CS 181 Spring 2023 Section 3 Notes (Classification)

Jeffrey Xu, Derek Zheng

1 Classification

Please refer to the Skills Check for Midterm document. You should be working to answer the following questions during this week's sections and OHs as well as on HW#2.

1. **Geometric Interpretation of Classification**
   - **(Math)** Can you define soft and hard classification?
   - **(Concept)** Can you describe under which real-life data scenarios we would prefer soft classification over hard classification?
   - **(Math)** Can you derive logistic regression as soft classification using (signed) Euclidean distance to a decision boundary given by the equation $\mathbf{w}^\top \mathbf{x} = 0$?
   - **(Math)** Can you derive logistic regression as soft classification using (signed) Euclidean distance to a decision boundary given by the equation $\mathbf{w}^\top \phi(\mathbf{x}) = 0$, where ϕ is a feature map?
   - **(Math)** Can you visualize $\mathbf{w}^\top \phi(\mathbf{x}) = 0$, where ϕ is a feature map, as a decision boundary for binary soft-classification?
   - **(Math)** Do you know how to make hard classification decisions by thresholding soft classification probabilities?
   - **(Concept)** Can you describe the role that the sigmoid function plays in logistic regression?
   - **(Math)** Given a different function $\sigma : \mathbb{R} \to (0, 1)$, and a decision boundary

     $$\mathbf{w}^\top \phi(\mathbf{x}) = 0$$

     can you derive a soft classification model – i.e. can you model $p(y = 1|\mathbf{x})$?
   - **(Concept)** Given a decision boundary $\mathbf{w}^\top \phi(\mathbf{x}) = 0$ for binary classification, can you reason about the classification probability of points close to the boundary and about points far from the boundary? Can you provide justification for why the classfication probability should behave in real-life as you described? Can you provide reasons why classification probability should **not** behave in real-life as you described?
2. **Probabilistic Interpretation of Classification**
   - **(Math)** For a single observation $\mathbf{x}, y$ can you write down the likelihood for logistic regression over an arbitrary basis given by feature map ϕ?
   - **(Concept)** Can you explain why this likelihood is a reasonable model for classifaction (why is this a good model for $\mathbf{x}, y$ where y is binary)?

- **(Math)** Can you write down the joint likelihood for logistic regression given a dataset and a feature map ϕ?
- **(Math)** Given a set of weights $\mathbf{w}$ for a logistic regression model, can you derive the impact a single weight $\mathbf{w}_d$ has on the log-odds?

3. **Inference and Optimization**
    - **(Concept)** If you are unable to analytically optimize your loss function, given the gradient, can you derive the steps of gradient descent (by which I mean describe the steps and justify why each step is reasonable)?
    - **(Concept)** What are the design choices (hyperparameters) of gradient descent (learning rate and stopping condition)? How does each choice affect the behavior of gradient descent?
    - **(Concept)** Do you know the two common failure modes of gradient descent (slow convergence and oscillation)? Do you know how to detect each type of failure mode? Do you know some common fixes for these failure modes?
    - **(Math)** Can you show why it is not possible to direcly optimize accuracy?
    - **(Math)** Can you find the gradient of the joint log-likelihood for logistic regression given a dataset and a feature map ϕ?
    - **(Math)** Can you show that analytic optimization of the logistic regression joint log-likelihood is not feasible?
    - **(Math)** Can you explain why computationally optimizing the logistic regression joint log-likelihood using gradient descent would be problemmatic (i.e. do you know where all the places you can encounter numeric instability)?
4. **6. Model Evaluation**
    - **(Math)** Can you define accuracy? Can you given an example in which classfication accuracy is a misleading metric?
    - **(Math)** Can you define the ROC?
    - **(Math)** Can you define false positive (negative) rate, true positive (negative) rate?
    - **(Concept)** Can you given examples of application where one rate should be prioritized above others?
    - **(Concept)** Can you interpret the meaning of a point on the ROC in terms of classifier behavior given a particular classification threshold?
    - **(Math)** Can you derive the ROC of a random classifer and a perfect classifier?
    - **(Concept)** Can you use the ROC to compare and select models?
    - **(Concept)** Can you give examples for when to prioritize accuracy and when to prioritize ROC in model selection/evaluation?

