18-661 Introduction to Machine Learning

Multi-Class Regression

Spring 2025

ECE – Carnegie Mellon University

Announcements

- Homework 1 grades have been released on Gradescope. If you notice a grading error, please submit a regrade request through Gradescope by this Friday, February 14. Review the posted solutions carefully before submitting a request.
- Homework 2 is due on Friday, February 21. After this lecture, you should be able to attempt most of the problems (except those on SVMs).

Outline

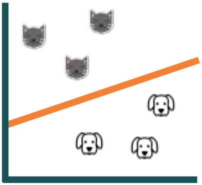
1. Review of Logistic Regression
2. Non-linear Decision Boundaries
3. Multi-class Classification
4. Evaluating Classification Methods

Review of Logistic Regression

Intuition: Logistic Regression

Learn the equation of the decision boundary $\mathbf{w}^\top \mathbf{x} = 0$ such that

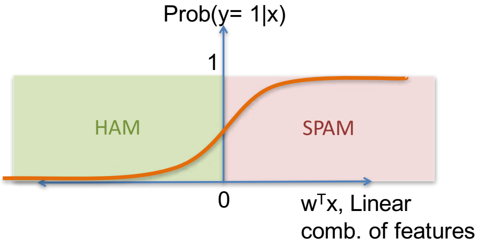
- If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (cat)
- If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (dog)



$y = 0$ for dog, $y = 1$ for cat

Intuition: Logistic Regression

- Suppose we want to output the probability of an email being spam/ham instead of just 0 or 1
- This gives information about the confidence in the decision
- Use a function $\sigma(\mathbf{w}^\top \mathbf{x})$ that maps $\mathbf{w}^\top \mathbf{x}$ to a value between 0 and 1



Probability that predicted label is 1 (spam)

Key Problem: Finding optimal weights $\mathbf{w}$ that accurately predict this probability for a new email

Formal Setup: Binary Logistic Classification

- Input/features: $\boldsymbol{x} = [1, x_1, x_2, \ldots x_D] \in \mathbb{R}^{D+1}$
- Output: $y \in \{0, 1\}$
- Training data: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n), n = 1, 2, \ldots, N\}$
- Model:

$$p(y = 1 | \boldsymbol{x}; \boldsymbol{w}) = \sigma[g(\boldsymbol{x})]$$

where

$$g(\boldsymbol{x}) = w_0 + \sum_d w_d x_d = \boldsymbol{w}^\top \boldsymbol{x}$$

and $\sigma[\cdot]$ stands for the sigmoid function

$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

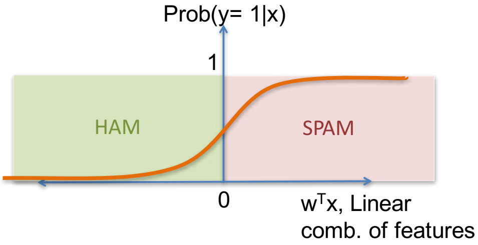
Likelihood Function

Probability of a single training sample $(\boldsymbol{x}_n, y_n)$...

$$p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \begin{cases} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{if} \quad y_n = 1 \\ 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) & \text{otherwise} \end{cases}$$

Simplify, using the fact that y_n is either 1 or 0

$$p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n}[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]^{1-y_n}$$



Probability that predicted label is 1 (spam)

Log Likelihood or Cross Entropy Error

Log-likelihood of the whole training data $\mathcal{D}$

$$P(\mathcal{D}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n; \boldsymbol{w}) = \prod_{n=1}^{N} \left\{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n} [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]^{1-y_n}\right\}$$

$$\log P(\mathcal{D}) = \sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

It is convenient to work with its negation, which is called the cross-entropy error function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Cross-Entropy as a Loss Function

We aim to build a function $h(\boldsymbol{x})$ to predict the true value y associated with $\boldsymbol{x}$. If we make a mistake, we incur a loss

$$\ell(h(\boldsymbol{x}), y)$$

Cross-entropy is also a sum over all data samples:

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

where $h(\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^T \boldsymbol{x}\right)$.

What is the loss function?

$$\ell(h(\boldsymbol{x}_n), y) = -\{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Gradient Descent Update for Logistic Regression

Finding the gradient of $\mathcal{E}(\boldsymbol{w})$ looks very hard, but it turns out to be simple and intuitive.

