CS 344 (Spring 2018): Class Test 4

Instructor: Shivaram Kalyanakrishnan

11.05 a.m. – 12.00 p.m., April 18, 2018, 103 New CSE Building

Total marks: 15

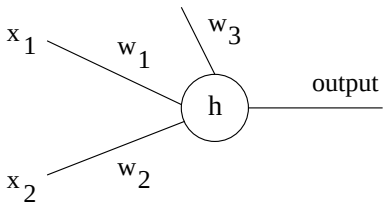
Note. Provide brief justifications and/or calculations along with each answer to illustrate how you arrived at the answer.

Question 1. In class we considered the sigmoid function $g(\alpha) = \frac{1}{1+e^{-\alpha}}$ for $\alpha \in \mathbb{R}$ as an activation function in neural networks. This question considers the use of the hyperbolic tangent (or *tanh*) function for the same purpose. For $\alpha \in \mathbb{R}$, this function is defined as

$$h(\alpha) = \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}}.$$

1a. Draw a graph of $h(\alpha)$ against α. In their role as activation functions, what are the similarities and differences between h and g? [2 marks]

1b. Consider the neural network shown in the figure below. For input (x_1, x_2), its output is $h(w_1x_1 + w_2x_2 + w_3)$, where $w_1, w_2, w_3 \in \mathbb{R}$.



Suppose for input (x_1, x_2), the specified target label is y, which makes the squared prediction error $(y - h(w_1x_1 + w_2x_2 + w_3))^2$. How would w_1 be updated if backpropagation is performed based on this error, with learning rate α? Your answer must be in terms of x_1, x_2, y, w_1, w_2, and w_3, combined using the basic arithmetic operations, exponentiation, and h. [3 marks]

Question 2. We are given a set of $n \geq 2$ points $D = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n\}$, where $\mathbf{x}^i \in \mathbb{R}^d$ for $i \in \{1, 2, \ldots, n\}$, and $d \geq 1$. Assume the n points are all distinct. Also, for every $k \in \{2, 3, \ldots, n\}$, assume that there is a unique optimal k-clustering of D, denoted

$$(\mathcal{C}_k^\star, \mu_k^\star) = \operatorname*{argmin}_{(\mathcal{C},\mu)} SSE(\mathcal{C}, \mu, D).$$

The minimum is over clusterings $\mathcal{C} : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ and sequences of cluster centres $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$, where for $k' \in \{1, 2, \ldots, k\}$, $\mu_{k'} \in \mathbb{R}^d$;

$$SSE(\mathcal{C}, \mu, D) = \sum_{i=1}^{n} ||\mathbf{x}^i - \mu_{\mathcal{C}(i)}||^2.$$

In your answers to the questions below, you can use any results that were derived in class.

2a. For $k \in \{2, 3, \ldots, n-1\}$, show that

$$SSE(\mathcal{C}_{k+1}^\star, \mu_{k+1}^\star, D) < SSE(\mathcal{C}_k^\star, \mu_k^\star, D).$$

In other words, show that the sum squared error of the optimal $(k+1)$-clustering will be smaller than the sum squared error of the optimal k-clustering. [3 marks]

2b. Taking $k = 2$, note that $(\mathcal{C}_2^\star, \mu_2^\star)$ is the unique optimal 2-clustering of D. Let $A \subset D$ be the set of points assigned cluster 1 by $(\mathcal{C}_2^\star, \mu_2^\star)$, and let $B = D \setminus A$ be the set of points assigned cluster 2 by $\mathcal{C}_2^\star$. Without loss of generality, assume $|A| \leq |B|$: that is, A is the smaller cluster (or the clusters are equal-sized). For a given number of points n and dimension d, What are the minimum and maximum values of the ratio $\frac{|A|}{|B|}$? Justify your answer. [2 marks]

Question 3. This question tests your understanding of Q-learning.

