Machine Learning & Data Mining
Final Exam

Caltech CS/CNS/EE 155
March 13, 2024

---

1 Multiple Choice Questions (35 points)

Just specify the correct answer. No need to justify.

Lasso (2 points)

Question A: (True or False)

Answer A: *True*

Bagging & Boostrap Sampling (3 points)

Question B: (Multiple Choice)

Answer B: *D*

Convolutional Filters (4 points)

Question C: (Multiple Choice)

Answer C: K_2, K_3, K_1

Multiclass SVMs (3 points)

Question D: (Multiple Choice)

Answer D: *4*

Feature Maps (3 points)

Question E: (Multiple Choice)

Answer E: *B*

Hard-Margin SVMs (5 points)

Question F: (True or False)

Answer F: *False*

Question G: (Multiple Choice)

Answer G: *A*

AdaBoost (2 points)

Question H: (True or False)

Answer H: *False*

Tensor Model Training (2 points)

Question I: (True or False)

Answer I: *True*

Bias-Variance Decomposition (2 points)

Question J: (True or False)

Answer J: *False*

HMM EM Learning (3 points)

Question K: (Multiple Choice)

Answer K: *B*

Non-Negative Matrix Factorization (2 points)

Question L: (True or False)

Answer L: *True*

Decision Trees (2 points)

Question M: (True or False)

Answer M: *True*

Overfitting (2 points)

Question N: (True or False)

Answer N: *True*

2 Naive Bayes (20 points)

We consider the following Naive Bayes model:

$$P(Happy?, Grade, Year) = P(Happy?)P(Grade|Happy?)P(Year|Happy?).$$

In other words, the y is the Happy? variable, and the two x's are the Grade and Year variables. We assume that all variables take two values, $Happy? \in \{Yes, No\}$, $Grade \in \{A, C\}$, and $Year \in \{Freshman, Senior\}$.

Consider the following training data:

Grade	Year	Happy?
A	Senior	Yes
A	Senior	Yes
A	Senior	No
A	Freshman	Yes
C	Freshman	No
C	Freshman	No
C	Senior	No
C	Senior	Yes

Question 1: (8 points) Fit the model parameters of the Naive Bayes using maximum likelihood with uniform unit pseudocounts. For instance, the maximum likelihood estimate for $P(Grade|Happy?)$ is:

$$P(Grade = A|Happy? = Yes) = \frac{1 + \sum_{(x,y)} 1_{[x_{Grade}=A \wedge y=Yes]}}{2 + \sum_{(x,y)} 1_{[y=Yes]}}.$$

Write out the final probability tables.

Answer :

$P(\text{grade} = A \mid \text{Happy?} = \text{Yes}) = \frac{1+3}{2+4} = \frac{2}{3}$

$P(\text{grade} = A \mid \text{Happy?} = \text{No}) = \frac{1+1}{2+4} = \frac{1}{3}$

$P(\text{grade} = C \mid \text{Happy?} = \text{Yes}) = \frac{1+1}{2+4} = \frac{1}{3}$

$P(\text{grade} = C \mid \text{Happy?} = \text{No}) = \frac{1+3}{2+4} = \frac{2}{3}$

$P(\text{Year} = \text{Freshman} \mid \text{Happy?} = \text{Yes}) = \frac{1+1}{2+4} = \frac{1}{3}$

$P(\text{Year} = \text{Freshman} \mid \text{Happy?} = \text{No}) = \frac{1+2}{2+4} = \frac{1}{2}$

$P(\text{Year} = \text{Senior} \mid \text{Happy?} = \text{Yes}) = \frac{1+3}{2+4} = \frac{2}{3}$

$P(\text{Year} = \text{Senior} \mid \text{Happy?} = \text{No}) = \frac{1+2}{2+4} = \frac{1}{2}$

$P(x|y)$

	grade = A	Year = Freshman
Happy? = Yes	2/3	1/3
Happy? = No	1/3	1/2

$P(y)$

	P(y)
Happy? = Yes	1/2
Happy? = No	1/2

Question 2: (5 points) Compute $P(Year = Freshman, Grade = C, Happy? = No)$ using the trained model from Question 1.