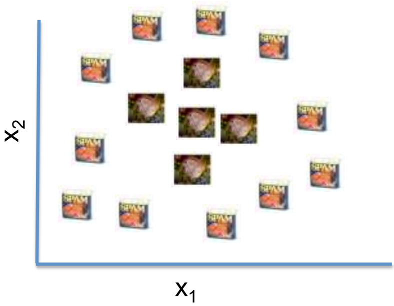
$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

Computing the gradient

$$\begin{aligned}
\frac{\partial \mathcal{E}(\boldsymbol{w})}{\partial \boldsymbol{w}} &= -\sum_n \Big\{ y_n \frac{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]}{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)} \boldsymbol{x}_n \\
&\qquad - (1 - y_n) \frac{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]}{1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)} \boldsymbol{x}_n \Big\} \\
&= -\sum_n \{ y_n [1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)] \boldsymbol{x}_n - (1 - y_n) \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) \boldsymbol{x}_n \} \\
&= \sum_n \underbrace{\{\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) - y_n\}}_{\text{Error of the } n\text{th training sample.}} \boldsymbol{x}_n
\end{aligned}$$

Non-linear Decision Boundaries

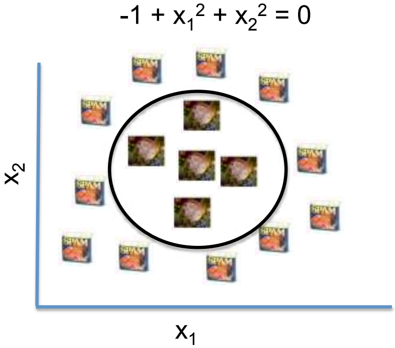
How to Handle More Complex Decision Boundaries?



- This data is not linearly separable...
- Use non-linear basis functions to add more features.

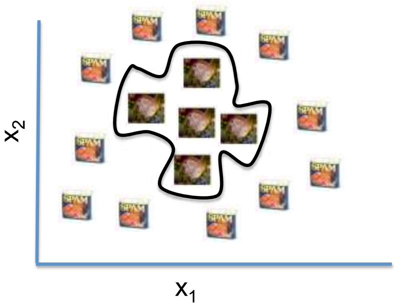
Adding Polynomial Features

- New feature vector is $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2]$
- $\Pr(y = 1|\mathbf{x}) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$
- If $\mathbf{w} = [-1, 0, 0, 1, 1]$, the boundary is $-1 + x_1^2 + x_2^2 = 0$
  - If $-1 + x_1^2 + x_2^2 \geq 0$ declare spam
  - If $-1 + x_1^2 + x_2^2 < 0$ declare ham



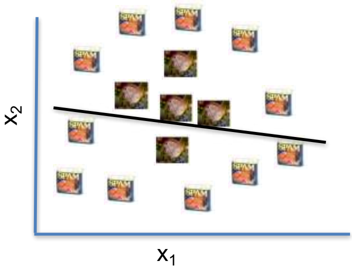
Adding Polynomial Features

- What if we add many more features and define $\mathbf{x} = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, \ldots]$?
- We get a complex decision boundary

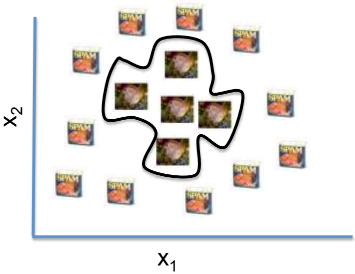


Can result in overfitting and bad generalization to new data points.

Concept-check: Bias-Variance Trade-off



high bias



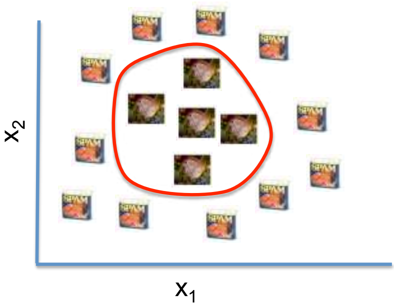
high variance

Solution to Overfitting: Regularization

- Add regularization term to be cross entropy loss function

$$\mathcal{E}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1-y_n)\log[1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\} + \underbrace{\frac{1}{2}\lambda\|w\|_2^2}_{\text{regularization}}$$

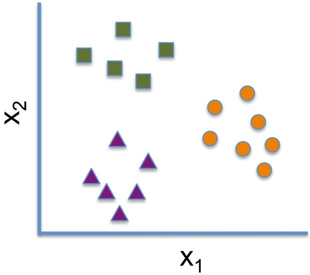
- Perform gradient descent on this regularized function
- Often, we do NOT regularize the bias term w_0



Multi-class Classification

What If There Are More than 2 Classes?

