Machine Learning

CS 165B

Prof. Matthew Turk

Wednesday, April 6, 2016

Today

- The ingredients of machine learning (Ch. 1)
- Classification (Ch. 2)

Notes

- HW#1 due on Friday at 4:30pm
  - Turn in via (1) homework box in HFH or (2) GauchoSpace
  - If turned in via GauchoSpace, must be typeset – NO pictures or scans!

- Problem 3:
  - LDL cholesterol levels are reported as numbers: e.g., 120 mg/dL

- Problem 5:
  - You fill in the missing value.

	class = 165B			**class** = basketweaving		
grade	**effort**=Small	Medium	Large	**effort**=Small	Medium	Large
A	0	0.025	???	0.05	0.1	0.15
B	0.025	0.04	0.06	0.05	0.05	0.025
C	0.025	0.05	0.025	0.05	0.025	0
D	0.05	0.02	0.005	0	0	0
F	0.05	0	0	0	0	0

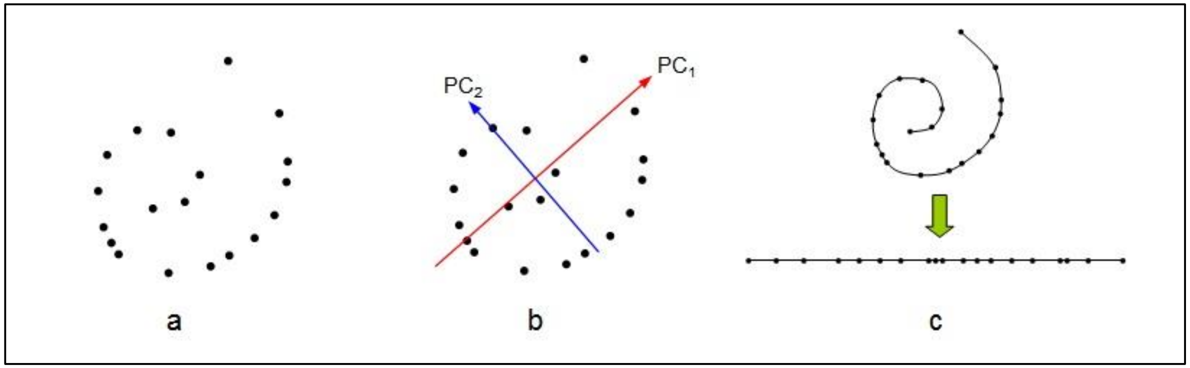
Dimensionality reduction

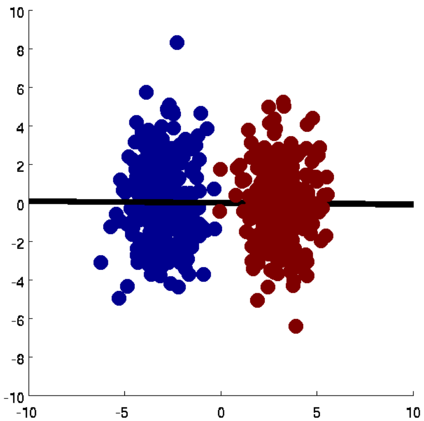
- Let's say we build a ML system to predict someone's occupation. The 12 features given to us are:
  – First name, last name, SSN, father's occupation, mother's occupation, highest educational degree, last year's salary, federal taxes paid last year, home address, model of car, miles driven last year, miles flown last year
- Some of these features may be useless, with no relevant information
- There may be redundancies – correlations among features
- Can we transform this 12-dimensional classification problem to a lower-dimensional problem?
  – Perhaps easier, computationally simpler…
- Yes, through dimensionality reduction