3a. In which context—in other words, to solve which problem—would an agent apply Q-learning? [1 mark]

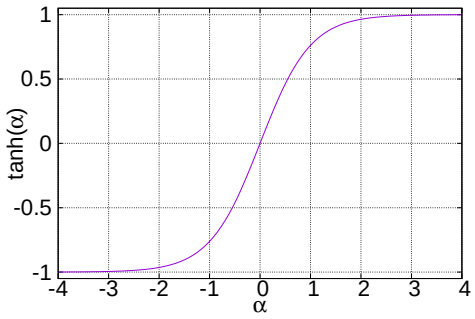
3b. What are the main differences between Q-learning and value iteration? [1 mark]

3c. Describe the Q-learning algorithm, specifying its main step (the "update rule") in 2–3 lines of pseudocode. [2 marks]

3d. Is the version of Q-learning used in the AlphaGo program guaranteed to converge? Why, or why not? [1 mark]

Solutions

1a. The plot of $h(\alpha) = \tanh(\alpha)$ is shown below.



We observe that for $\alpha \in \mathbb{R}$, $h(\alpha) = 1 - 2g(-2\alpha)$. Thus, h can be obtained by applying linear transforms both to the input and the output of g, and vice versa. The functions are both monotonically increasing, each with its maximum at ∞, minimum and $-\infty$, and "mid-point" at 0. The main difference is in their ranges. g takes values in $(0, 1)$, and $g(0) = 1/2$. h takes values in $(-1, 1)$, and $h(0) = 0$.

1b. First we compute $h'(\alpha)$:

$$h'(\alpha) = \frac{d}{d\alpha}\left(\frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}}\right) = -\frac{(e^{\alpha} - e^{-\alpha})^2}{(e^{\alpha} + e^{-\alpha})^2} + 1 = 1 - (h(\alpha))^2.$$

Taking $Error = (y - h(w_1x_1 + w_2x_2 + w_3))^2$, we get

$$\frac{\partial Error}{\partial w_1} = 2(y - h(w_1x_1 + w_2x_2 + w_3))h'(w_1x_1 + w_2x_2 + w_3)(-x_1).$$

The weight w_1 is set by the backpropagation operation to be $w_1 - \alpha\frac{\partial Error}{\partial w_1}$, which, upon substituting, is obtained to be

$$w_1 + 2\alpha x_1(y - h(w_1x_1 + w_2x_2 + w_3))(1 - (h(w_1x_1 + w_2x_2 + w_3))^2).$$

2a. In order to show that $SSE(\mathcal{C}^{\star}_{k+1}, \mu^{\star}_{k+1}, D) < SSE(\mathcal{C}^{\star}_{k}, \mu^{\star}_{k}, D)$, it suffices to show that there is *some* $(k+1)$-clustering $(\bar{\mathcal{C}}_{k+1}, \bar{\mu}_{k+1})$ such that $SSE(\bar{\mathcal{C}}_{k+1}, \bar{\mu}_{k+1}, D) < SSE(\mathcal{C}^{\star}_{k}, \mu^{\star}_{k}, D)$. The inequality would follow since $(\mathcal{C}^{\star}_{k+1}, \mu^{\star}_{k+1}, D)$ is the optimal $(k+1)$-clustering.

We construct $(\bar{\mathcal{C}}_{k+1}, \bar{\mu}_{k+1})$ as follows. Take an arbitrary cluster i in $\mathcal{C}^{\star}_{k}$ that has at least two points (there must be some such cluster since $k < n$). At least one point p in i must be different from the cluster centre (since the points are all distinct), which is also the cluster centroid.

We set $\bar{\mathcal{C}}_{k+1}$ to be identical to $\mathcal{C}^{\star}_{k}$ for all points other than p. The point p is assigned to a new cluster $k+1$, and is the only point in this cluster. $\bar{\mu}_{k+1}$ is identical to $\mu^{\star}_{k}$ on clusters $1, 2, \ldots, i-1, i+1, \ldots, k$. The cluster centre $\bar{\mu}_{k+1}(i)$ is set to be the centroid of all the points $\bar{\mathcal{C}}_{k+1}$ assigns to cluster i. The cluster centre cluster centre $\bar{\mu}_{k+1}(k+1)$ is set to p.