- Dog vs. cat. vs crocodile
- Movie genres (action, horror, comedy, ...)
- Part of speech tagging (verb, noun, adjective, ...)
- ...



Setup

Predict multiple classes/outcomes $C_1, C_2, \ldots, C_M$**:**

- Weather prediction: sunny, cloudy, raining, etc
- Optical character recognition: 10 digits + 26 characters (lower and upper cases) + special characters, etc.

$M =$number of classes

Methods we've studied for binary classification:

- Naïve Bayes
- Logistic regression

Do they generalize to multi-class classification?

Naïve Bayes Is Already Multi-class!

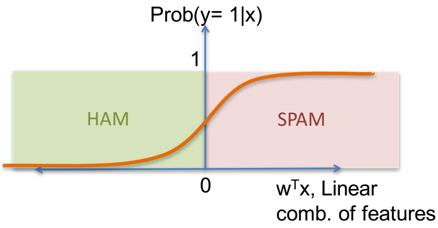
Formal Definition
Given a random vector $\mathbf{X} \in \mathbb{R}^{\mathrm{K}}$ and a dependent variable $Y \in [\mathrm{C}]$, the Naïve Bayes model defines the joint distribution

$$
\begin{aligned}
P(\mathbf{X} = \mathbf{x}, Y = c) &= P(Y = c) P(\mathbf{X} = \mathbf{x} | Y = c) && (1) \\
&= P(Y = c) \prod_{k=1}^{\mathrm{K}} P(\mathrm{word}_k | Y = c)^{x_k} && (2) \\
&= \pi_c \prod_{k=1}^{\mathrm{K}} \theta_{ck}^{x_k} && (3)
\end{aligned}
$$

where x_k is the number of occurrences of the kth word, π_c is the prior probability of class c (which allows multiple classes!), and θ_{ck} is the weight of the kth word for the cth class.

Logistic Regression for Predicting Multiple Classes?

- The linear decision boundary that we optimized was specific to binary classification.
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) \geq 0.5$ declare $y = 1$ (spam)
  - If $\sigma(\mathbf{w}^\top \mathbf{x}) < 0.5$ declare $y = 0$ (ham)
- How to extend it to multi-class classification?

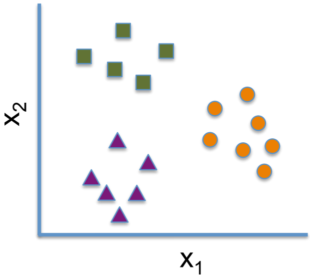


$y = 1$ for spam, $y = 0$ for ham

Idea: Express as multiple binary classification problems

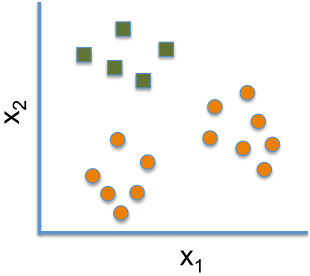
The One-versus-Rest or One-versus-All Approach

- For each class c, change the problem into binary classification
  1. Relabel training data with label c, into POSITIVE (or '1').
  2. Relabel all the rest data into NEGATIVE (or '0').
- Repeat this multiple times: Train C binary classifiers, using logistic regression to differentiate the two classes each time.



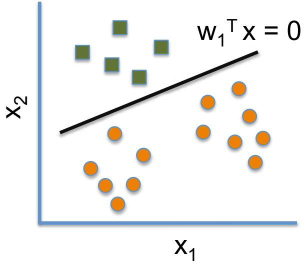
The One-versus-Rest or One-versus-All Approach

- For each class c, change the problem into binary classification
  1. Relabel training data with label c, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train C binary classifiers, using logistic regression to differentiate the two classes each time.



The One-versus-Rest or One-versus-All Approach

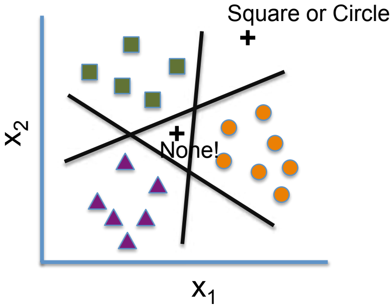
- For each class c, change the problem into binary classification
  1. Relabel training data with label c, into POSITIVE (or '1')
  2. Relabel all the rest data into NEGATIVE (or '0')
- Repeat this multiple times: Train C binary classifiers, using logistic regression to differentiate the two classes each time.



