Machine Learning

CSE 142

Xin (Eric) Wang

Wednesday, October 6, 2021

Today

- Classification, Ch. 2

Note

- Kaggle: https://www.kaggle.com

All Competitions

Active Completed InClass All Categories Default Sort

Jane Street Market Prediction — $100,000
Test your model against future real market data
Featured • a month to go • Code Competition • 2375 Teams

HuBMAP - Hacking the Kidney — $60,000
Identify glomeruli in human kidney tissue images
Research • 2 months to go • Code Competition • 855 Teams

RANZCR CLiP - Catheter and Line Position Challenge — $50,000
Classify the presence and correct placement of tubes on chest x-rays to save lives
Featured • 2 months to go • Code Competition • 506 Teams

VinBigData Chest X-ray Abnormalities Detection — $50,000
Automatically localize and classify thoracic abnormalities from chest radiographs
Featured • 3 months to go • 265 Teams

Acea Smart Water Analytics — $25,000
Can you help preserve "blue gold" using data to predict water availability?
Analytics • a month to go

Classifier margin and loss function

Review

- True class function $c(x) = \begin{cases} +1 \text{ for positive training examples} \\ -1 \text{ for negative training examples} \end{cases}$
- The scoring classifier assigns a margin $z(x)$ to each instance x:

$$z(x) = c(x)\hat{s}(x)$$

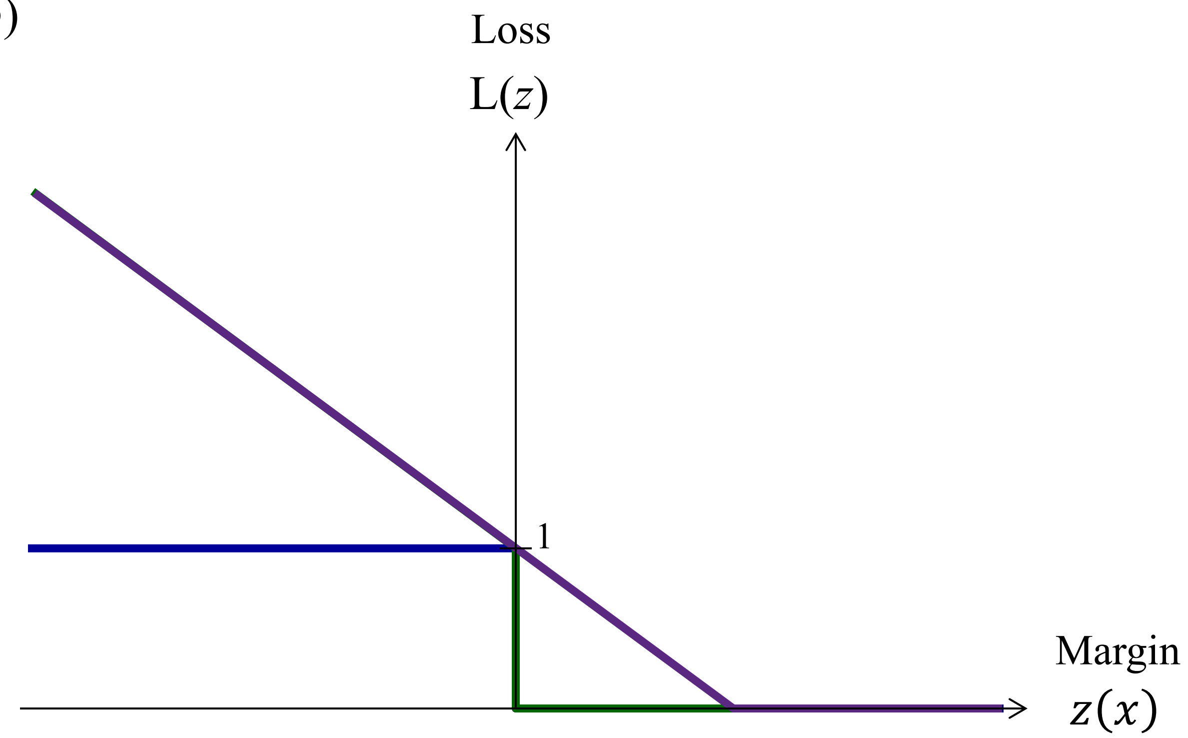
  - Positive if the estimate $\hat{s}(x)$ is correct
  - Negative if $\hat{s}(x)$ is incorrect
    - Since $\hat{s} > 0$ indicates positive estimate and $\hat{s} < 0$ negative
  - Large positive margins mean the classifier is "strongly correct"
  - Large negative margins are bad – they mean the classifier screwed up!
- In learning a classifier, we'd like to penalize *negative* margins by the use of a loss function **L(*z*)** that maps the margin to an associated loss

$$\mathrm{L} : \mathbb{R} \rightarrow [0, \infty)$$

The loss function, L(z)

What should the loss function look like?

