

# CS 189 : Introduction to Machine Learning

Isabelle Guyon

Alexei Efros

Fall 2015

UC Berkeley

# Course Instructors

Prof. Isabelle Guyon



Prof. Alexei Efros



# Course Instructors

## Prof. Isabelle Guyon

Isabelle Guyon is an independent consultant, specialized in statistical data analysis, pattern recognition and machine learning. Her areas of expertise include computer vision and and bioinformatics. Her recent interest is in applications of machine learning to the discovery of causal relationships. Prior to starting her consulting practice in 1996, Isabelle Guyon was a researcher at AT&T Bell Laboratories, where she pioneered applications of neural networks to pen computer interfaces and co-invented Support Vector Machines (SVM), a machine learning technique, which has become a textbook method. She is also the primary inventor of SVM-RFE, a variable selection technique based on SVM. The SVM-RFE paper has thousands of citations and is often used as a reference method against which new feature selection methods are benchmarked. She also authored a seminal paper on feature selection that received thousands of citations. She organized many challenges in Machine Learning over the past few years supported by the EU network Pascal2, NSF, and DARPA, with prizes sponsored by Microsoft, Google, and Texas Instrument. Isabelle Guyon holds a Ph.D. degree in Physical Sciences of the University Pierre and Marie Curie, Paris, France. She is president of Chalearn, a non-profit dedicated to organizing challenges, vice-president of the Unipen foundation, adjunct professor at New-York University, action editor of the Journal of Machine Learning Research, and editor of the Challenges in Machine Learning book series of Microtome.

## Prof. Alexei Efros

Alexei (Alyosha) Efros joined UC Berkeley in 2013 as associate professor of Electrical Engineering and Computer Science. Prior to that, he was nine years on the faculty of Carnegie Mellon University, and has also been affiliated with École Normale Supérieure/INRIA and University of Oxford. His research is in the area of computer vision and computer graphics, especially at the intersection of the two. He is particularly interested in using data-driven techniques to tackle problems which are very hard to model parametrically but where large quantities of data are readily available. Alyosha received his PhD in 2003 from UC Berkeley.

# GSI Team

- Nihar Shah (Head GSI)
- Brian Chu
- Weicheng Kuo
- Deepak Pathak
- Shaun Singh
- Faraz Tavakoli

# Resources

<http://www-inst.eecs.berkeley.edu/cs189>

- bCourses (+ piazza, kaggle), office hours, syllabus, assignments, readings, lecture slides, announcements, etc.