Answer :

$$P(\text{Year} = \text{Freshman}, \text{grade} = C, \text{Happy?} = \text{No})$$

$$= P(\text{Happy?} = \text{No})\, P(\text{grade} = C \mid \text{Happy?} = \text{No})\, P(\text{Year} = \text{Fresh} \mid \text{Happy?} = \text{No})$$

$$= \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{2} = \boxed{\frac{1}{6}}$$

Question 3: (7 points) Write out the pseudocode for drawing a sample from any trained model. Assume you have repeated access to a function `random()` that returns a uniform random number in $[0, 1]$.

Answer :

```
rand_y = random()
if rand_y < P(happy = Yes)
    happy = Yes
else
    happy = No
rand_grade = random()
if rand_grade < P(grade = A | happy = Yes)
    grade = A
else
    grade = C
rand_year = random()
if rand_year < P(year = Freshman | happy = Yes)
    year = Freshman
else
    year = Senior

return (grade, year, happy)
```

3 Data Transformations (15 points)

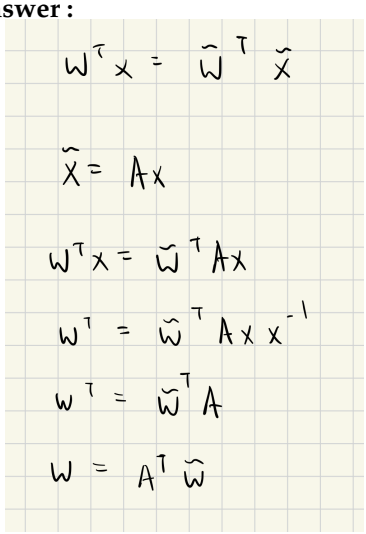
We consider the problem of linear regression, where we predict real values given input features x via $w^T x$ using a linear model w (ignoring the bias term). Suppose we want to tranform the data points x to a new representation via a transformation matrix: $\tilde{x} = Ax$. For instance, A can be a rescaling of the dimensions of x:

$$A = \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_D \end{bmatrix}, \tag{1}$$

where each $a_d > 0$ scales each dimension.

Question 1: (5 points) What is the relationship between w and $\tilde{w}$? In other words, write w as a function of $\tilde{w}$ and A such that $w^T x = \tilde{w}^T \tilde{x}$. Assume that A is a square, invertable matrix.

Answer :



Consider the ridge regression learning objective on the transformed data:

$$\arg\min_{\tilde{w}} \frac{\lambda}{2}\|\tilde{w}\|^2 + \sum_i (y_i - \tilde{w}^T \tilde{x}_i)^2, \tag{2}$$

Question 2: (5 points) Rewrite (2) using w, x, and A. In other words, what is the optimization problem that yields the w that corresponds to the $\tilde{w}$ learned in (2) (with the correspondence established in Question 1)? Assume that A is a square, invertible matrix.

Answer :

$$\tilde{w} = (A^T)^{-1} w$$

$$\underset{w}{\operatorname{argmin}} \quad \frac{\lambda}{2} \| (A^T)^{-1} w \|^2 + \sum_i \left(y_i - \tilde{w}^T A^{-1} A x_i \right)^2$$

$$\underset{w}{\operatorname{argmin}} \quad \frac{\lambda}{2} \| (A^T)^{-1} w \|^2 + \sum_i \left(y_i - w^T x_i \right)^2$$

Question 3: (5 points) Interpret your answers to Question 1 and Question 2 when A is a rescaling transformation such as (1). In other words, how is your answer to Question 2 different from standard ridge regression for w:

$$\arg\min_w \frac{\lambda}{2}\|w\|^2 + \sum_i (y_i - w^T x_i)^2.$$

Answer : *We see that as opposed to the standard ridge regression for w, we have an extra $(A^T)^{-1}$ factor applied to w in the ridge regression weight penalization term. Our goal was to transform the data points x to a new representation $\tilde{x} = Ax$. However, instead of using the transformed representations of our data points x in our minimization objective, we can directly reflect the changes in $\tilde{w}$ by applying the $(A^T)^{-1}$ factor. We know that $(A^T)^{-1}w$ gets us right back to $\tilde{w}$. Therefore, we are still minimizing the transformed weight vector, which also minimizes the original weight vector, since $w^T x = \tilde{w}^T \tilde{x}$.*

4 Latent Markov Embedding (15 points)

Question 1: (8 points) Show that the data likelihood $P(S)$ for the dual-point model (7) is never less than the data likelihood for the single-point model (9).

