Machine Learning

CSE 142

Xin (Eric) Wang

Monday, October 4, 2021

Today

- Classification, Ch. 2

Notes

- HW#1 posted and due by October 18, 23:59pm PT
  - Done individually
  - Re-read the *Policy on Academic Integrity* on Canvas
  - No extensions (but you can wisely use your four late days)
  - Justify every answer you give – **show the work** that achieves the answer or **explain** your response
  - Submit through Gradescope

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	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = \mathbb{R}^{\vert\mathscr{C}\vert}$	learn a model that outputs a score vector over classes
Probability estimation	$\mathscr{L} = \mathscr{C}$	$\mathscr{Y} = [0,1]^{\vert\mathscr{C}\vert}$	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

Classification

Set of possible classes

Instance

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

Classification

- A **classifier** is a mapping

$$\hat{c} : \mathcal{X} \rightarrow \mathcal{C}$$

  - where $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ is a (usually small) set of class labels
  - I.e., a labeling function $\hat{c}(x)$ that maps instances to classes

- $\hat{c}(x)$ is an estimate of the (presumably) true but unknown labeling function $c(x)$ (a.k.a. the *class function* or the *oracle*)

- The training data comprises labeled instances: $\{\ (x_i, l(x_i))\ \}$

- Ideally, $l(x_i) = c(x_i)$ (accurate training data), but not always!

- **Binary classification**: only two classes, ***positive*** and ***negative***.
  - E.g., ham or spam?

Contingency tables

We can summarize performance of a model on a binary classification task with a contingency table

		Actual class C: 1	Actual class C: 0	
Predicted class $\hat{C}$	1	TP	FP Type I Error	*Estimated positive* $\hat{P}$
	0	FN Type II Error	TN	*Estimated negative* $\hat{N}$
		Positives P	*Negatives* N	TOTAL

TP – true positives

FP – false positives

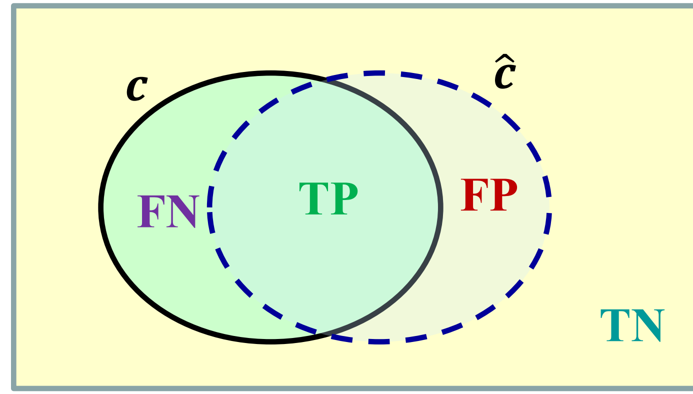
TN – true negatives

FN – false negatives

Correct: TP, TN

Errors: FP, FN

Instance space (all emails)



Actual class c ← Ground truth

Classifier output → Predicted class $\hat{c}$

Predicted class $\hat{c}$ \ Actual class c	1	0	
1	TP	FP Type I Error	*Estimated positive* $\hat{P}$
0	FN Type II Error	TN	*Estimated negative* $\hat{N}$
	Positives P	*Negatives* N	TOTAL

$$TP + FN = P$$

$$FP + TN = N$$

$$TP + FP = \hat{P}$$

$$FN + TN = \hat{N}$$

$$P + N = TOTAL$$

$$\hat{P} + \hat{N} = TOTAL$$

Key terminology

False positive rate (FPR) $= \frac{FP}{N} = \alpha$

Accuracy $= \frac{TP+TN}{P+N} = \left(\frac{P}{P+N}\right) TPR + \left(\frac{N}{P+N}\right) TNR$