1.1 What is Classification?

In the last couple sections, we explored ways of predicting a continuous, real-number target, whether it was through KNN, Kernelized Regression, or Ordinary Least-Squares (OLS.) In this section, we're going to focus on a different problem: our target output is now going to be discrete-valued. Our "y's" now will come as discrete classes, such as predicting star types or filtering emails for spam. This type of problem, one where we make a prediction by choosing between finite class options, is known as **classification**.

2 Probabilistic Classification

2.1 Takeaways

2.1.1 Probabilistic Model

1. In general, our goal with probabilistic modeling is to model $p(y|\mathbf{x})$.
2. Intuitively, and importantly, this means that we do not care about how $\mathbf{x}$ is generated – we just care about the following: *given* $\mathbf{x}$, what is the distribution of y?

A specific type of probabilistic modeling in the *binary case* is **logistic regression**.

2.1.2 Logistic Regression

1. In binary logistic regression, we only have *two* classes, which we will denote as 0 and 1. Note that we are *not* using -1 and 1 anymore!
2. We will model our probability distribution for the label of a certain data point y, given its features $\mathbf{x}$, as follows, for some weights $\mathbf{w}$ and intercept w_0:

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

$$p(y = 0|\mathbf{x}) = 1 - \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

   Note: In some texts, we might just see $\mathbf{w}^T\mathbf{x}$ instead of $\mathbf{w}^T\mathbf{x} + w_0$, because of the bias trick. They mean the same thing.

3. The σ denotes the **sigmoid function**, which is defined as:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

   The **sigmoid function** is important because it takes any value z on the real line (i.e., $\mathbb{R}$) and returns an output on $(0, 1)$. This is very important because probabilities must be between 0 and 1. For clarity, $\exp(-z) = e^{-z}$.

4. To find the best weights $\mathbf{w}$, we need to set up a loss function. We will use what is called the **negative log-likelihood** loss function.
   (a) Intuitively, we want to find $\mathbf{w}$ that maximizes the *likelihood* of our data.

(b) However, unlike ordinary least squares linear regression from last week, there is no clean-cut analytical solution. Thus, we have to use gradient descent.

(c) The problem is – gradient descent is used to minimize a function. Well, minimizing the negative likelihood (i.e., likelihood $\times$ -1) is the same thing as maximizing the likelihood. Furthermore, we know that minimizing the log-likelihood is the same thing as minimizing the likelihood, but more mathematically tractable.

(d) By definition, a "loss function" is something that we want to minimize when trying to find our optimal weights $\mathbf{w}$. Thus, we use the negative log-likelihood as our loss function.

(e) In practice, we will use an iterative method like (stochastic) gradient descent to minimize our negative log-likelihood loss and obtain our optimal weights $\mathbf{w}$.

(f) With a training data set of N points of the form $(\mathbf{x_i}, y_i)$, our negative log-likelihood loss (which, fun fact, is also sometimes called the "cross-entropy loss") is defined as follows:

$$\mathcal{L}(\theta) = -\sum_{n=1}^{N} \left(y_n \ln p(y_n = 1|\mathbf{x}_n; \theta) + (1 - y_n) \ln p(y_n = 0|\mathbf{x}_n; \theta)\right)$$

(g) After fitting our model, if we want to predict the class y^* for a new data point $\mathbf{x}^*$, we will calculate the following class probabilities, and assign this new data point to which ever class has the higher probability.

$$p(y^* = 1|\mathbf{x}^*; \mathbf{w})$$

$$p(y^* = 0|\mathbf{x}^*; \mathbf{w}) = 1 - p(y^* = 1|\mathbf{x}^*; \mathbf{w})$$

(h) Because of the $\mathbf{w}^T\mathbf{x}$, logistic regression has *linear* decision boundaries! Yes, the sigmoid function isn't a straight line/hyperplane, but the $\mathbf{w}^T\mathbf{x}$ ensures that we have a linear decision boundary!