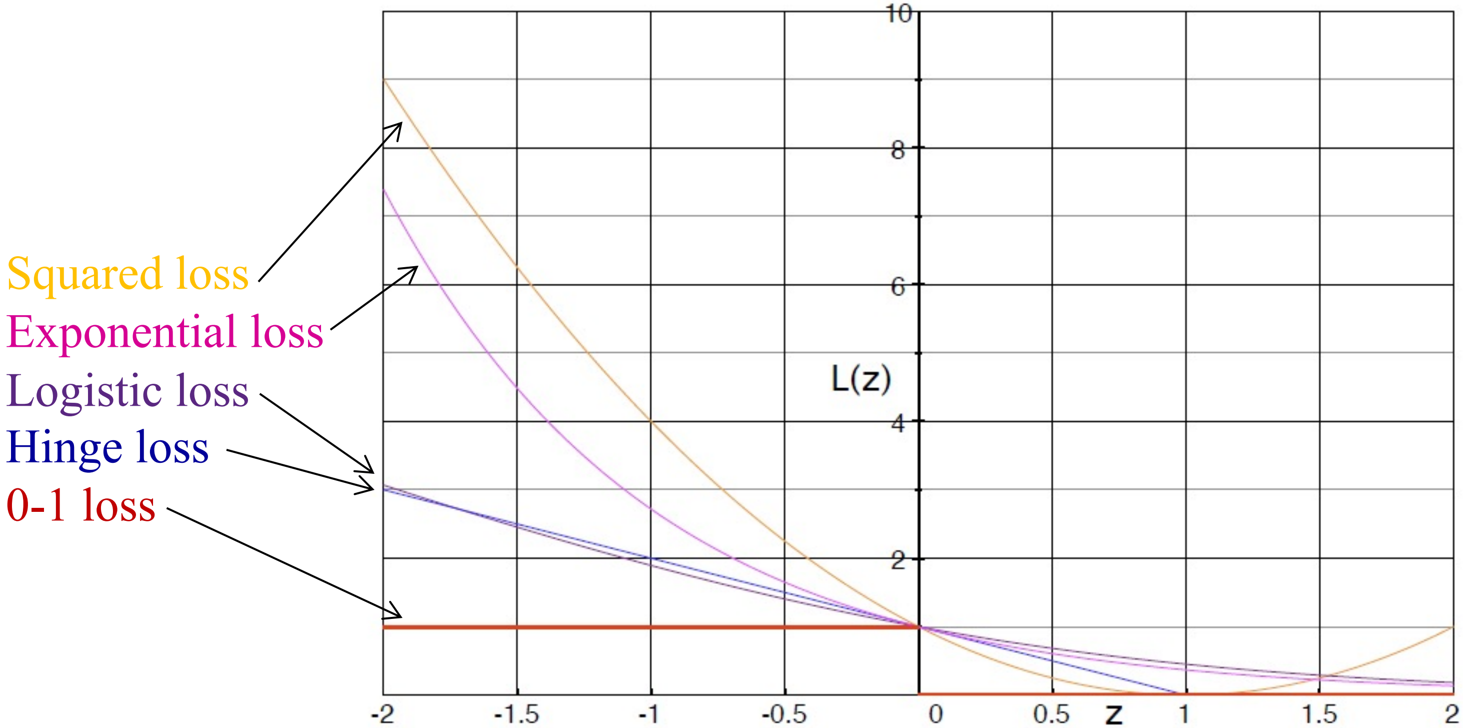
$\mathrm{L}: \mathbb{R} \rightarrow [0, \infty)$