False negative (miss) rate (FNR) $= \frac{FN}{P} = \beta$

Error rate $= \frac{FP+FN}{P+N} = 1 - \text{Accuracy}$

True positive rate (TPR) $= \frac{TP}{P}$ = Sensitivity = **Recall** = 1 - β

Precision $= \frac{TP}{\hat{P}}$

True negative rate (TNR) $= \frac{TN}{N}$ = Specificity = 1 - α

F1 score $= \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{P + \hat{P}}$

Actual class C

Predicted class $\hat{C}$

	1	0	
1	TP	FP	*Estimated positive* $\hat{P}$
0	FN	TN	*Estimated negative* $\hat{N}$
	Positives P	*Negatives* N	TOTAL

Note

Note that I tend to draw contingency tables transposed from how the book does it

	Predicted ⊕	Predicted ⊖	
Actual ⊕	**30**	**20**	50
Actual ⊖	**10**	**40**	50
	40	60	100

← Book's contingency table

There's no standard, so always check to verify which axis is ***actual*** and which is ***predicted***

My contingency table →

Predicted class $\hat{c}$		Actual class c: 1	0	
	1	30	10	*40*
	0	20	40	*60*
		50	*50*	100

Quiz: Accuracy, Precision, Recall, and F-1

- What is the accuracy of this model?
  - Acc = $\frac{TP+TN}{P+N}$ = (5 + 100) / (9 + 105) = 92.1
  - What if the positive here refers to COVID-19 carriers?
- Precision:
  - Precision = $\frac{TP}{\hat{P}}$ = 5 / 10 = 50%
  - FP (Type I Error): ham emails misclassified as spams
- Recall:
  - Recall = $\frac{TP}{P}$ = 5 / 9 = 55.6%
  - FN (Type II Error): spams misclassified as hams
- Precision or Recall when positive is?
  - COVID-19 carrier
  - Google search results
- F1:
  - F1 score = $\frac{2 \cdot precision \cdot recall}{precision + recall}$
  - Balance between Precision & Recall

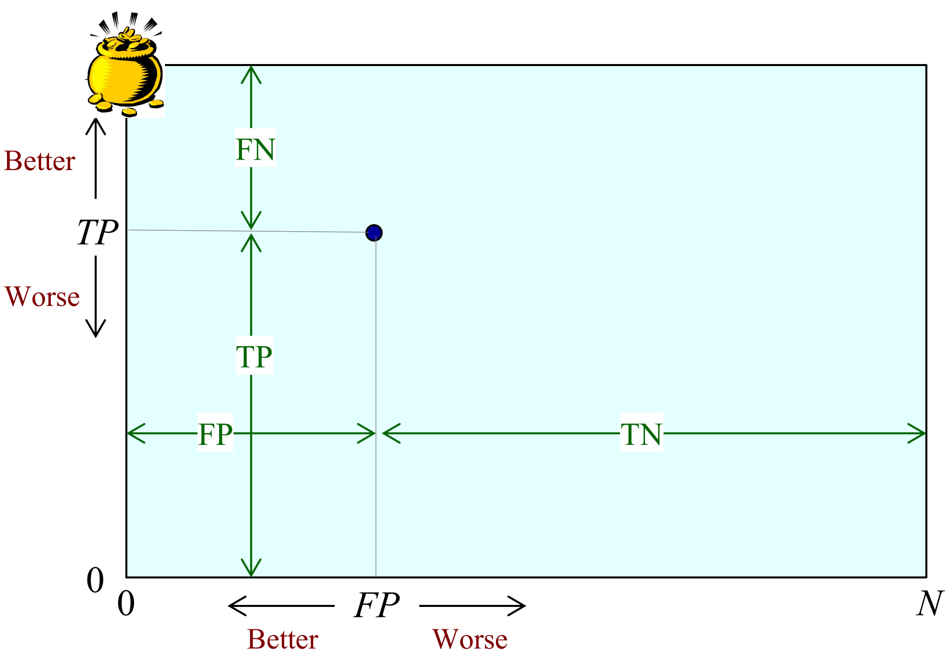
Ground truth c

Classifier output $\hat{c}$	spam	ham		
spam	5 (TP)	5 (FP)	*10 ($\hat{P}$)*	10 classified as spam
ham	4 (FN)	100 (TN)	*104 ($\hat{N}$)*	104 classifie d as ham
	9 (P)	*105 (N)*	114	