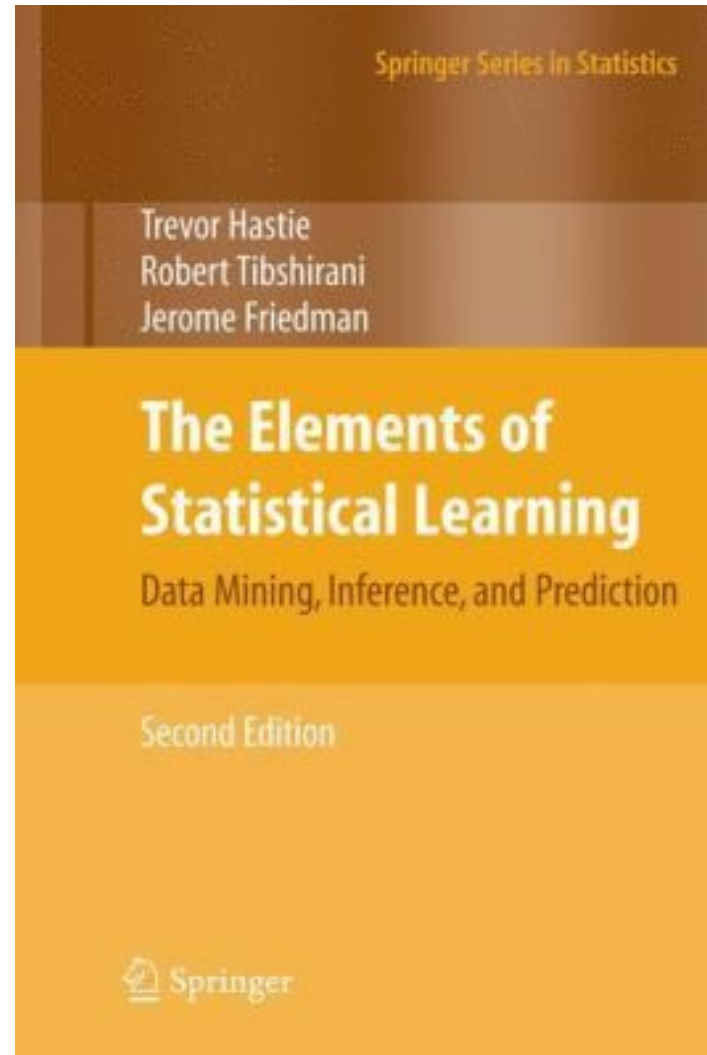
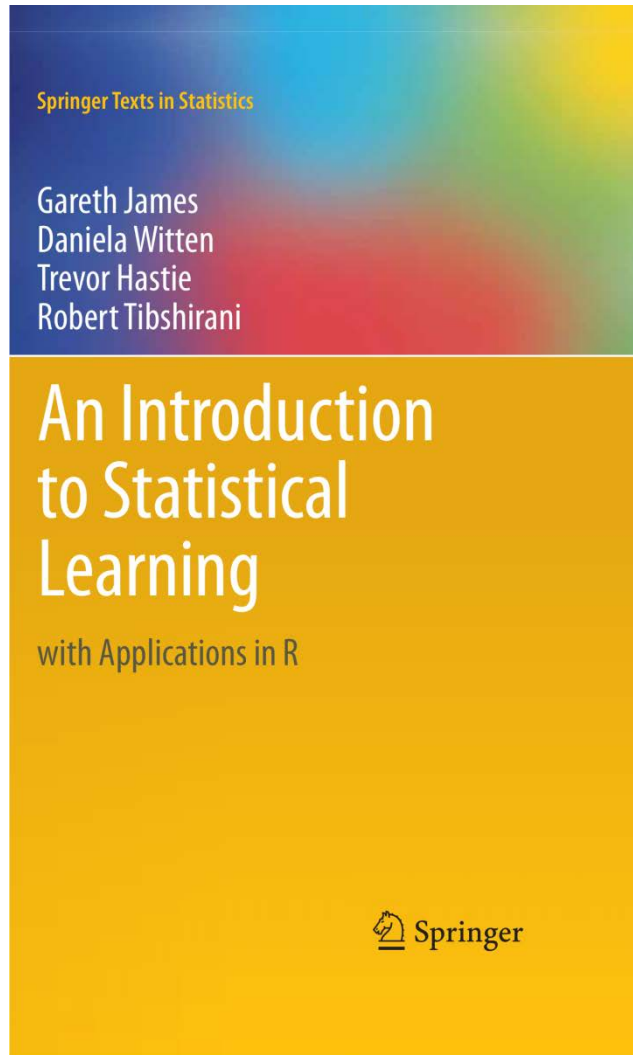
# Organizational Issues

- Homework:
  - Discussion of homework problems with other students is encouraged.
  - All homework must be written individually (including programming components).
  - NEW! We will have homework parties (TBA)
  - **Late policy:** 5 emergency slip days per semester. After that, no credit.
  - **Midterm:** Oct 27, in class

# Organizational Issues

- During lecture:
  - No food
  - no screens (to see why, google “[laptops in class](#)”)
- For questions:
  - Use Piazza (public or private)
  - For organizational questions/issues: talk to your GSI or Head GSI. If still unresolved, talk to instructors.