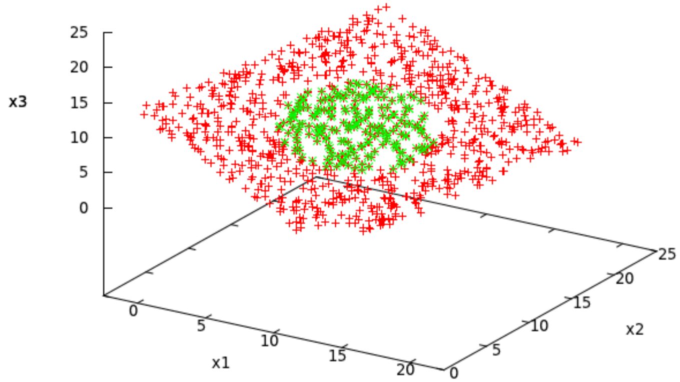
Intrinsic dimensionality

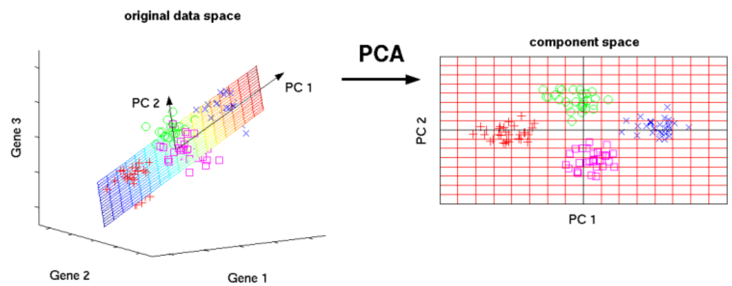
- The intrinsic dimensionality of (N-dimensional) data describes the real structure of the data, embedded in N-space
  - I.e., how many variables are needed to (minimally) represent the data?
- N-dimensional data could be intrinsically:
  - 0 dimensional – tightly clustered around a point
  - 1 dimensional – defined by a line or contour
  - 2-dimensional – defined by a plane or 2D manifold
  - M-dimensional ($M \leq N$) – defined by an M-dimensional hyperplane or M-dimensional manifold

Manifold – locally M-dimensional surface



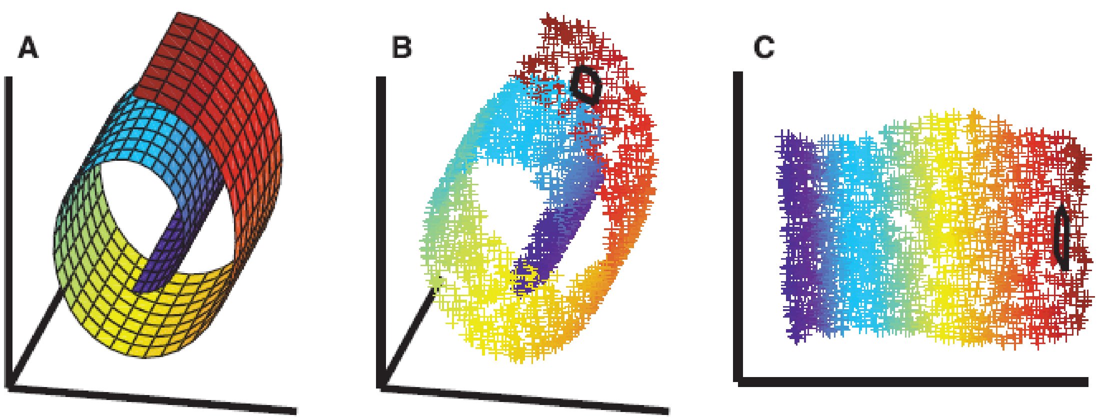


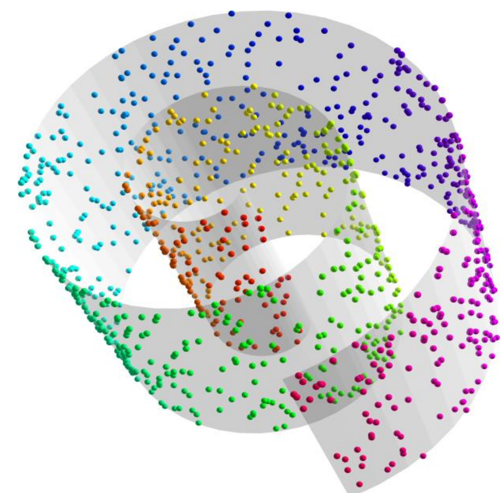




There are many dimensionality reduction methods:

- PCA
- ICA
- LDA
- LLE
- Factor analysis
- Random methods
- etc….