114 instances in the test dataset (9 spam, 105 ham)

Coverage plot

- It's very important to understand contingency tables and the values derived from them (false positive rate, accuracy, error rate, precision, etc.)
- The coverage plot provides a way to visualize classifier performance on test data: { TP, FP, P, N }
  - A contingency table becomes a single point in a coverage plot

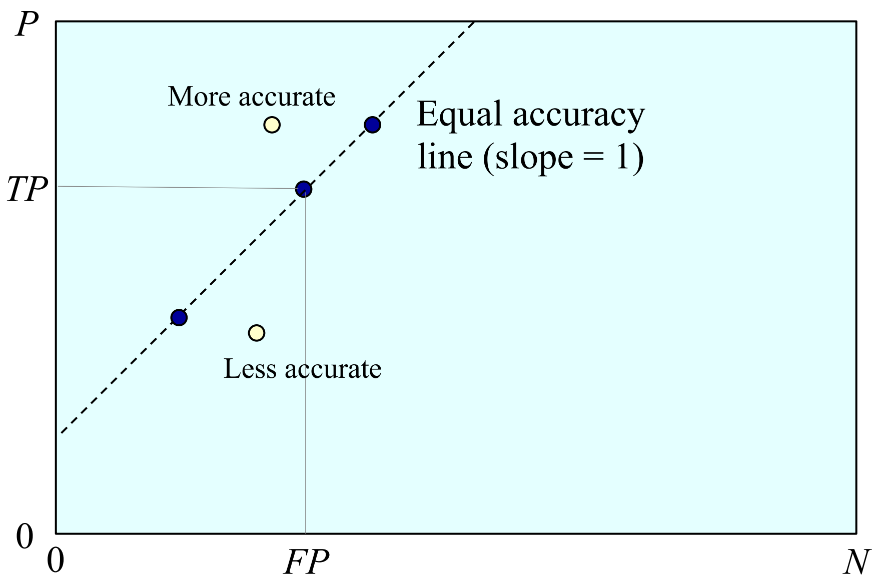


		c		
		1	0	
$\hat{c}$	1	TP	FP	*Estimated positive* $\hat{P}$
	0	FN	TN	*Estimated negative* $\hat{N}$
		Positives P	*Negatives* N	TOTAL

Coverage plot

- In a coverage plot, classifiers with the same accuracy are connected by line segments with slope 1