Penalize wrong classifications more

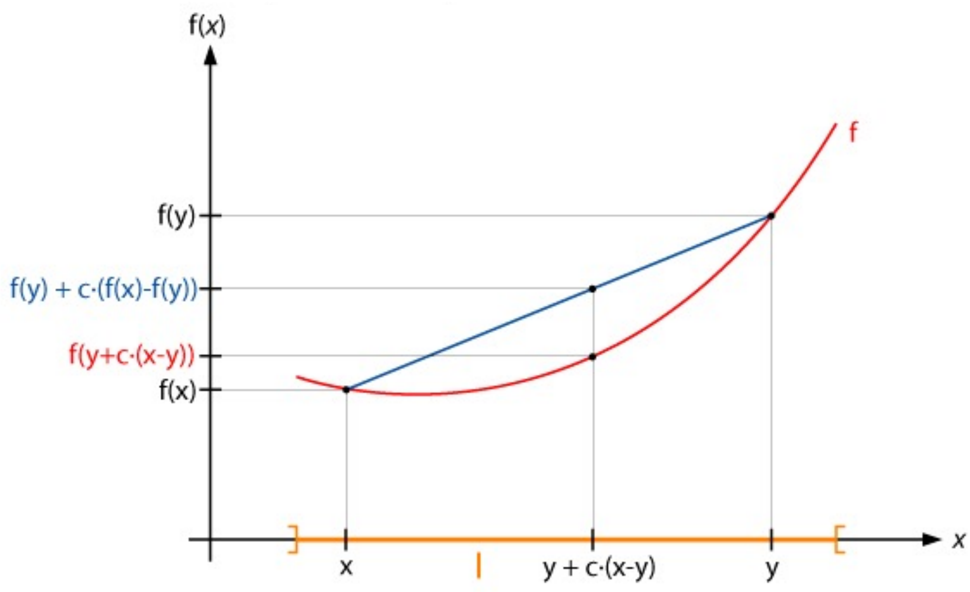
The loss function, L(z)

- Characteristics of the loss function:
  - For an example on the decision boundary, L(0) = 1
  - $L(z) \geq 1$ for $z < 0$
  - $0 \leq L(z) < 1$ for $z > 0$



The loss function, L(z)

- Loss functions are often used in optimization problems (to minimize a function) that lead to modifying weights in training
  - Typically it is squared – thus the mapping to $[0, \infty)$
- To help make this solvable, the loss function is often chosen to be convex, since optimizing a convex function is computationally more tractable



A convex function lies below the line connecting any two points on the function

Typical predictive machine learning scenarios

Task	*Label space*	*Output space*	*Learning problem*
Classification	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathscr{C}$	learn an approximation $\hat{c} : \mathscr{X} \rightarrow \mathscr{C}$ to the true labelling function c
Scoring and ranking	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathbb{R}^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a score vector over classes
Probability estimation	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = [0,1]^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a probability vector over classes
Regression	$\mathscr{L} = \mathbb{R}$	$\mathscr{Y} = \mathbb{R}$	learn an approximation $\hat{f} : \mathscr{X} \rightarrow \mathbb{R}$ to the true labelling function f

Ranking classifier

- The scores from a scoring classifier may not be particularly meaningful – they are not derived from any "true" scores – so it may be preferable to ignore the magnitude and just keep the **order** of the scores on a set of instances
  - This is less sensitive to outliers – i.e., more robust to noise/errors
- All positive examples should (ideally) be ranked higher than all negative examples
  - Exceptions to this are ranking errors
  - Count the ranking errors (*err*): For all (pos, neg) example pairs, how many rank neg higher than pos?
    - Ties count ½

Ranking error rate: *rank-err* = $^{err}/_{PN}$

Ranking accuracy: *rank-acc* = 1 – *rank-err*

Ranking classifier performance

Score: Low  High

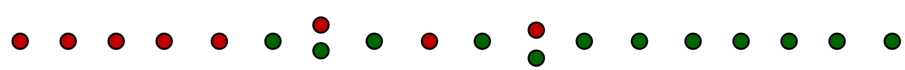
# of ranking errors? Ranking error rate? Ranking accuracy?