Summary so far…

- Key machine learning concepts
  - Core ML problem formulation
  - Important types of ML problems
  - Data sets (training, validation, test)
  - Linear classification
  - Models: generalization and overfitting
  - Models: geometric, probabilistic, logical
  - Distance measures
  - Tasks: predictive and descriptive
  - Features
  - The curse of dimensionality; intrinsic dimensionality
- Next:
  - Classification
    - Formulation, assessment, methods

Classification

Chapters 2 and 3 in the textbook

You be the classifier – learn the concept

Training data:

Engaged	Yesterday
Dedicated	Running
Devotion	Play
Work	Giraffe
Ground	Coupon
Live	Russia
Fathers	Coffee
Advanced	Ceramic

Verification data:

Party	Leisure
Restaurant	Power
Equal	Resting
Kitten	Eating
Proposition	Nation
Great	Minus
Computer	Kiss
Court	Field
Honored	Deltopia

Form a hypothesis (model, function)…

Let's test it on a verification data set

You be the classifier – learn the concept

Training data:

Engaged	Yesterday
Dedicated	Running
Devotion	Play
Work	Giraffe
Ground	Coupon
Live	Russia
Fathers	Coffee
Advanced	Ceramic
Score	Wedding
Four	Job
Seven	Sky
Continent	Phoenix
Nation	City
War	Peace
Consecrate	Marrow

Verification data:

Party	Leisure
Restaurant	Power
Equal	Resting
Kitten	Eating
Proposition	Nation
Great	Minus
Computer	Kiss
Court	Field
Honored	Deltopia

Try again with more training data…

Now we're finished training…!

You be the classifier – learn the concept

Training data:

Engaged	Yesterday
Dedicated	Running
Devotion	Play
Work	Giraffe
Ground	Coupon
Live	Russia
Fathers	Coffee
Advanced	Ceramic
Score	Wedding
Four	Job
Seven	Sky
Continent	Phoenix
Nation	City
War	Peace
Consecrate	Marrow

Test data:

Grief	Cause
Years	Camera
Forth	Pool
Endure	Freedom
California	Flashlight
Random	History
Battle	Nobly
Esteemed	Birth
Strike	People
Union	Newspaper
Resting	Perish
Education	Saga
Dead	Money
Earth	Moon

You be the classifier – learn the concept

Training data:

Engaged	Yesterday
Dedicated	Running
Devotion	Play
Work	Giraffe
Ground	Coupon
Live	Russia
Fathers	Coffee
Advanced	Ceramic
Score	Wedding
Four	Job
Seven	Sky
Continent	Phoenix
Nation	City
War	Peace
Consecrate	Marrow

Test data:

Grief	Cause
Years	Camera
Forth	Pool
Endure	Freedom
California	Flashlight
Random	History
Battle	Nobly
Esteemed	Birth
Strike	People
Union	Newspaper
Resting	Perish
Education	Saga
Dead	Money
Earth	Moon

You be the classifier – learn the concept

A few observations/questions:

- What if the word "iPad" were accidentally included in the training positives? If the word "ago" were accidentally included in the training negatives?
  - Machine learning has to allow for the possibility of noisy training data
    - In most cases, the *certainty* of noisy training data (errors)
- What if you had never heard of the Gettysburg Address?
  - This task assumed that the learner (you) had certain semantic knowledge at your disposal – this knowledge allowed you to extract meaningful features from the data (the word instances)
  - Acquiring such semantic knowledge is a very hard problem!
  - Without it, no classifier could do well on this task
- What is the *chance* (random) *performance* in this task?

