- In the naïve Bayes formula below, replace the letters t, u, v, w, x, y, and z with the appropriate terms from this list: number of data points, candidate class label, set of class labels, argmin, min, argmax, max, feature value, number of features. Note that not all of the terms might be used and some terms may be used more than once.

$$t_{u \in v} P(w) \sum_{i=1..x} P(y|z)$$

t = argmax

u = candidate class label

v = set of class labels

w = candidate class label

x = number of data points

y = feature value

z = candidate class label

- Match the type of regularization with the corresponding description of the impact of applying the regularization term to a loss function.

L1 -> prefers sparse vectors

L2 -> spreads error among all terms

L infinity -> focuses on outliers

- Can we can get multiple local optimum solutions if we solve a linear regression problem by minimizing the sum of squared errors using gradient descent? Why or why not?

No, because this is a convex loss function.

- Suppose we perform a 10-fold cross validation experiment using two algorithms, A and B, to learn a concept from 300 training examples. You may assume that the positive and negative class are completely balanced for each fold and the differences between the errors for each fold is computed for you. The following table on the left summarizes the results.

  Does algorithm B perform significantly better than algorithm A in this case? Why or why not?

fold	$error_A$	$error_B$	δ
1	0.30	0.20	0.1
2	0.36	0.25	0.11
3	0.31	0.22	0.09
4	0.29	0.19	0.1
5	0.30	0.20	0.1
6	0.30	0.21	0.09
7	0.33	0.22	0.11
8	0.34	0.23	0.11
9	0.28	0.18	0.1
10	0.29	0.20	0.09
Mean	0.31	0.21	0.1

dof	$t_{95\%,dof}$
1	12.71
2	4.30
3	3.18
4	2.78
5	2.57
6	2.45
7	2.36
8	2.31
9	2.26
10	2.23
11	2.20

$$t = (\mu_a - \mu_b)\sqrt{\frac{N(N-1)}{\sum_n (\hat{a}_n - \hat{b}_n)^2}}$$

t = (0.31-0.21)*sqrt((10*9) / ($.1^2$+ $.11^2$ + $.09^2$ + $.1^2$ + $.1^2$ + $.09^2$ + $.11^2$ + $.11^2$ + $.1^2$ + $.09^2$)

= .1 * sqrt(90 / .1106) = .1* 813.74 = 2.85 (or .1 * sqrt(90 / .1006) = .1*29.91 = 2.99)

df = n-1 = 9, $t_{95\%}$ = 2.26 < 2.85 (and 2.26 < 2.99), so the difference is significant

- Use this confusion matrix for the performance of B on fold 1 to calculate the following measures:

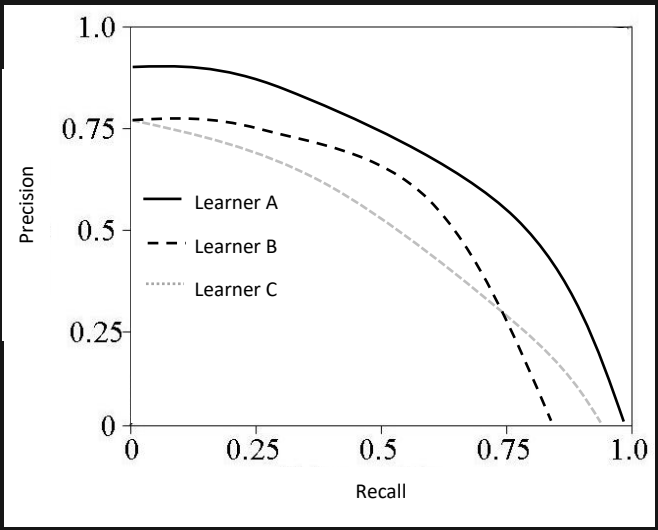
		Actual	
		Positive	Negative
Predicted	Positive	110	20
	Negative	40	130

Accuracy
Accuracy = (TP + TN) / (P + N) = (110+130)/(300) = 0.8

Precision (of all the + that you spotted, how many were actually +?)
TP / (TP + FP) = 110 / (110 + 20) = 0.846

Recall (of all the + that exist, how many did you spot and label as +?)
TP / (TP + FN) = 110 / (110 + 40) = 0.733

- Consider the PR curves below for three learning algorithms, A, B, and C.



Is there a best algorithm according to the PR curves? Why or why not?

Learner A generates a PR curve that is higher than the others for all values of Recall. This dominates the others and is best.

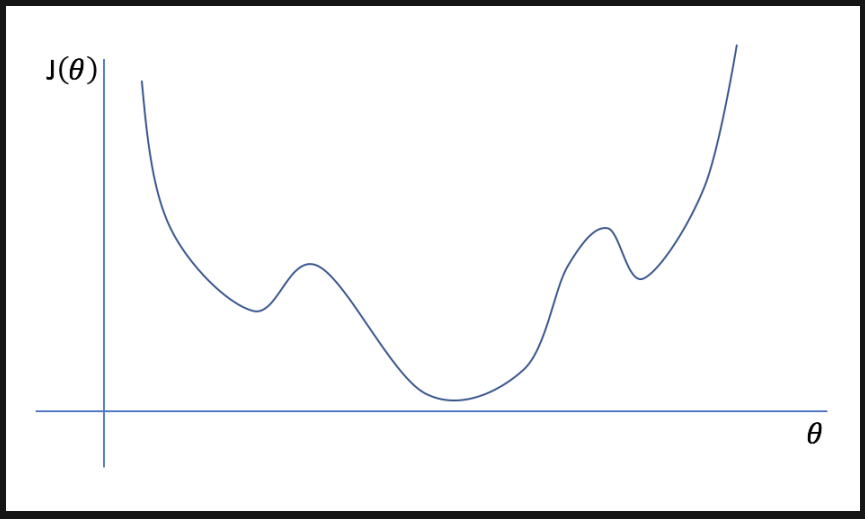
Is there a worst algorithm according to the PR curves? Why or why not?