(i) Remarks on Notation:

i. In some texts, you may see a $\hat{y}_i$ term. Don't be scared! In the context of logistic regression,

$$\hat{y}_i = p(y_i = 1|\mathbf{x_i}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0), \text{ or with the bias trick, just } \sigma(\mathbf{w}^T\mathbf{x}).$$

ii. Instead of $y_i = 1$, in some course materials, you might sometimes see $y_i = C_1$. They mean literally the same thing. Analogously, C_2 corresponds to class 0.

iii. In the expression for the negative log-likelihood loss above, θ refers to the parameters of our model. In the context of logistic regression, θ and $\mathbf{w}$ (with maybe w_0) mean the same thing.

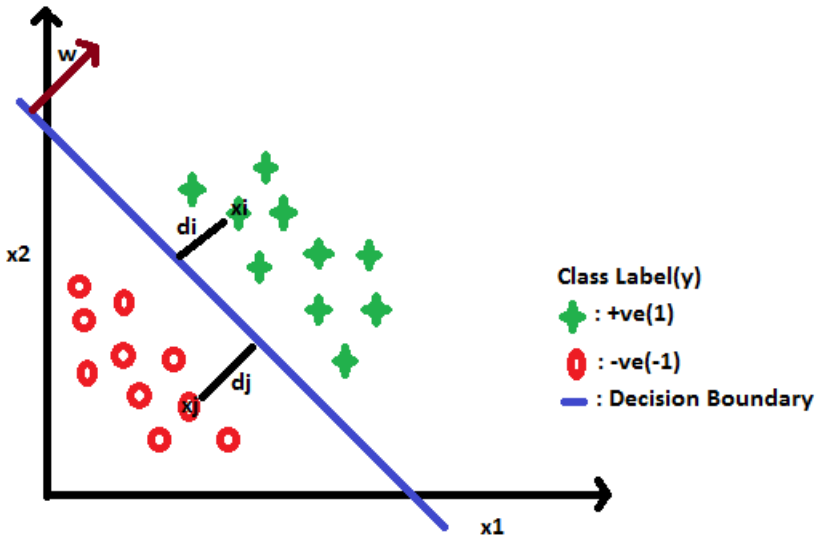
iv. $\mathcal{L}$ refers to the **loss function**, while L refers to the likelihood, and ℓ refers to the log-likelihood. Be sure to check the context in which these symbols are used!

(j) We can also extend logistic regression to the multi-class case using the softmax function. See pg. 42 in *Undergraduate Fundamentals of Machine Learning* for a deeper treatment of this extension.

2.2 Geometric Interpretation of Logistic Regression

As we know now, in logistic regression, the goal is to predict a binary outcome, such as whether a sample belongs to class A or class B. The logistic function maps an input to a probability, which signifies how likely it is to belong to class A. It does this by using the sigmoid function, which maps to a number between 0 and 1 (a probability).

What's important to note here is that the logistic regression model creates a decision boundary in feature space that separates samples into two classes, A and B. This is shown here:



Samples that are closer to the decision boundary are more uncertain, and therefore have a probability closer to 0.5 of belonging to either class. Samples that are further away from the decision boundary have a higher probability of belonging to one class or the other.

Essentially, the logistic regression model uses the distance of a sample from the decision boundary to predict its class, with samples that are further from the boundary having a higher probability of belonging to a specific class, and samples that are closer to the boundary having a higher probability of being ambiguous.

2.3 Interpretation of Logistic Regression as a Generalized Linear Model

Logistic regression can be considered a type of generalized linear model, which is a framework for modeling linear relationships between the response variable and the predictor variables. The key difference between logistic regression and traditional linear regression is the type of response variable being modeled.