Some key terms in classification

- Task, model, features, instances
- Feature space, instance space, label space, output space
- Training set of labeled instances
- Instance noise, label noise
- Labeling function

- Set theory (discrete math) terms
  - Set, null set, power set
  - Intersection, union, difference, complement, cardinality
  - Cartesian product
  - Set relations
  - Properties of relations (reflexive, symmetric, antisymmetric, transitive, total
  - Equivalence relation, equivalence class, partition

Typical predictive machine learning scenarios

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
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

Testing/assessing a classifier's performance

- E.g., for a spam/ham binary classification problem
- Performance on the test set:
  - # of correctly classified emails (true positives, true negatives)
  - # of incorrectly classified emails (false positives, false negatives)

Ground truth

c

Classifier output $\hat{c}$

	spam	ham		
spam	9	5	*14*	14 classified as spam
ham	4	10	*14*	14 classified as ham
	13	*15*	28	

28 instances in the test dataset (13 spam, 15 ham)

Contingency tables

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

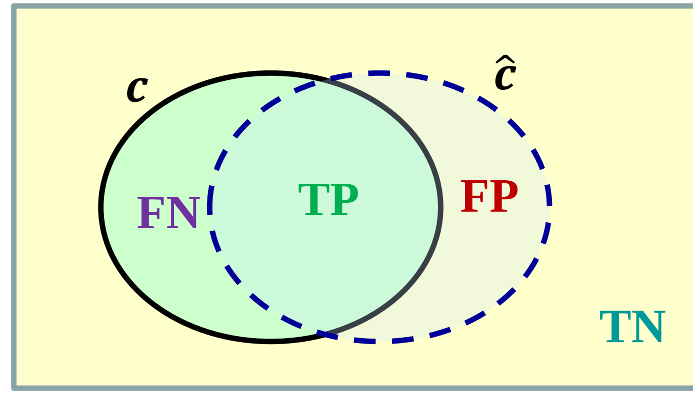
Actual class c

Predicted class $\hat{c}$		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 – true positives
FP – false positives
TN – true negatives
FN – false negatives

Correct
Errors

Instance space (all emails)



Actual class
c

		1	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

Key terminology

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

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

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

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

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

Error rate $= \frac{FP+FN}{P+N}$

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

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 →

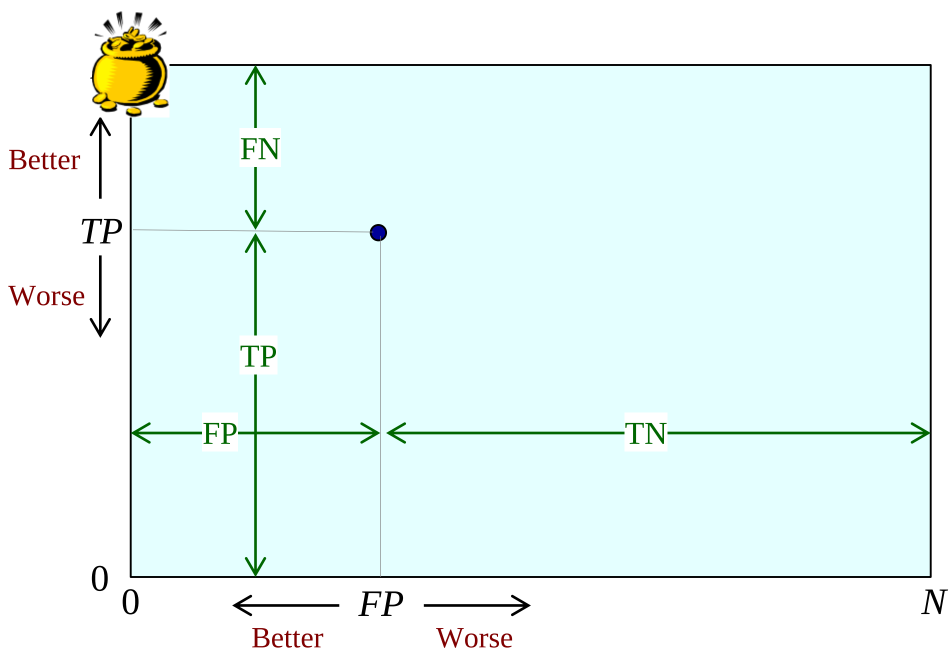
Actual class c

Predicted class $\hat{c}$

	1	0	
1	30	10	*40*
0	20	40	*60*
	50	*50*	100

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: { P, N, TP, FP }
  - A contingency table becomes a single point in a coverage plot