2 2/(5)(5) = 0.08 0.92

Low  High

# of ranking errors? Ranking error rate? Ranking accuracy?

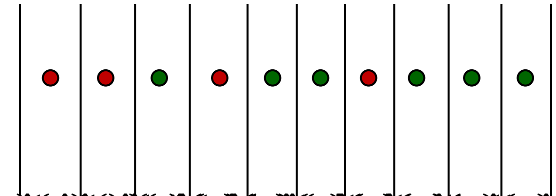
9 9/(8)(12) = 0.09375 0.90625

Quiz Low  High

# of ranking errors? Ranking error rate? Ranking accuracy?

9 9/(8)(12) = 0.09375 0.90625

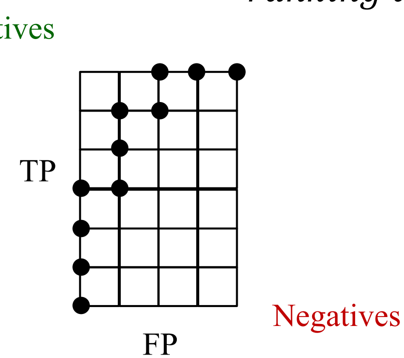
Ranking classifier and the coverage curve

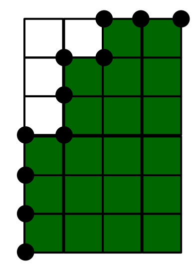
Low  High

Move the decision line and count:

FP = ? TP = ?

$$ranking\ acc = \frac{area\ under\ curve}{total\ area} = {}^{20}/_{24} = 0.83$$





This is the coverage curve

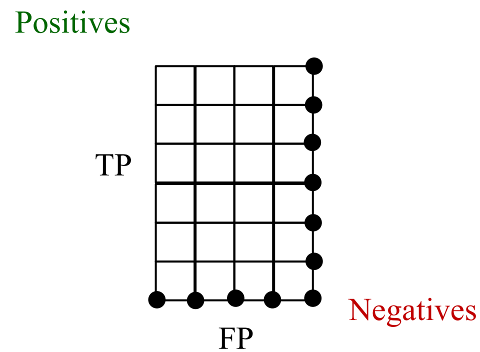
If we normalize to a square graph, we get the ROC curve

The Area Under the Curve (AUC) is the ranking accuracy

Ranking classifier and the coverage curve

Low  High

What about this case? It appears that the ranking is terrible!



What is the ranking accuracy? Zero (0%)

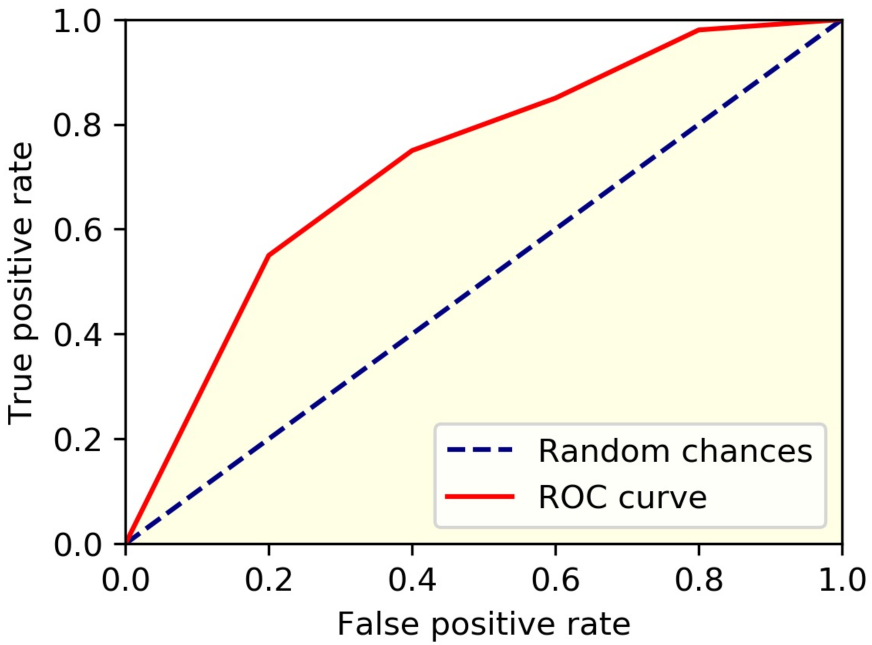
Q: What would the ranking accuracy be of this ranking?