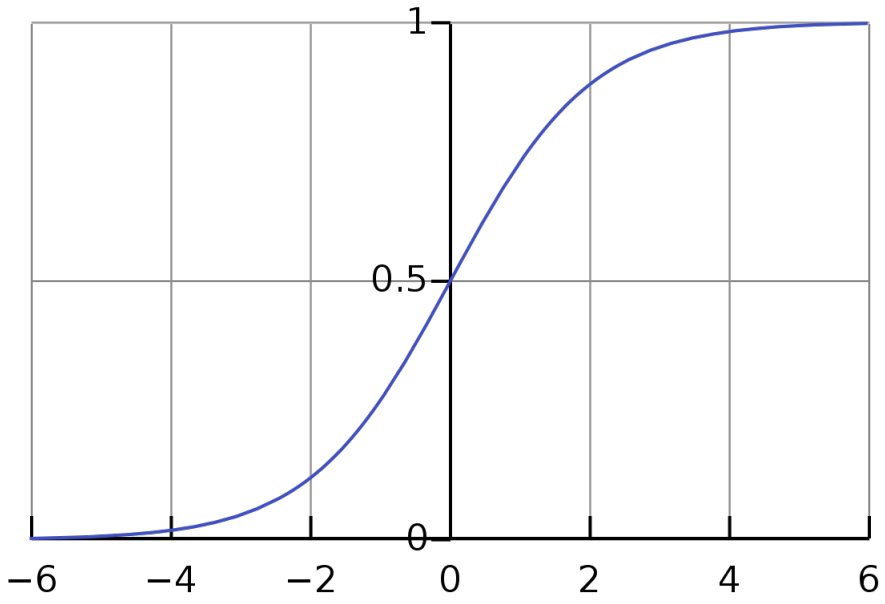
In traditional linear regression, the response variable is continuous and unbounded, while the response variable in logistic regression is binary, taking on only two possible values (e.g. 0 or 1, Yes

or No, etc.). To achieve a binary value, the logistic regression model transforms the linear relationship between the predictor variables and the response variable using the logistic or sigmoid function.

The logistic function takes the form of a S-shaped curve that maps any real number to the range (0,1), which can be interpreted as a probability. We have the formula above, but once again it is:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

Its graph is depicted here:



So ultimately, we can see here that from an output of a linear model, we can transform our $\hat{y}$, with the addition of other scale or location transformations, such that it maps to a value between 0 and 1 with the sigmoid function.

This is how we can arrive at a classification probability through something that started as a linear model, showing how logistic regression can be thought of as a generalized linear model.

3 Other Classification Models

3.1 Decision Trees

Decision Trees can be used as a simple yet sufficiently powerful method for classification. This method **divides an input space into key regions**, with **each region representing a specific label**. Some things to note:

- The decision tree algorithm seeks out an optimal boundary, or boundaries, until reaching a **stopping condition**.
- Stopping conditions for decision trees generally fall into two categories:
  - **Maximum depth** stopping conditions ensure that the depth (or height) of a decision tree does not pass a certain threshold.
  - **Minimum purity** stopping conditions ensure that the resulting boundaries from decision trees meet a minimum threshold for 'purity', which describes how similar labels inside a node or region are.

3.2 KNN Classification

In previous weeks we learned about KNN (K-Nearest Neighbors) for *regression* tasks, but the same KNN methods can be applied to **classification** as well.

For some new data point x, KNN Classifiers predict some class $\hat{y}$ by selecting the most common class label l of the k closest points to x. "Closeness" can have different meanings based on the scenario, but points are classified based on labels of points that are most similar to them.

3.3 Random Forest

Based on our search algorithm for the optimal criteria for decision trees, we could expect there to be significant fluctuations in the resulting decision tree for each training run. One technique that we learned about last week is *ensembling*, which combines some number of decision trees into a common model, called a **Random Forest**. Due to this ensembling technique, which combines models produced by multiple training runs, we can frequently get more accurate and generalizable results.

4 Evaluation & Tradeoffs

Training models and selecting hyperparameters can seem like a daunting task, especially as there are so many considerations for each choice. Here we'll discuss some common tradeoffs to make when thinking about different classifiers:

- **Bias-Variance Tradeoff:** How well does the train/test data represent the overall distribution? Should we optimize for generalizability or accuracy on train/test sets? This tradeoff is still crucial to consider when evaluating your classifiers:
  - How complex should your decision tree be?

  - What is the best k for you K-Neighbors classifier?
  - What complexity would you expect from a logistic classifier?
- **Regularization:** Including a regularizing term could decrease the accuracy of models performing on train/test data, but produce more generalizable or desirable models to use. In certain cases, we might want to reduce the total number of weight parameters (think L1 Regularization) or the magnitude of our weights (think L2 Regularization.
  - Does your classifier or decision boundary need a regularization term to generalize to other data?
  - How noisy is your data?
- **Ensemble:** Finally, ensembling offers another strategy to mitigate under/overfitting in your model training. One of the best examples is for decision trees — does one tree suffice to classify your points across all examples? How much variance is there in the decision boundaries that it finds?