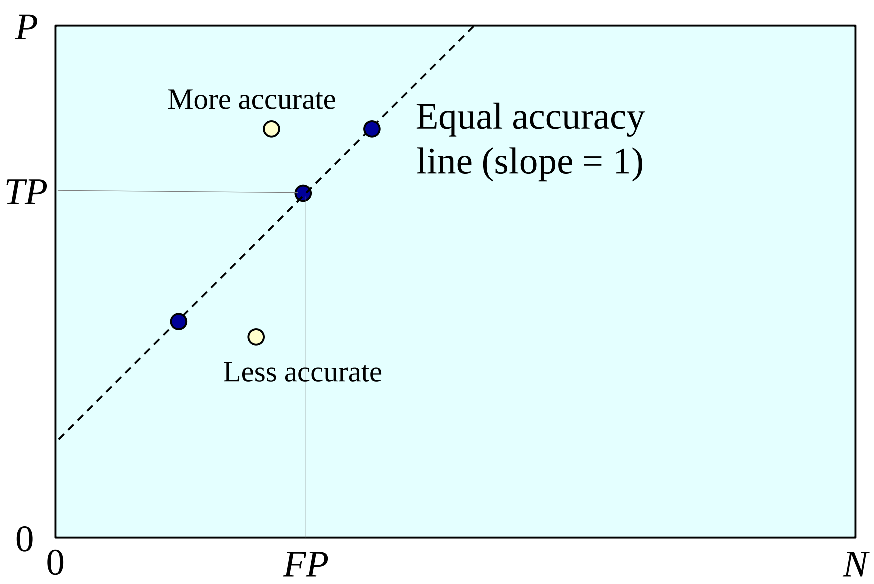


		c		
		1	0	
$\hat{c}$	1	TP	FP	*Estimated positive* $\hat{P}$
	0	FN	TN	*Estimated negative* $\hat{N}$
		Positives *P*	*Negatives* *N*	TOTAL

Coverage plot

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

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

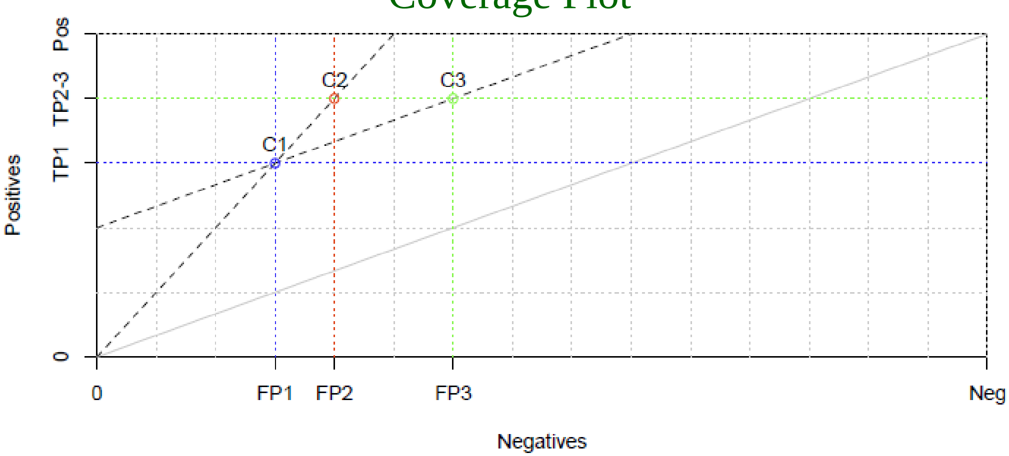


		c		
		1	0	
$\hat{c}$	1	TP	FP	*Estimated positive $\hat{P}$*
	0	FN	TN	*Estimated negative $\hat{N}$*
N		*Positives P*	*Negatives N*	TOTAL