The One-versus-Rest or One-versus-All Approach

How to combine these linear decision boundaries?

- There is ambiguity in some of the regions (the 4 triangular areas).
- How do we resolve this?

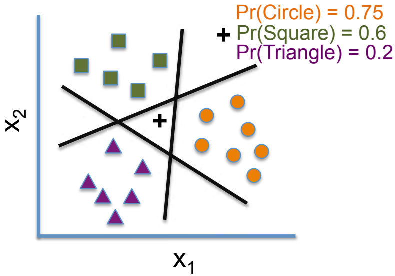


The One-versus-Rest or One-versus-All Approach

How to combine these linear decision boundaries?

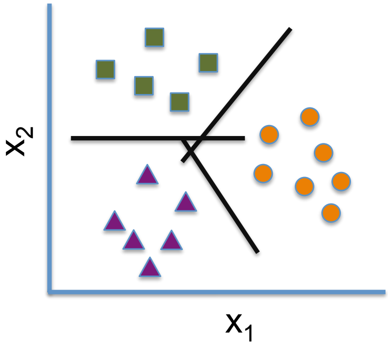
- Use the confidence estimates $\Pr(y = 1|\mathbf{x}) = \sigma(\mathbf{w}_1^\top \mathbf{x})$, $\ldots \Pr(y = C|\mathbf{x}) = \sigma(\mathbf{w}_C^\top \mathbf{x})$
- Declare class c^* that maximizes

$$c^* = \arg \max_{c=1,\ldots,C} \Pr(y = c|\mathbf{x}) = \sigma(\mathbf{w}_c^\top \mathbf{x})$$



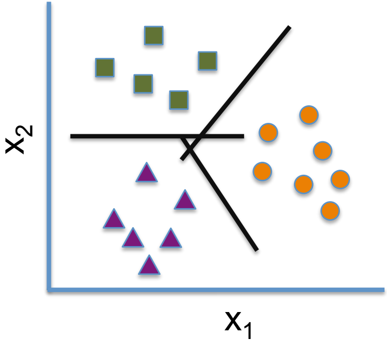
The One-versus-One Approach

- For each **pair** of classes c and c', change the problem into binary classification.
  1. Relabel training data with label c, into POSITIVE (or '1')
  2. Relabel training data with label c' into NEGATIVE (or '0')
  3. **Disregard** all other data



The One-versus-One Approach

- How many binary classifiers for C classes? $C(C-1)/2$
- How to combine their outputs?
- Given $\boldsymbol{x}$, count the $C(C-1)/2$ votes from outputs of all binary classifiers and declare the winner as the predicted class.
- Use confidence scores to resolve ties.



Contrast These Approaches

Number of binary classifiers to be trained

- One-versus-All: C classifiers.
- One-versus-One: $C(C-1)/2$ classifiers – bad if C is large

Effect of relabeling and splitting training data

- One-versus-All: imbalance in the number of positive and negative samples can cause bias in each trained classifier.
- One-versus-One: each classifier trained on a small subset of data (only data in two classes), which can result in high variance.

Any other ideas?

- Hierarchical classification – we will see this in decision trees
- Multinomial logistic regression – directly output probabilities of y being in each of the C classes.

Multinomial Logistic Regression

Intuition:
from the decision rule of our Naïve Bayes classifier

$$
\begin{aligned}
y^* &= \arg\max_c P(y = c|\boldsymbol{x}) = \arg\max_c \log p(\boldsymbol{x}|y = c)p(y = c) \\
&= \arg\max_c \log \pi_c + \sum_k x_k \log \theta_{ck} = \arg\max_c \boldsymbol{w}_c^\top \boldsymbol{x}
\end{aligned}
$$

Essentially, we are comparing

$$
\boldsymbol{w}_1^\top \boldsymbol{x}, \boldsymbol{w}_2^\top \boldsymbol{x}, \cdots, \boldsymbol{w}_\mathrm{C}^\top \boldsymbol{x}
$$

with **one** for each category.

First Try

So, can we define the following conditional model?

$$P(y = c|\boldsymbol{x}) = \sigma[\boldsymbol{w}_c^\top \boldsymbol{x}].$$

This would **not** work because:

$$\sum_c P(y = c|\boldsymbol{x}) = \sum_c \sigma[\boldsymbol{w}_c^\top \boldsymbol{x}] \neq 1,$$

so each summand can be any number (independently) between 0 and 1.

But we are close!
Learn the C linear models jointly to ensure this property holds!