Answer : *The flexibility provided by having two separate embedding spaces in the dual-point model (as opposed to a single embedding space in the single-point model) allows for at least as good of a fit to the data, if not better (meaning that that data likelihood $P(S)$ for the dual-point model is never less than the data likelihood for the single-point model). We will prove this using a reduction argument. All instances of the dual-point model can be reduced to the single-point model with the following construction: $U = V = X$. Here we can see that when $U(s) = V(s) = X(s) \forall s$, we can express any dual-point model as a single-point model. In this case, the likelihoods are equivalent. Now we see that for any single-point model, the dual-point model is at least as expressive. In fact, for most cases, the likelihood of the dual-point model will be larger than the likelihood of the single-point model because the dual-point model has more parameters and hence more flexibility to fit the data.*

Question 2: (7 points) If Eq. (6) is equal to Eq. (8) for every pair of songs s and s', what does that imply about the relationship between U, V, and X? (Hint: the answers to the two questions are related.)

Answer : *This means that the single-point model and dual-point model are equivalent. As expressed in Question 1, this requires*

$$U = V = X$$

5 Neural Net Backprop Gradient Derivation (15 points)

Question 1: (5 points) For a given training data point (x, y), compute the stochastic gradient of the squared-loss of (x, y) w.r.t. w_{11}:

$$\frac{\partial}{\partial w_{11}} L(y, f(x)) \equiv \frac{\partial}{\partial w_{11}} (y - f(x))^2 .$$

Hint: write the formula using the chain rule and use the following definition of the derivative of $\sigma(s)$:

$$\frac{\partial}{\partial s}\sigma(s) = \sigma(s)(1 - \sigma(s)).$$

Answer :

$$\frac{d}{dw_{11}} L(y, f(x)) = \frac{d}{dw_{11}} (y - f(x))^2$$

$$= \underbrace{\frac{dL}{df(x)}}_{\text{I}} \cdot \underbrace{\frac{df(x)}{d\left(\sum_{i=1}^{2} u_i h_i(x)\right)}}_{\text{II}} \cdot \underbrace{\frac{d\left(\sum_{i=1}^{2} u_i h_i(x)\right)}{dh_1(x)}}_{\text{III}} \cdot \underbrace{\frac{dh_1(x)}{d\left(\sum_{j=1}^{2} w_{j1} x_j\right)}}_{\text{IV}} \cdot \underbrace{\frac{d\left(\sum_{j=1}^{2} w_{j1} x_j\right)}{dw_{11}}}_{\text{V}}$$

$$= \underbrace{2\left(f(x) - y\right)}_{\text{I}} \cdot \underbrace{\sigma\left(\sum_{i=1}^{2} u_i h_i(x)\right)\left(1 - \sigma\left(\sum_{i=1}^{2} u_i h_i(x)\right)\right)}_{\text{II}} \cdot \underbrace{u_1}_{\text{III}} \cdot \underbrace{\sigma\left(\sum_{j=1}^{2} w_{j1} x_j\right)\left(1 - \sigma\left(\sum_{j=1}^{2} w_{j1} x_j\right)\right)}_{\text{IV}} \cdot \underbrace{x_1}_{\text{V}}$$