5 Confusion Matrices, TPR, FPR and AUC

In classification tasks, we can run into all sorts of errors. What happens if our training set is highly unbalanced between the classes?

What kind of biases can this create in our model? In this section, we'll discuss metrics and associated strategies for measuring the performance of our models so that they achieve what we want.

A common evaluation metric for classification models is the receiver operating characteristic (ROC) curve, which plots the true positive rate (TPR) against the false positive rate (FPR). The TPR and FPR are calculated based on the results of a confusion matrix.

5.1 Confusion Matrix

What is a Confusion Matrix? How does this help us with improving our model? Before we dive in, let's review some classification definitions to know.

A confusion matrix is a table that is frequently used to summarize the performance of a classification model. It contains four elements: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

- True Positive/True Negative (TP/TN): Predicted values align with true values (a positive is actually a Positive, negative actually a negative).
- False Positive/False Negative (FP/FN): Falsely (incorrectly) predicted values for positive or negative labels (mislabeling a False as a True = False Positive).

5.2 TPR and FPR

True Positive Rate (TPR) is defined as the ratio of true positive predictions to the total number of positive instances, regardless of whether they were correctly or incorrectly classified. TPR is also

known as Sensitivity or Recall.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1}$$

False Positive Rate (FPR) is defined as the ratio of false positive predictions to the total number of negative instances, regardless of whether they were correctly or incorrectly classified:

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{2}$$

5.2.1 TNR and FNR

To complete the picture, we have 2 more definitions that are worth mentioning — True Negative Rate (TNR) and False Negative Rate (FNR).

True Negative Rate describes the ratio of true negative predictions to the total number of negative instances, regardless of classification:

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} = 1 - \text{FPR} \tag{3}$$

False Negative Rate describes the ratio of false negative predictions to the total number of positive instances, regardless of classification:

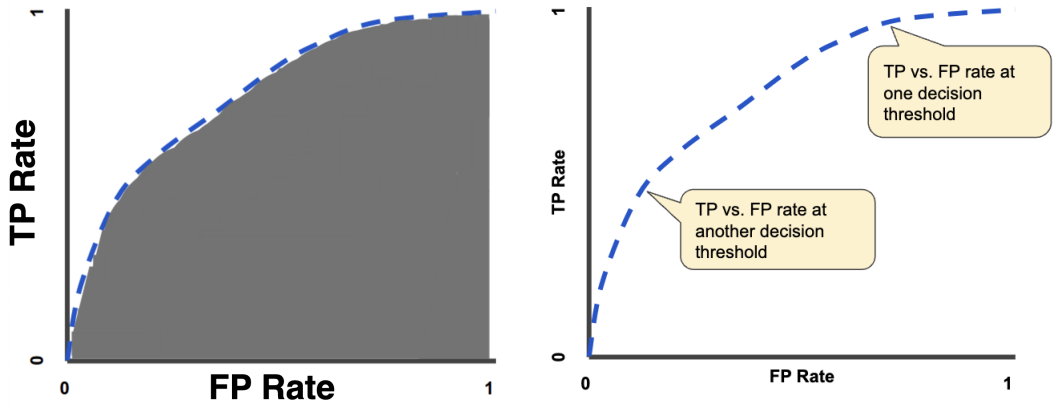
$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} = 1 - \text{TPR} \tag{4}$$

5.3 ROC Curve and AUC

5.3.1 ROC Curve

The ROC Curve, or the Receiver Operating Characteristics Curve, is used to plot TPR and FPR at different classification thresholds. Depending on a model's classification threshold, there may be a lower false negative rate, but as a consequence a higher true positive and false positive rates.

Consider a model that classifies everything as positive, creating a perfect TPR and FNR (threshold of 0). While this doesn't seem desirable, we can select thresholds from $t = [0, 1]$ that optimize for the best tradeoff between TPR and FPR of the model. An ROC curve helps us visualize the changes to our rates depending on the model threshold. Remember, the best model on the ROC curve has the highest TPR and lowest FPR.