Low  High

One (100%)

Classifier design – operating point

- You, as a classifier designer, can often move decision boundaries (modify thresholds) to make the false positive rate as high or as low as you wish
  - A very high threshold (don't let anything through!) results in no false positives – but lots of false negatives
  - A very low threshold (let everything through!) results in no false negatives – but lots of false positives

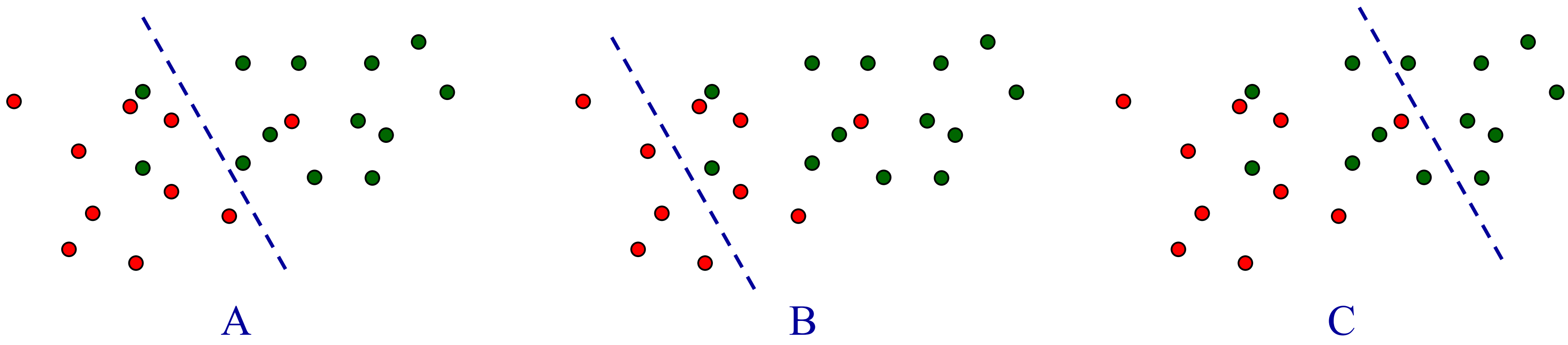


- This doesn't necessarily make the classifier better or worse – it just changes the operating point of the classifier

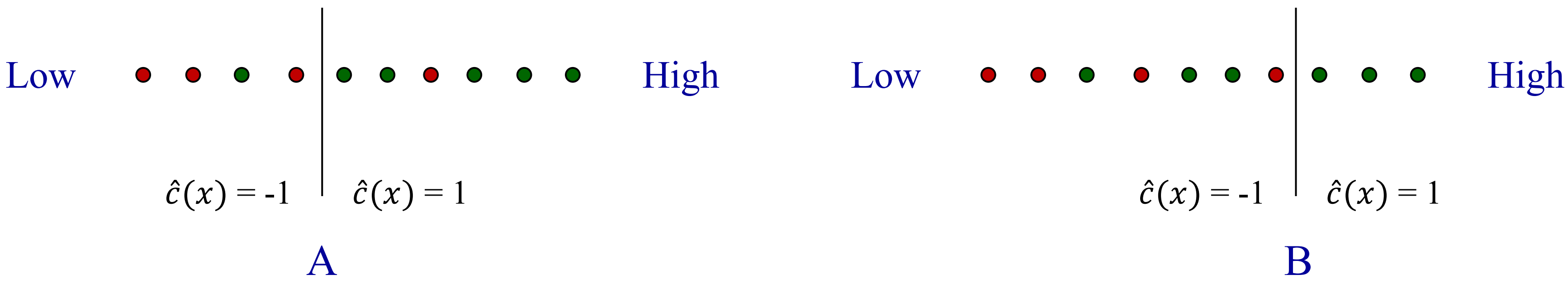
- This is often application-specific:
  - When might false positives be especially undesirable?
  - When might false negatives be especially undesirable?
  - We can encode these preferences in a cost function to compute an optimal threshold, given this information

Classifier design

Which classifier is best: A, B, or C?



Which is better: A or B?



<u>It depends</u> on what you want to optimize:

– TPR, FPR, error rate, accuracy, precision, …

Typical predictive machine learning scenarios

Task	*Label space*	*Output space*	*Learning problem*
Classification	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathscr{C}$	learn an approximation $\hat{c} : \mathscr{X} \to \mathscr{C}$ to the true labelling function c
Scoring and ranking	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathbb{R}^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a score vector over classes
Probability estimation	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = [0,1]^{\lvert\mathscr{C}\rvert}$	learn a model that outputs a probability vector over classes
Regression	$\mathscr{L} = \mathbb{R}$	$\mathscr{Y} = \mathbb{R}$	learn an approximation $\hat{f} : \mathscr{X} \to \mathbb{R}$ to the true labelling function f