CS 188 Fall 2021 — Introduction to Artificial Intelligence

Exam Prep 11 Solutions

Q1. Machine Learning: Potpourri

(a) What it the **minimum** number of parameters needed to fully model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label Y and n features F_i? Assume binary class where each feature can possibly take on k distinct values. $2k^n - 1$

(b) Under the **Naive Bayes assumption**, what is the **minimum** number of parameters needed to model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label Y and n features F_i? Assume binary class where each feature can take on k distinct values. $2n(k-1)+1$

(c) You suspect that you are overfitting with your Naive Bayes with Laplace Smoothing. How would you adjust the strength k in Laplace Smoothing?

[✓ Increase k]　Decrease k

(d) While using Naive Bayes with Laplace Smoothing, increasing the strength k in Laplace Smoothing can:

[✓ Increase training error]　Decrease training error
[✓ Increase validation error]　[✓ Decrease validation error]

(e) It is possible for the perceptron algorithm to never terminate on a dataset that is linearly separable in its feature space.

True　[✓ False]

(f) If the perceptron algorithm terminates, then it is guaranteed to find a max-margin separating decision boundary.

True　[✓ False]

(g) In binary perceptron where the initial weight vector is $\vec{0}$, the final weight vector can be written as a linear combination of the training data feature vectors.

[✓ True]　False

(h) For binary class classification, logistic regression produces a linear decision boundary.

[✓ True]　False

(i) In the binary classification case, logistic regression is exactly equivalent to a single-layer neural network with a sigmoid activation and the cross-entropy loss function.

[✓ True]　False

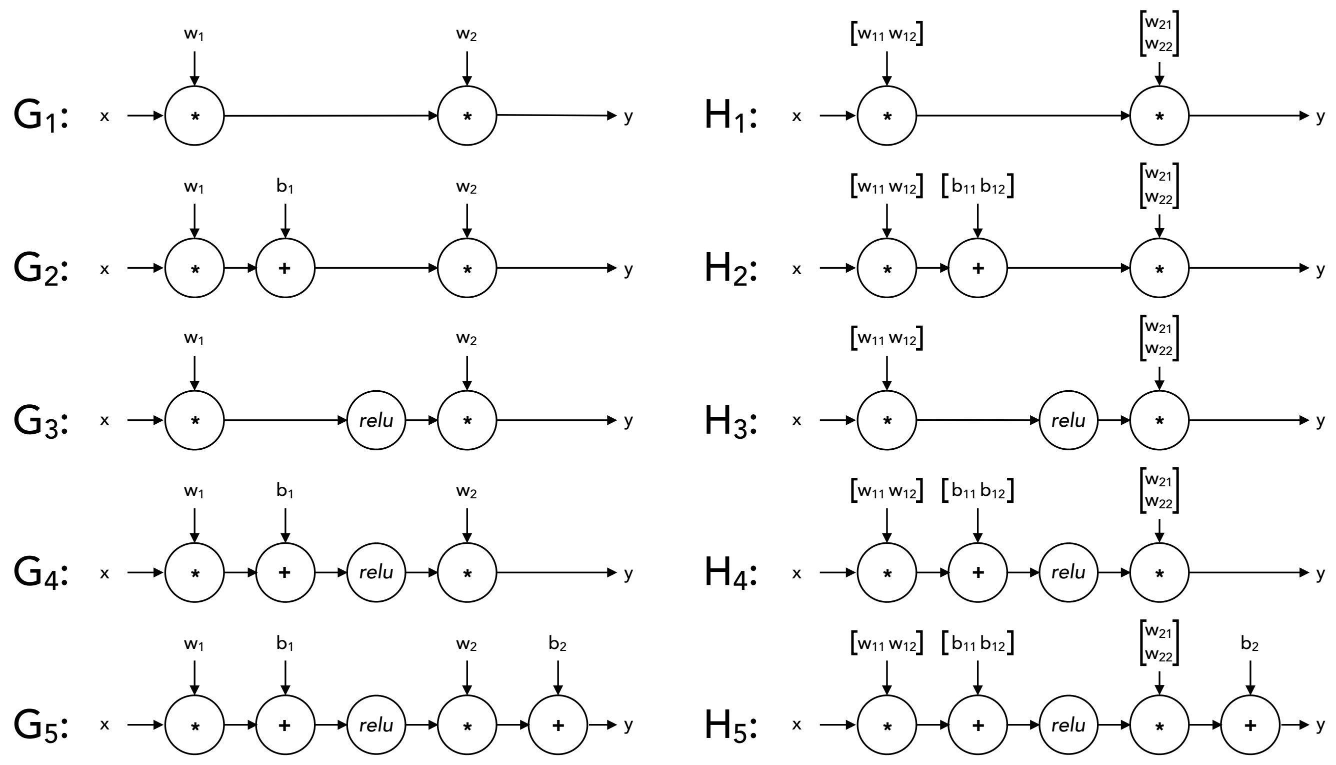
(j) You train a linear classifier on 1,000 training points and discover that the training accuracy is only 50%. Which of the following, if done in isolation, has a good chance of improving your training accuracy?

[✓ Add novel features]　Train on more data

(k) You now try training a neural network but you find that the training accuracy is still very low. Which of the following, if done in isolation, has a good chance of improving your training accuracy?