Learner B performs worst for some values of Recall and Learner C performs worst for others. There is no one worst algorithm according to the PR curves.

Indicate on the great where the curve would lie for an optimal algorithm.

The optimal PR curve lies along the top of the vertical access until Recall=1, at which point it lies along the far right of the horizontal access (like an upside-down L).

- Can the loss function graphed below be used for a gradient descent algorithm? Why or why not?



No, because the function is not convex.

- For each of the following methods, indicate whether they are traditionally used for classification (C), regression (R), neither (N), or both (B). You can optionally provide a brief justification of your answer.

Perceptron	C
Backpropagation neural network	B
Support vector machine	C
Linear regression	R
Logistic regression	C
PCA	N

- Describe two methods to use the classifiers we discussed in class for multi-class classification (the number of classes is > 2).

A set of one vs all classifiers (one for each class label) or a (larger) set of one vs one classifiers (one for each pair of class labels)

- This question focuses on the naïve Bayes classifier.
    a. (4 points) State the simplifying assumption that is made by naïve Bayes classifiers when estimating P(X|Y).

The "naïve" assumption is conditional independence, specifically that each feature a_i is conditionally independent of every other feature a_j ($j \neq i$) given the class value Y_k. This means

$P(a_1,a_2,..,a_n|Y_k) = \prod_i P(a_i|y_j)$. As a result, the NBC prediction is generated using the formula $vNB = argmax_{yj \in Y} P(y_j) \prod_i P(a_i|y_j)$.

    b. Given a classification problem in which the class labels can take *k* different values and the dimension is *d*, state the number of parameters that would need to be estimated **with** and **without** the simplifying assumption.

Without the simplifying assumption we need to use the chain rule. This results in $k*2^d-1$ parameters to estimate. With the simplifying assumption we only need to estimate k*d parameters. The simplifying

assumption is needed because there are not typically enough instances of each combination of feature value to estimate the probabilities required by the chain rule so the result will be more inaccurate in that case.

- Consider three data points with corresponding distances to the decision boundary: x1 (negative class, distance 2.0), x2 (positive class, distance 2.0), and x3 (positive class, distance 10.0). Why would the SVM decision boundary be unaffected by x3 but the logistic regression decision boundary be affected?

  The hinge loss used by SVMs gives zero weight to points far away from the decision boundary but the log-loss used by logistic regression assigns some weight to these points.

  How does the number of support vectors in an SVM affect the likelihood of overfitting the data?

  If a SVM overfits the data, then most or all of the data are support vectors.

- What is the trade-off between bias and variance?

  Bias is error due to erroneous or overly simplistic assumptions in the learning algorithm you're using. This can lead to the model underfitting your data, making it hard for it to have high predictive accuracy and for you to generalize your knowledge from the training set to the test set.

  Variance is error due to too much complexity in the learning algorithm you're using. This leads to the algorithm being highly sensitive to high degrees of variation in your training data, which can lead your model to overfit the data. You'll be carrying too much noise from your training data for your model to be very useful for your test data.

  The bias-variance decomposition essentially decomposes the learning error from any algorithm by adding the bias, the variance and a bit of irreducible error due to noise in the underlying dataset. Essentially, if you make the model more complex and add more variables, you'll lose bias but gain some variance — in order to get the optimally reduced amount of error, you'll have to tradeoff bias and variance. You don't want either high bias or high variance in your model.

- Consider the formula below for selecting weights and bias that minimize a loss function with a regularizer.

$$min_{w,b} \sum_n L(y_n, w.x_n + b) + \lambda R(w, b)$$

  What will the R(w,b) term look like for a 1-norm (L1-norm)?

  $\sum_d |w_d|$

What will the R(w,b) term look like for a 1-norm (L2-norm)?

$(\sum_d |w_d|^2)^{1/2}$

- Suppose we want to use a naïve Bayes classifier to learn the concept of whether the Cougar football team will win or home based on the attributes Location, Month, and Injuries. Based on the training data given below, what will the classifier predict for a new data point with attributes Location=Home, Month=Nov, and Injuries=Light? Show all of your work.

#	Location	Month	Injuries	Win?
1	Home	Oct	Light	No
2	Home	Oct	Light	Yes
3	Home	Oct	Light	No
4	Away	Oct	Light	Yes
5	Away	Oct	Heavy	No
6	Away	Nov	Heavy	Yes
7	Away	Nov	Heavy	No
8	Away	Nov	Light	Yes
9	Home	Oct	Heavy	No
10	Home	Nov	Heavy	Yes

P(Yes|x) = P(Yes) * P(Home|Yes) * P(Nov|Yes) * P(Light|Yes) = 0.5 * 2/5 * 3/5 * 3/5 = 0.072

P(No|x) = P(No) * P(Home|No) * P(Nov|No) * P(Light|No) = 0.5 * 3/5 * 1/5 * 2/5 = 0.024