$$\text{Accuracy} = \frac{TP + TN}{P + N}$$

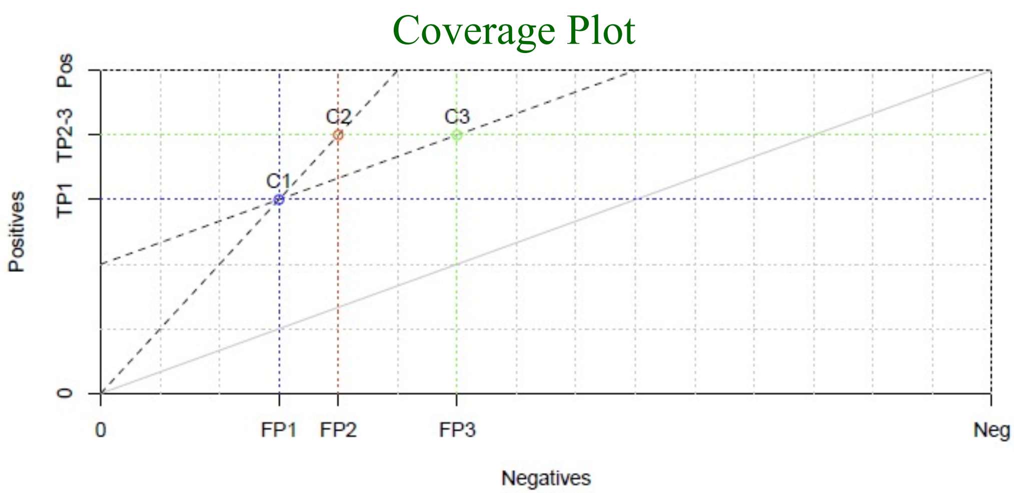


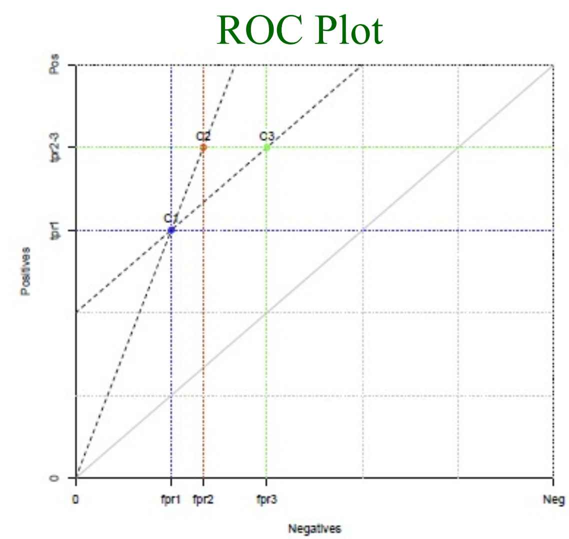
		c = 1	c = 0	
$\hat{c}$	1	TP	FP	*Estimated positive* $\hat{P}$
	0	FN	TN	*Estimated negative* $\hat{N}$
N		*Positives* P	*Negatives* N	TOTAL

ROC plot

True positive rate (TPR) = $\frac{TP}{P}$ = Sensitivity = **Recall**

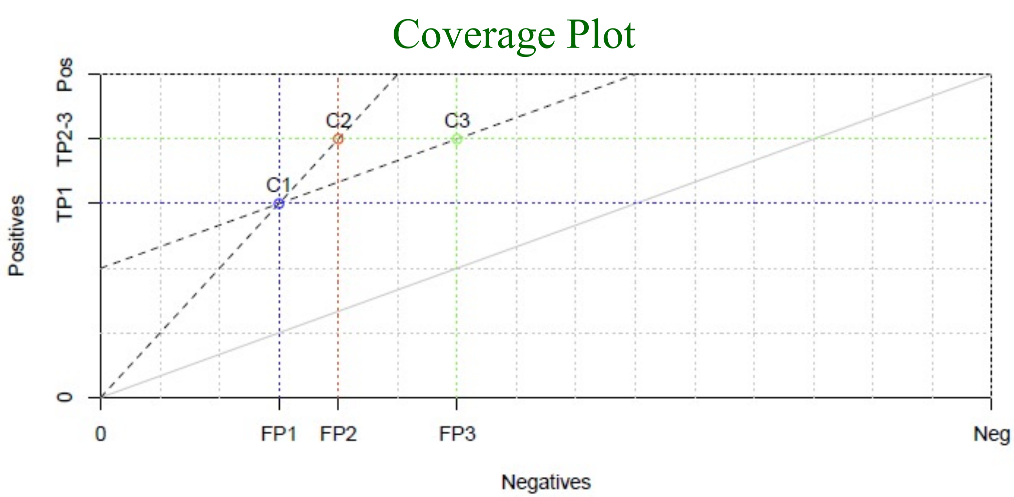
- If we normalize the coverage plot to a square, with each axis ranging from 0 to 1, we can plot TPR and FPR (instead of TP and FP)
- This gives us an ROC plot
  - ROC - "receiver operating characteristic"
    - Comes from signal detection theory