Multinomial Logistic Regression

- **Model:** For each class c, we have a parameter vector $\boldsymbol{w}_c$ and model the posterior probability as:

$$P(c|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_c^\top \boldsymbol{x}}}{\sum_{c'} e^{\boldsymbol{w}_{c'}^\top \boldsymbol{x}}} \qquad \leftarrow \qquad \textit{This is called the softmax function.}$$

- **Decision boundary:** Assign $\boldsymbol{x}$ with the label that is the maximum of posterior:

$$\arg\max_c P(c|\boldsymbol{x}) \rightarrow \arg\max_c \boldsymbol{w}_c^\top \boldsymbol{x}.$$

How Does the Softmax Function Behave?

Suppose we have

$$\boldsymbol{w}_1^\top \boldsymbol{x} = 100, \quad \boldsymbol{w}_2^\top \boldsymbol{x} = 50, \quad \boldsymbol{w}_3^\top \boldsymbol{x} = -20.$$

We would pick the winning class label 1.

Softmax translates these scores into well-formed conditional probabilities

$$P(y = 1|\boldsymbol{x}) = \frac{e^{100}}{e^{100} + e^{50} + e^{-20}} < 1$$

- Preserves relative ordering of scores.
- Maps scores to values between 0 and 1 that also sum to 1.

Parameter Estimation for Multinomial Logistic Regression

Discriminative approach: Maximize conditional likelihood

$$\log P(\mathcal{D}) = \sum_n \log P(y_n | \boldsymbol{x}_n)$$

We will change y_n to $\boldsymbol{y}_n = [y_{n1} \; y_{n2} \; \cdots \; y_{nC}]^\top$, a C-dimensional vector using 1-of-C encoding.

$$y_{nc} = \begin{cases} 1 & \text{if } y_n = c \\ 0 & \text{otherwise} \end{cases}$$

Ex: if $y_n = 2$, then, $\boldsymbol{y}_n = [0 \; 1 \; 0 \; 0 \; \cdots \; 0]^\top$.

$$\Rightarrow \sum_n \log P(y_n | \boldsymbol{x}_n) = \sum_n \log \prod_{c=1}^{C} P(c | \boldsymbol{x}_n)^{y_{nc}} = \sum_n \sum_c y_{nc} \log P(c | \boldsymbol{x}_n)$$

Cross-entropy Error Function

Definition: negative log-likelihood

$$\begin{aligned}\mathcal{E}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_C) &= -\sum_n \sum_c y_{nc} \log P(c|\boldsymbol{x}_n) \\ &= -\sum_n \sum_c y_{nc} \log \left(\frac{e^{\boldsymbol{w}_c^\top \boldsymbol{x}_n}}{\sum_{c'} e^{\boldsymbol{w}_{c'}^\top \boldsymbol{x}_n}} \right)\end{aligned}$$

Properties of cross-entropy

- Convex in the **w** vectors, therefore local minimum = global optimum
- Optimization requires numerical procedures, analogous to those used for binary logistic regression.

Evaluating Classification Methods

Loss Functions

$$\mathcal{E}(\boldsymbol{w}) = -\sum_{n} \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

- Easy to optimize!
- Average loss over the (training, validation, test) dataset
- ...but what does it mean?

Interpretable Classification Metrics

True positive	False positive
False negative	True negative

- Measure the accuracy within each class
- Accounts for imbalance between classes

- Sensitivity: true positive rate

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- Specificity: true negative rate

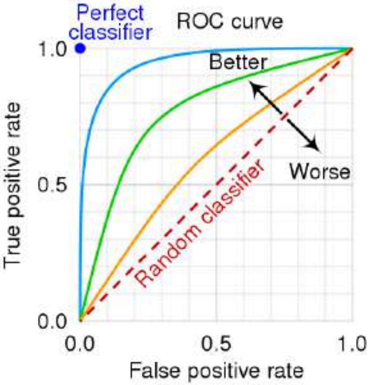
$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

- Precision: positive predictive value

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

These metrics are difficult to optimize directly, but they have the advantage of being easily interpretable.

Combining These Metrics: the ROC Curve



Receiver Operating Characteristic (ROC)

- Define a "threshold" for the positive/negative split
- Increasing the threshold: more samples are predicted to be positive
- Area Under the ROC Curve: want this as large as possible

You Should Know

- How to generalize logistic regression to handle nonlinear decision boundaries.
- How to handle multiclass classification: one-versus-all, one-versus-one, multinomial regression.
- How to measure (binary) classification accuracy