaf path write a polynomial that outputs the leaf value if this path is taken and zero if not.)

Solution: Every root-to-leaf path can be described by a function that evaluates to its leaf value when this path is taken and zero otherwise. The point functions describing the three path (in a left-to-right ordering) are $\text{point}_{++}(x, y)$, $-z\text{point}_{+-}(x, y)$, and $z\text{point}_{-}(x)$. f is the sum of these three functions:

$$\begin{aligned} f(x,y,z) &= \text{point}_{++}(x,y) - z\text{point}_{+-}(x,y) + z\text{point}_{-}(x) \\ &= \frac{1+x}{2}\cdot\frac{1+y}{2} - z\cdot\frac{1+x}{2}\cdot\frac{1-y}{2} + z\cdot\frac{1-x}{2} \\ &= \tfrac{1}{4} + \tfrac{1}{4}x + \tfrac{1}{4}y + \tfrac{1}{4}z + \tfrac{1}{4}xy - \tfrac{3}{4}xz + \tfrac{1}{4}yz + \tfrac{1}{4}xyz. \end{aligned}$$

As a sanity check, the sum of the squares of the coefficients equals one as required by Parseval's identity.

(c) The *depth* of the decision tree is the maximum number of nested loops plus one, which equals the length of the longest root-to-leaf path in the tree representation plus one. For example the decision tree in Figure 1 has depth three. Prove that for any decision tree, the degree of its polynomial representation is at most its depth.

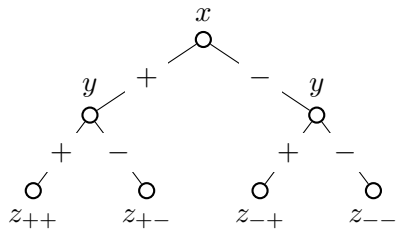
Solution: Each path p in the decision tree is labeled by a sequence of variable-value pairs $(x_1, v_1), \ldots, (x_d, v_d)$ and an output variable y. The function $f_p = y \cdot \text{point}_{v_1,\ldots,v_d}(x_1, \ldots, x_d)$ takes its leaf value when the path is taken and zero otherwise. This function f_p depends on $d+1$ inputs so its degree can be at most $d+1$, which is at most the depth. The decision tree itself is the sum of all such f_p so its degree is also at most the depth.

(d) The *2-bit addressing function* is the function $\mathrm{addr}_2\colon \{-1,+1\}^6 \to \{-1,+1\}$ given by

$$\mathrm{addr}_2(x, y, z_{++}, z_{+-}, z_{-+}, z_{--}) = z_{xy}.$$

Describe a decision tree for addr_2. What is the degree of its polynomial representation? Prove both an upper bound (it is at most ...) and a lower bound (it is at least ... because ...).

Solution: The decision tree reads x, then reads y, then outputs z_{xy}:



The degree of addr_2 is at most 3 as this decision tree has depth 2. To argue that its degree is at least 3 we can show that the monomial xyz_{++} appears in the polynomial representation of addr_2 with nonzero Fourier coefficient. This Fourier coefficient is the average of addr_2 times xyz_{++}.

This average can be calculated by summing up all $2^6 = 64$ possible assignments. An easier way (if you know probability) is to apply the law of total expectation: Conditioned on $x = y = +1$, $\mathrm{addr}_2 = z_{++}$ and $\mathrm{addr}_2 \cdot (xyz_{++}) = 1$. Otherwise addr_2 is conditionally independent of z_{++} and $\mathrm{addr}_2 \cdot (xyz_{++})$ averages out to zero.

$$\begin{aligned}
\mathrm{E}[\mathrm{addr}_2 \cdot (xyz_{++})] &= \tfrac{1}{4}\,\mathrm{E}[\mathrm{addr}_2 \cdot (xyz_{++}) | x = y = 1] + \tfrac{3}{4}\,\mathrm{E}[\mathrm{addr}_2 \cdot (xyz_{++}) | \textsc{not}\ (x = y = 1)] \\
&= \tfrac{1}{4}\,\mathrm{E}[z_{++} \cdot (1 \cdot 1 \cdot z_{++}) | x = y = 1] + \tfrac{3}{4}\,\mathrm{E}[\mathrm{addr}_2 | \textsc{not}\ (x = y = 1)]\,\mathrm{E}[xyz_{++} | \textsc{not}\ (x = y = 1)] \\
&= \tfrac{1}{4} \cdot 1 + \tfrac{3}{4} \cdot \mathrm{E}[\mathrm{addr}_2 | \textsc{not}\ (x = y = 1)] \cdot 0,
\end{aligned}$$

which evaluates to $1/4$, not zero.

Question 4

You have recorded the graph of mutual friendships among sixteen people (Alice, Bob, up to Patrick). They cluster into two groups. Most of the friendships are among people within the same group. Find the two groups. Your personalized instance is available here.

Your solution should consist of two lists, one for the members in each group (e.g., group 1: Alice, Charlie, Fred; group 2: Bob, Dave, Eve).

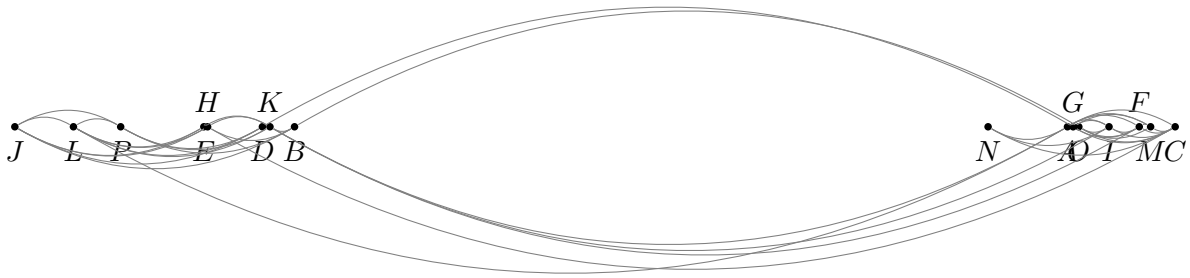
You may use any of the algorithms covered in class or write your own. Explain clearly how you arrived at your solution. Undocumented computer code will not be entertained as a satisfactory explanation.

Solution: My instance is

```
. ABCDEFGHIJKLMNOP
A  -+--++-+-+++++-
B - -----+-++---++
C +- --++++---+++-
D --- --+--+++---+
E ---- --+-+++----
F +-+-- +-+-+-+-+-
G +-++-+ -+---+---
H -++-+-- -+++----
I +-+--++- -+--++-
J -+-++--+- -+---+
```

```
K ++-+++-++- +---+
L +--++--+-++ ---+
M +-+--++----- -+-
N +-+-----+---- --
O +++--+--+---+- -
P -+-+-----+++---
```

with 7 friendships among the groups. I ran 50 rounds of subspace iteration on the adjacency matrix (in which + and − are replaced by 1 and 0, respectively), initialized with the vectors $\mathbf{e}_1 = (1, 0, 0, \ldots, 0)$ and $\mathbf{e}_2 = (0, 1, 0, \ldots, 0)$. The algorithm produces two vectors that approximate the top two eigenvectors of the adjacency matrix. Here is a plot of the second eigenvector:



This embedding visually splits into the groups `B D E H J K L P` and `A C F G I M N O` with precisely seven friendships between groups.