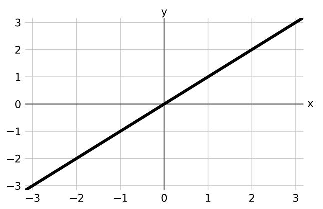
[✓ Add more hidden layers]　[✓ Add more units to the hidden layers]

Q2. Neural Networks: Representation

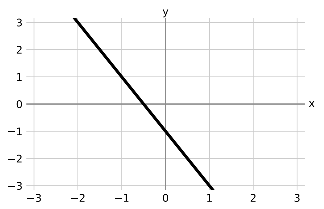

G₁: x → * (w₁) → * (w₂) → y
G₂: x → * (w₁) → + (b₁) → * (w₂) → y
G₃: x → * (w₁) → relu → * (w₂) → y
G₄: x → * (w₁) → + (b₁) → relu → * (w₂) → y
G₅: x → * (w₁) → + (b₁) → relu → * (w₂) → + (b₂) → y
H₁: x → * ([w₁₁ w₁₂]) → * ([w₂₁ w₂₂]ᵀ) → y
H₂: x → * ([w₁₁ w₁₂]) → + ([b₁₁ b₁₂]) → * ([w₂₁ w₂₂]ᵀ) → y
H₃: x → * ([w₁₁ w₁₂]) → relu → * ([w₂₁ w₂₂]ᵀ) → y
H₄: x → * ([w₁₁ w₁₂]) → + ([b₁₁ b₁₂]) → relu → * ([w₂₁ w₂₂]ᵀ) → y
H₅: x → * ([w₁₁ w₁₂]) → + ([b₁₁ b₁₂]) → relu → * ([w₂₁ w₂₂]ᵀ) → + (b₂) → y

For each of the piecewise-linear functions below, mark all networks from the list above that can represent the function **exactly** on the range $x \in (-\infty, \infty)$. In the networks above, *relu* denotes the element-wise ReLU nonlinearity: $relu(z) = max(0, z)$. The networks G_i use 1-dimensional layers, while the networks H_i have some 2-dimensional intermediate layers.

(a)



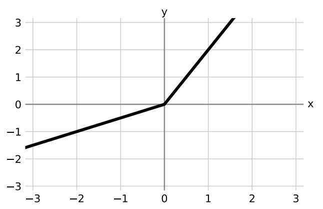
[✓ G_1] [✓ H_1]
[✓ G_2] [✓ H_2]
G_3 [✓ H_3]
G_4 [✓ H_4]
G_5 [✓ H_5]
◯ None of the above



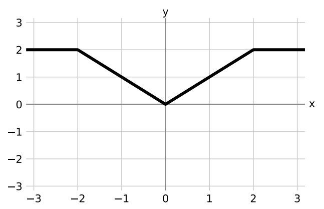
G_1 H_1
[✓ G_2] [✓ H_2]
G_3 H_3
G_4 [✓ H_4]
G_5 [✓ H_5]
◯ None of the above

The networks G_3, G_4, G_5 include a ReLU nonlinearity on a scalar quantity, so it is impossible for their output to represent a non-horizontal straight line. On the other hand, H_3, H_4, H_5 have a 2-dimensional hidden layer, which allows two ReLU elements facing in opposite directions to be added together to form a straight line. The second subpart requires a bias term because the line does not pass through the origin.

(b)



G_1 H_1
G_2 H_2
G_3 [✓ H_3]
G_4 [✓ H_4]
G_5 [✓ H_5]
◯ None of the above



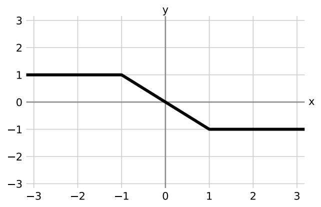
G_1 H_1
G_2 H_2
G_3 H_3
G_4 H_4
G_5 H_5
[✓ None of the above]

These functions include multiple non-horizontal linear regions, so they cannot be represented by any of the networks G_i which apply ReLU no more than once to a scalar quantity.

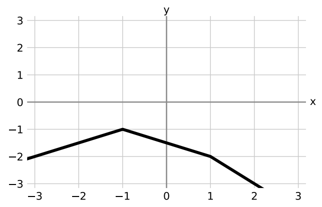
The first subpart can be represented by any of the networks with 2-dimensional ReLU nodes. The point of nonlinearity occurs at the origin, so nonzero bias terms are not required.

The second subpart has 3 points where the slope changes, but the networks H_i only have a single 2-dimensional ReLU node. Each application of ReLU to one element can only introduce a change of slope for a single value of x.

(c)



☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 ☐ H_4
☐ G_5 [✓ H_5]
○ None of the above



☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 [✓ H_4]
☐ G_5 [✓ H_5]
○ None of the above

Both functions have two points where the slope changes, so none of the networks G_i; H_1, H_2 can represent them.

An output bias term is required for the first subpart because one of the flat regions must be generated by the flat part of a ReLU function, but neither one of them is at $y = 0$.

The second subpart doesn't require a bias term at the output: it can be represented as $-relu(\frac{-x+1}{2}) - relu(x+1)$. Note how if the segment at $x > 2$ were to be extended to cross the x axis, it would cross exactly at $x = -1$, the location of the other slope change. A similar statement is true for the segment at $x < -1$.