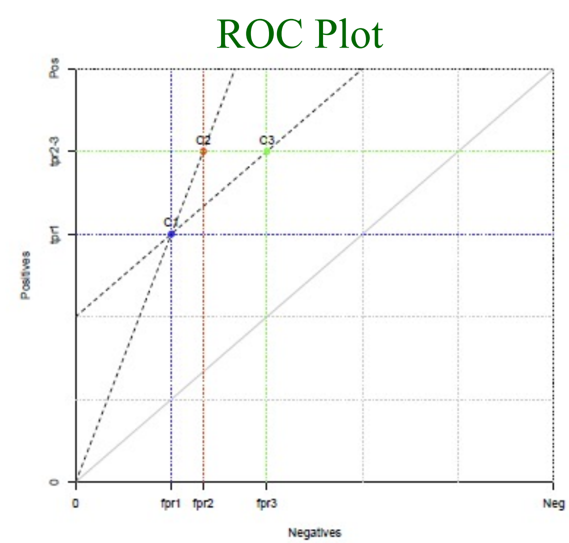




ROC plot

- In a ROC plot, classifiers with the same average recall are connected by line segments with slope 1
- Average recall = (TPR + TNR) / 2
  - Recall = TPR, negative recall = TNR
- Observations:
  - C1 and C2 have the same accuracy, C1 and C3 have the same average recall
  - C2 is higher on both





Quiz: Coverage plot vs. ROC plot

- When to use Coverage plot and ROC plot?
  - Canvas

- Coverage plot
  - Explicitly take the class distribution into account
  - Limitation: single dataset
- ROC plot
  - Different datasets with different class distributions
  - More common