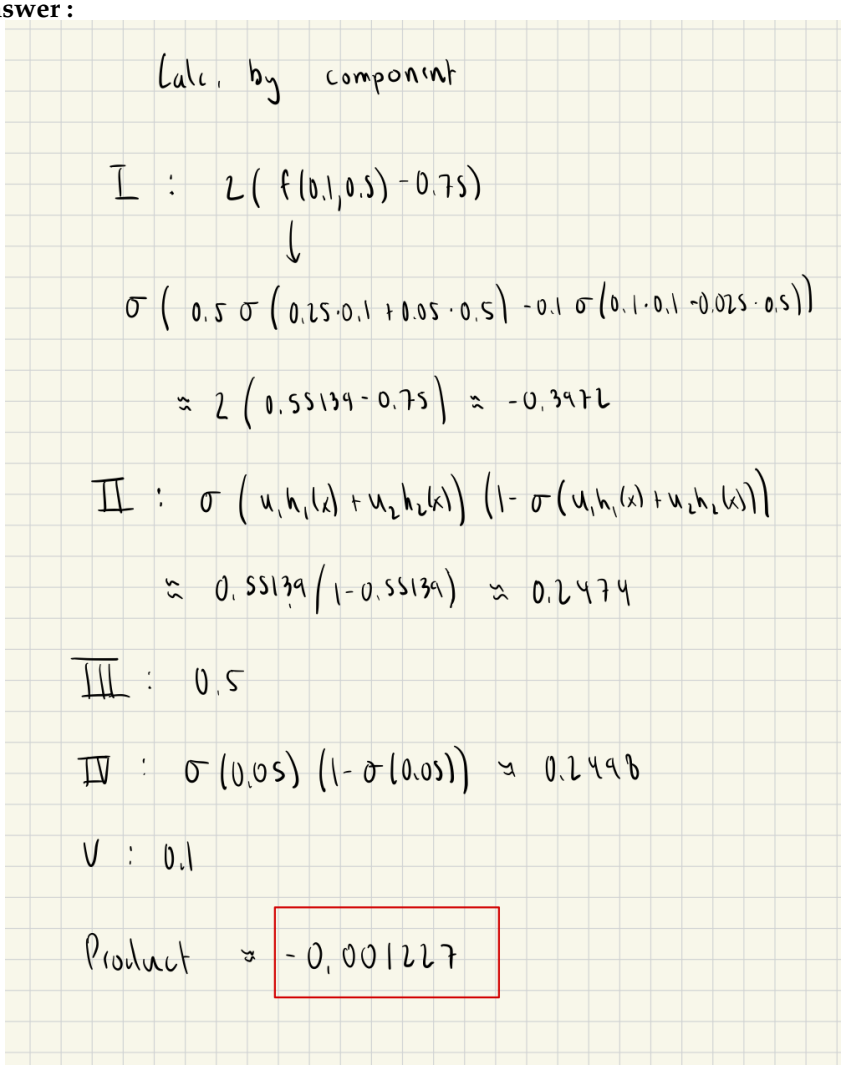
Question 2: (5 points) Find $\frac{\partial}{\partial w_{11}} L(y, f(x))$ where:

$$(x, y) = ((0.1, 0.5), 0.75)$$

$$(u_1, u_2) = (0.5, -0.1)$$

$$(w_{11}, w_{12}, w_{21}, w_{22}) = (0.25, 0.1, 0.05, -0.25)$$

Answer :



Question 3: (5 points) Use your derivations from the previous two questions to identify which term in the gradient derivation results in the vanishing gradient problem, and briefly explain how the problem is exacerbated in neural networks with more layers.

Answer :

We derived

$$\frac{d}{dw_{11}} L(y, f(x)) = \frac{d}{dw_{11}} (y - f(x))^2 = \frac{d}{df} (y - f(x))^2 \cdot \frac{df}{dw_{11}}$$

We see that the term

$\frac{df}{dw_{11}}$ results in vanishing gradient problem.

This is because $\frac{df}{dw_{11}}$ requires several more layers of chain rule (4 more) and this brings about several repeated terms from previous gradient layers. It multiplies together a product of small numbers from weights w_i and w_j yielding vanishing gradients. The problem is exacerbated in neural nets w/ more layers b/c even adding 1 layer can have an exponential effect.