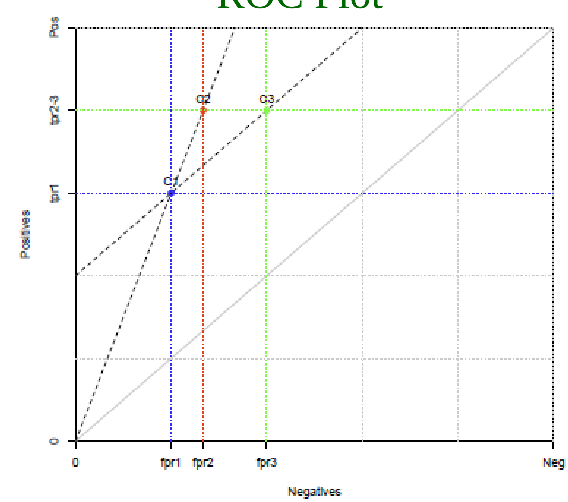
ROC plot

- 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
  - In an ROC plot, line segments with slope 1 connect classifiers with the same average recall

Coverage Plot



ROC Plot



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

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 function $\hat{c}(x)$ that maps instances to classes

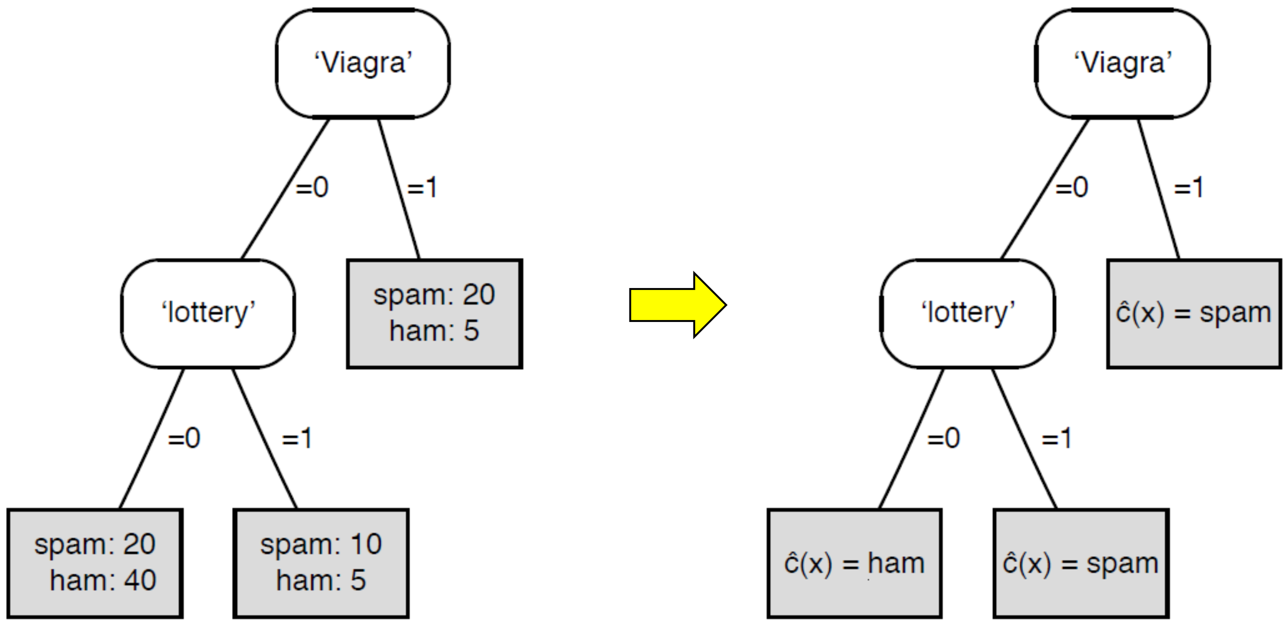
$\hat{c}(x)$ is an estimate of the (presumably) true but unknown function $c(x)$ (a.k.a. the *oracle*)