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

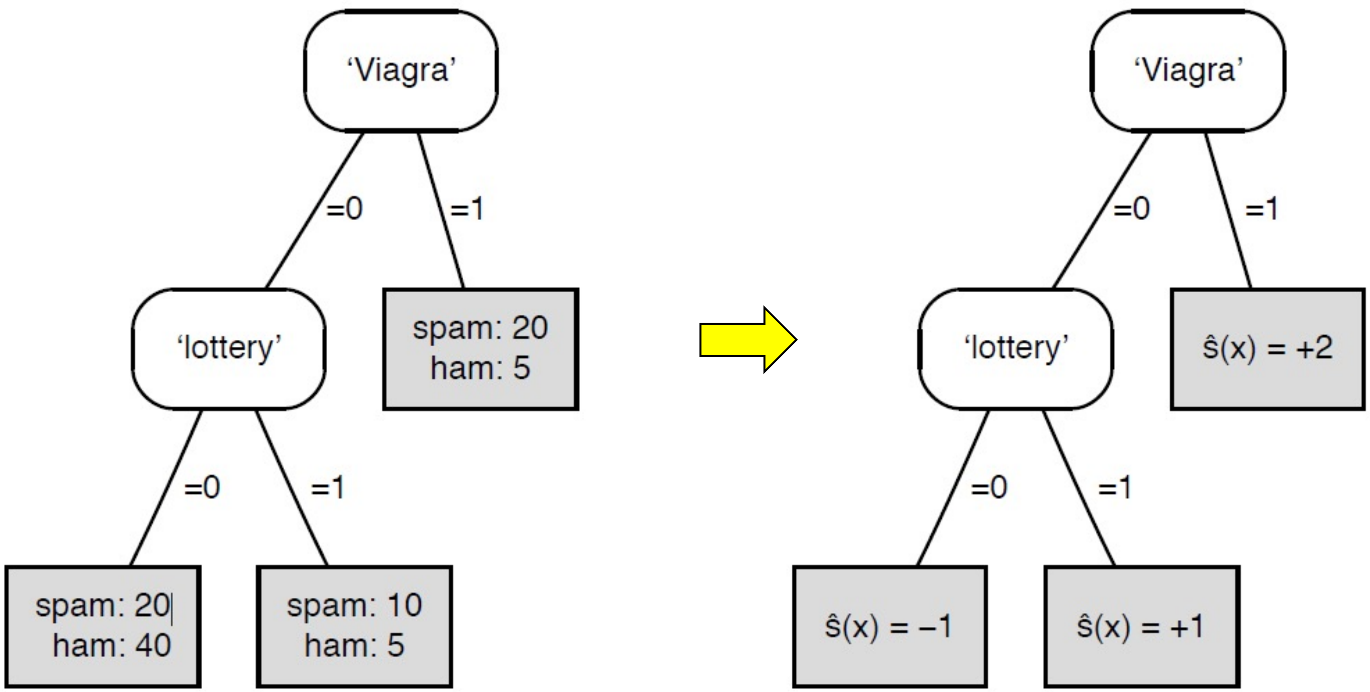
Scoring classifier

- Many classifiers produce scores (e.g., matching scores) on which their class predictions are based
  - E.g., with SpamAssassin – a score over 5.0 is classified as spam
- A scoring classifier is a mapping $\hat{\mathbf{s}} : \mathcal{X} \rightarrow \mathbb{R}^k$ along with a class decision based on the scores (typically *highest score*)
  - Given an instance *x*, output a (scalar) score for each of the *k* classes
    - Often just for one class in a binary classifier
  - Indicates how likely the class label $\mathcal{C}_i$ applies to *x*
  - This is not (in general) a probability – scores can be any scalars
- Typically the score is normalized to zero – i.e., $\hat{s} > 0$ indicates positive class, $\hat{s} < 0$ indicates negative class

Scoring classifier

We can turn the feature tree into a scoring tree by computing a score for each leaf

$$\hat{s}(x) = \log_2 \frac{\#spam}{\#ham}$$



Feature tree

Scoring tree

Classifier margin and loss function

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

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

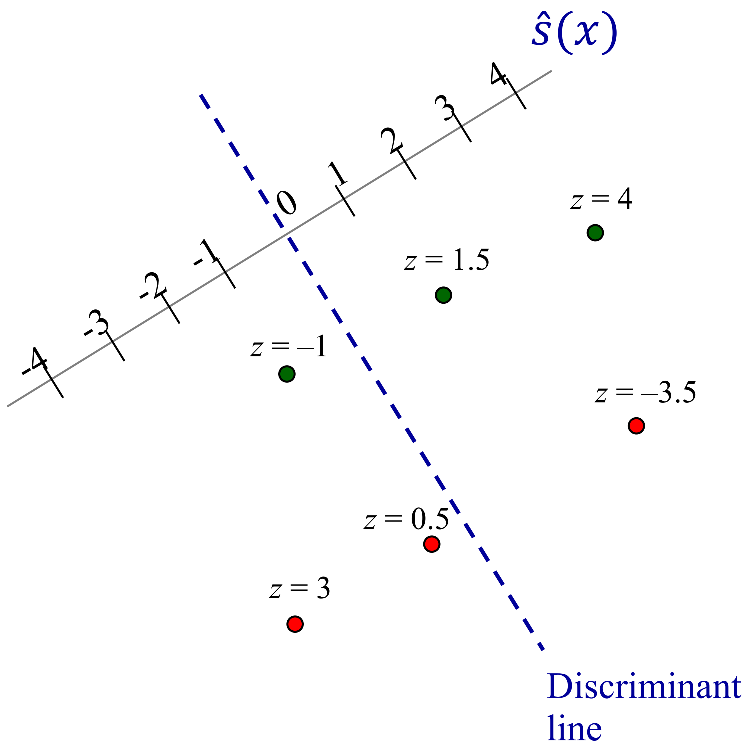
  - Positive if the estimate $\hat{s}(x)$ is correct
  - Negative if $\hat{s}(x)$ is incorrect
    - Since $\hat{s} > 0$ indicates positive estimate and $\hat{s} < 0$ negative
  - Large positive margins mean the classifier is "strongly correct"
  - Large negative margins are bad – they mean the classifier screwed up!

Classifier margin and loss function

Training data:

Positive class: $c = +1$

Negative class: $c = -1$



Score $\hat{s}(x)$

True class function $c(x)$

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

How should each training data point impact the classifier learned from this data?

The loss function $L(z)$ will determine this

At this point in the iterative classifier training algorithm, which training data points are the most important?

Classifier margin and loss function

- 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

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