By this construction, $\bar{\mathcal{C}}_{k+1}$ has the same contribution to its sum squared error as $\mathcal{C}^{\star}_{k}$ does from clusters other than i and $k+1$. Using the property that the centroid of a set of points minimises

the sum squared error, it follows that cluster i in $\bar{\mathcal{C}}_{k+1}$ will have a lower sum squared error than cluster i in $\mathcal{C}_k^\star$ (since p has been removed and the centre readjusted). Cluster $k+1$ in $\bar{\mathcal{C}}_{k+1}$ has a zero sum squared error. Thus, $SSE(\bar{\mathcal{C}}_{k+1}, \bar{\mu}_{k+1}, D) < SSE(\mathcal{C}_k^\star, \mu_k^\star, D)$, which completes our proof.

2b. From the working in 2a, we notice that any optimal k-clustering must have at least one point in each cluster: thus, $|A| \geq 1$. Are there problem instances in which $|A| = 1$? Can $|A|$ be as high as $\lfloor \frac{n}{2} \rfloor$? We answer both questions in the affirmative.

Consider two spheres S_1 and S_2 in $\mathbb{R}^d$, each of diameter 1, with centres at least $2n+2$ apart (which implies any point $P_1 \in S_1$ is at least distance $2n$ away from any point $P_2 \in S_2$). Now consider only data sets whose points all lie in S_1 or S_2, with each sphere containing at least one point. For a given data set D, let there be m points inside S_1 and $n-m$ points inside S_2, with $1 \leq m \leq \lfloor n/2 \rfloor$. Now imagine forming two clusters from D. Any cluster that includes one point each from S_1 and S_2 will have an SSE of at least n^2, since at least one point will have a cluster centre n or more distance away. However, the clustering that puts all the points in S_1 into one cluster and those in S_2 into another (with optimised cluster centres) will have an aggregate SSE of at most n, since each point will have its cluster centre within a distance of 1. Since this property holds for all $1 \leq m \leq \lfloor n/2 \rfloor$, we have essentially shown that there exist data sets for which $|A| = 1$, $|A| = 2$, $\ldots$, $|A| = \lfloor n/2 \rfloor$. Thus $1/(n-1) \leq |A|/|B| \leq \lfloor n/2 \rfloor / \lceil n/2 \rceil$ (the latter quantity is 1 if n is even, and slightly less otherwise.)

3a. Q-learning is used by an agent to solve the *reinforcement learning* problem. In other words, the agent is interacting sequentially with an MDP whose transition and reward functions are unknown. For $t \geq 0$: the agent, from state s^t and takes action a^t, to which the environment responds with a reward r^t and next state s^{t+1}. The next state and reward are drawn based on the environment's transition and reward functions, respectively. Q-learning can enable the agent to eventually take actions that are optimal, in thee sense that they maximise the expected long-term discounted reward.

3b. Q-learning is used by an agent to solve a *learning* problem (with the transition and reward functions of the MDP not given), while value iteration solves the *planning* problem (computing an optimal policy with the MDP given). Another distinction is that Q-learning maintains and updates an estimate of the optimal action value function $Q^\star$, whereas value iteration *typically* updates an estimate of the optimal value function $V^\star$. This second distinction is more a matter of convention than something intrinsic to the structure of value iteration (which, in principle, can also work with an estimate of $Q^\star$).

3c. For Q-learning to converge to $Q^\star$, actions must be picked such that every state-action pair is visited infinitely often in the limit. A common way to achieve this criterion is to pick actions ϵ-greedily with respect to the running estimate Q, for some $\epsilon \in (0, 1]$. If action a^t is taken from state s^t, and results in reward r^t and next state s^{t+1}, then Q is updated as follows.

$$Q(s^t, a^t) \leftarrow Q(s^t, a^t) + \alpha_t (r^t + \gamma \max_{a \in A} Q(s^{t+1}, a) - Q(s^t, a^t)).$$

Here γ is the discount factor of the MDP and $\alpha_t \in (0, 1]$ the learning rate.

3d. Q-learning is guaranteed convergence only when applied in "tabular form" on a small state-action space. In AlphaGo, the Q function is approximated using a neural network, which generalises across states, and thereby loses the assurance of convergence.