Source: Google ML Crash Course

5.3.2 AUC

How do we manage this tradeoff? We can evaluate this tradeoff using the ROC curve and computing the area under it!

We have a great tool at our disposal: **Area Under (the ROC) Curve**, or **AUC**, describes the total area *underneath* the ROC curve for our model. Note that because the ROC curve is graphed on a 1 x 1 plot, AUC has an upper bound of 1 (total area of the graph) and lower bound of 0.

Here are some things to remember about using AUC:

- AUC can simply be thought of as the integral of the ROC curve from 0 to 1 (considering FPR as our "x" axis).
- Ideally, our classifier should perform better than random — a random classifier has an ROC curve that is a straight line from (0,0) to (1,1). Based on this, the AUC of a random classifier would be 0.5.
- On the flip side, a perfect classifier would have an AUC of 1 — we want to strive for classification models that perform as close to 1 as possible.
- AUC is helpful to compare across classifiers, not only against a random one.

6 Exercises

(Probabilistic Classification) Given a dataset $\mathcal{D} = \{(-\pi, 1), (0, 0), (\pi, 1)\}$ for binary classification. For each of the following feature maps,

- $\boldsymbol{\phi}(x) = [1, x]^T$
- $\boldsymbol{\phi}(x) = [1, x, x^2]^T$
- $\boldsymbol{\phi}(x) = [1, x, x^4]^T$
- $\boldsymbol{\phi}(x) = [1, \cos x]^T$

Answer the following questions:

- Plot the dataset in *input* space, represent points in class 1 using $\times$, represent points in class 0 using $\circ$. Draw the decision boundary that would perfectly classify this dataset.
- Write down the joint log-likelihood of a logistic regression model with the given feature map.
- Determine if a logistic regression model with the given feature map is capable of perfectly classifying $\mathcal{D}$.
- If so, by trial-and-error or inspection, find parameters $\mathbf{w}$ of the logistic regression model that correctly classifies the data-points. For this set of parameters $\mathbf{w}$:
  - Plot the ROC curve by hand and compute the AUC
  - Describe the affect of changing the classification threshold on your decision boundary
  - Derive the impact on log-odds for each parameter in $\mathbf{w}$
- Are the parameters $\mathbf{w}$ you found above the MLE of $\mathbf{w}$? (*Hint:* check the gradient)
- Are the parameters $\mathbf{w}$ you found above the global optimum of ℓ_2-regularized logistic regression? (*Hint:* check the gradient)

Variations:

- Repeat the above for your own choice of feature map ϕ, and your own 1D binary classification dataset of three or four points.
- Repeat the above for $\mathcal{D} = \{(-\pi, 0, 1), (-\pi, -1, 0), (\pi, 0, 2), (\pi, -1, 1)\}$ for three-class classification $y = 0, 1, 2$. Try different rearrangements of these four points and make up your own feature map.

Solution:

- There are a few examples of how this could be done perfectly — use your judgement! To plot each point, makes ure to transform your input data by plugging it into the feature map. For each basis, here's an example of a function that could perfectly classify the data:
  - $\sin(x_1)$
  - $x_2 = \pi$
  - $x_2 = \pi^4/2$
  - $x_1 = 0$
- Start with likelihood of each logistic regression model (remember, likelihood should consider data $\mathcal{D}$, your decision boundaries), then take the log to derive the joint log-likelihood.

$$\begin{aligned}\ell(\mathbf{y}|\mathcal{D};\theta) &= \sum_{n=1}^{3} \left(y_n \ln p(y_n = 1|\vec{x_n};\theta) + (1-y_n)\ln p(y_n = 0|\vec{x_n};\theta)\right) \\ &= \sum_{n=1}^{3} \left(y_n \ln(\sigma(\mathbf{w}^T\vec{x}_n)) + (1-y_n)\ln(1-\sigma(\mathbf{w}^T\vec{x}_n))\right)\end{aligned}$$