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 and concept learning

- Two-class classification (k = 2, binary classification) is known as concept learning
  - Learning to distinguish a concept ***c*** from all else – true or false
  - For example, learning the concept of *spam*:



Feature tree

Decision tree using majority class decision rule

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

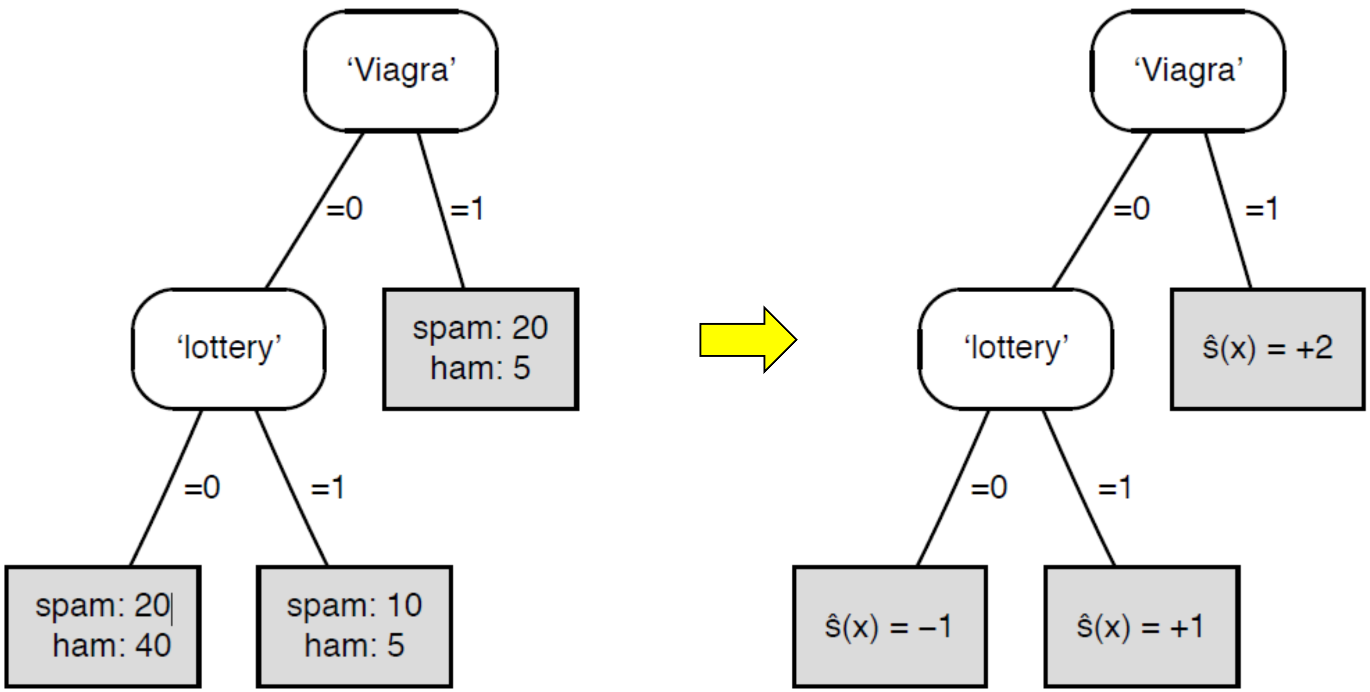
Scoring classifier

- 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 $\hat{s} : \mathcal{X} \to \mathbb{R}^k$ 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 does the class label $\mathcal{C}_i$ apply to *x* ?
  - This is not (in general) a probability – scores can be any scalars
- Typically the score is normalize 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