- Answers to perfect classification of $\mathcal{D}$ with each of the 4 feature maps:
  a. No, this choice of basis cannot perfectly separate the data.
  b. Yes. Set $\mathbf{w} = [-1, 0, 1]^T$.
  c. Yes. Set $\mathbf{w} = [-1, 0, 1]^T$.
  d. Yes. Set $\mathbf{w} = [0, -1]^T$.
- To plot ROC curves, evaluate each classifier at different thresholds of $t : [0, 1]$. Plot each point (corresponding to some threshold) and calculate your TPR and FPR to place the point on the ROC curve graph. To calculate AUC, find the area underneath this ROC curve.
- To visualize this tradeoff, consider sorting each of your dataset points $\mathcal{D}$ by probability — adjust your threshold (increasing or decreasing) and note how the threshold change influences how points are classified. For each feature map, tuning the classification threshold will enable you to find the optimal classifier — remember, you want to maximize TPR while minimizing FPR.
- Odds can be interpreted by the probability p of positive classification (arbitrarily, 1) divided by the probability $1 - p$ of negative classification: Odds $= \frac{p}{1-p} = \frac{p}{w}$. Note that we know that $p = p(y_n = 1|\mathbf{x}_n;\theta) = \sigma(\mathbf{w}^T\mathbf{x}_n)$, and thus $q = p(y_n = 0|\mathbf{x}_n;\theta) = 1 - \sigma(\mathbf{w}^T\mathbf{x}_n)$. Recall also that $\sigma(z) = \frac{1}{1+e^{-z}}$. To derive the impact on log-odds for each parameter:

$$\begin{aligned}\text{Odds} &= \frac{\sigma(z)}{1-\sigma(z)} = \frac{\frac{1}{1+e^{-z}}}{\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}} = \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = e^z \\ \text{Log-Odds} &= \log(e^z) = z \\ &= \mathbf{w}^T\mathbf{x}_n \\ &= w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n\end{aligned}$$

  To examine the impact of each parameter of $\mathbf{w}$, we can take the derivative w.r.t. some w_i:

$$\frac{\partial}{\partial w_i}\text{Log-Odds} = \frac{\partial}{\partial w_i} w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = x_i$$

  Thus, each parameter of $\mathbf{w}$ correspondingly weighs a feature of input $\mathbf{x}_n$. This confirms our modeling set up, each parameter w_i weighs a specific feature x_i which impacts Log-Odds.

- To figure out if what we found is the MLE of $\mathbf{w}$, we can check the gradient of our log likelihood with respect to $\mathbf{w}$.

$$\ell(\mathbf{y}|\mathcal{D};\theta) = \sum_{n=1}^{3} \left(y_n \ln(\sigma(\mathbf{w}^T\vec{x}_n)) + (1-y_n)\ln(1-\sigma(\mathbf{w}^T\vec{x}_n))\right)$$

$$\frac{\partial \ell}{\partial \mathbf{w}} = \sum_{n=1}^{3} \frac{y_n}{\sigma(\mathbf{w}^T x_n)}\sigma(\mathbf{w}^T x_n)(1-\sigma(\mathbf{w}^T x_n))x_n + \frac{1-y_n}{1-\sigma(\mathbf{w}^T x_n)}(-\sigma(\mathbf{w}^T x_n))(1-\sigma(\mathbf{w}^T x_n))x_n$$

$$= \sum_{n=1}^{3} y_n(1-\sigma(\mathbf{w}^T x_n))x_n - (1-y_n)\sigma(\mathbf{w}^T x_n)x_n$$

$$= \sum_{n=1}^{3} y_n x_n - \sigma(\mathbf{w}^T x_n)x_n$$

$$= \sum_{n=1}^{3} (y_n - \sigma(\mathbf{w}^T x_n))x_n$$

Plugging in the values we had found previously, we can see that for each of the three sets of weights we found, the gradient was 0. Thus, it must be true that we have found the MLE.

- Now, we can do the same thing, but we are also checking for the regularization term now.

$$= \sum_{n=1}^{3} (y_n - \sigma(\mathbf{w}^T x_n))x_n + 2\lambda \sum w_i$$

The right term has become $2\lambda \sum w_i$, since this is the derivative of the L2 Norm regularization term.
Once again plugging in the weights we found earlier, we can see that we still have a 0 gradient for the $[1, x, x^2]$ and $[1, x, x^4]$ bases. However, for the weights found for the $[1, cosx]$ basis, our gradient becomes negative due to the regularization. So, it is not a global optimum there, but it is a global optimum for the first two mentioned.