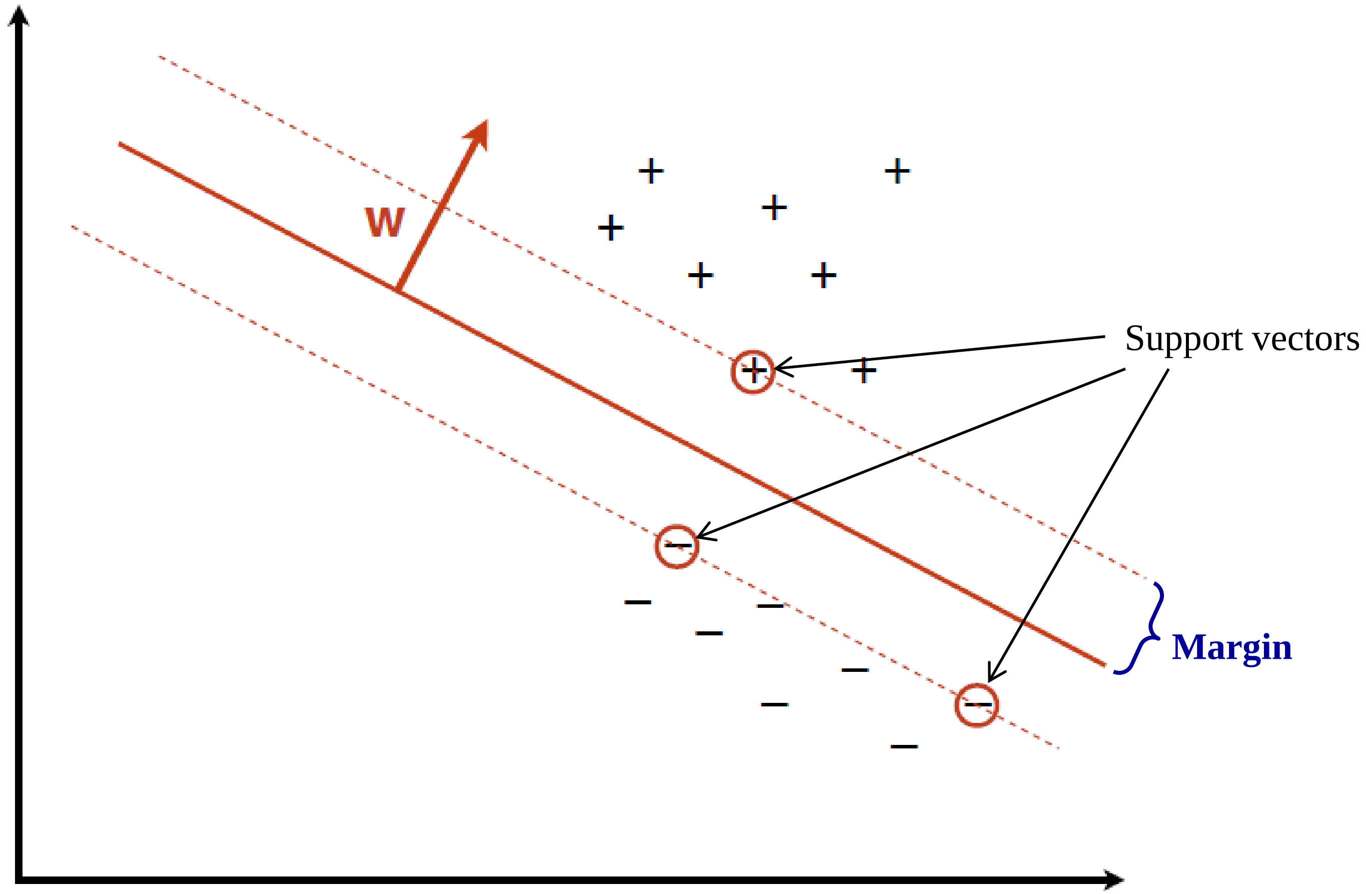
- True class function $c(x) = \begin{cases} +1 \text{ for positive examples} \\ -1 \text{ for negative examples} \end{cases}$
- The scoring classifier assigns a margin *z*(*x*) to each instance *x*:

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

  - 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!

Support Vector Machine (SVM) classifier

SVM learns the optimal decision boundary from linearly separable data, maximizing the ***margin***



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!
- In learning a classifier, we'd like to reward large *positive* margins and penalize large *negative* margins by the use of a loss function **L(z)** that maps the margin to an associated loss

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