# Textbooks



Plus other materials, as needed



# Situating this course

- DS10: Intro to Data Science (*future offering*)
- Data Science (CS194-16): Intro to Data Science
- Math/Stats preparation: Math53 (vector calculus); Math54 (linear algebra); CS70 (discrete math, probability)
- CS188 Intro to AI
- **This course**
- Advanced Machine Learning (CS281A/B)
- Domain-Specific ML
  - Computer Vision (CS280)
  - statistical Natural Language Processing (CS288)
  - Learning in Robotics (e.g. CS287)
- various Stats/EE/Math courses, e.g. optimization

# Managing expectations: bear with us

- This class started as a CS194 special topics course with 20 people
  - Exponential growth: 100, 200, 400, 400x2
- We are doing our best to keep up, but don't expect CS188-like smoothness!
- **OK, let's go!**

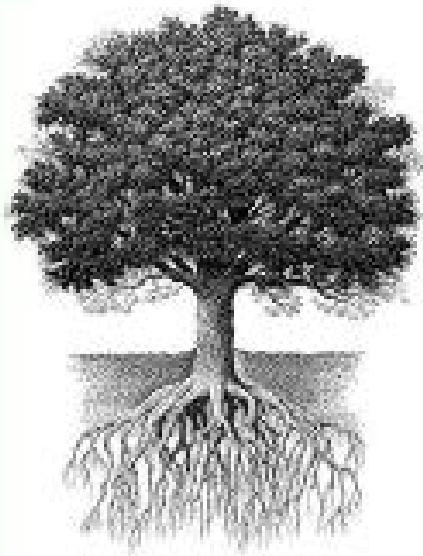
# Let's Define a Tree



*Brown trunk moving upward and branching with leaves*

# Are these trees?

Defining is Hard; Recognizing is Easy



- Hard to give a good definition of tree
- But any 3-year-old can tell a tree from a non-tree
- The 3-year-old has **learned from data!**

# “Unreasonable Effectiveness of Data”

[Halevy, Norvig, Pereira 2009]

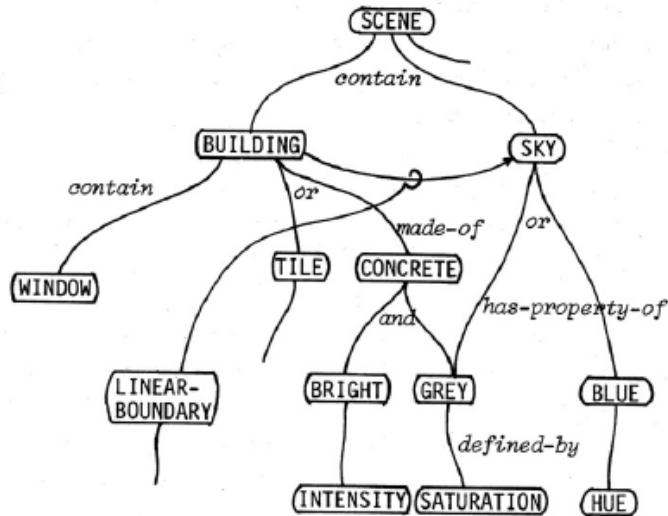
- Parts of our world can be explained by elegant mathematics:
  - physics, chemistry, astronomy, etc.
- But much cannot:
  - psychology, genetics, economics, etc.
- Enter: The Magic of **Data**



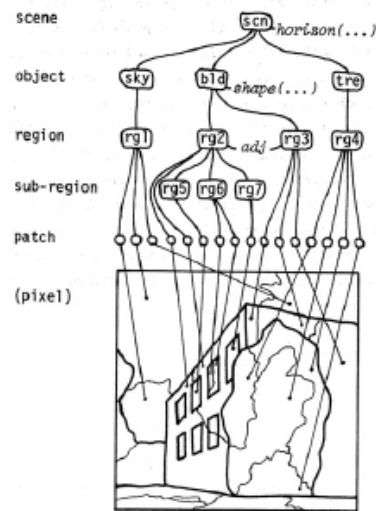
- A.I. for the postmodern world



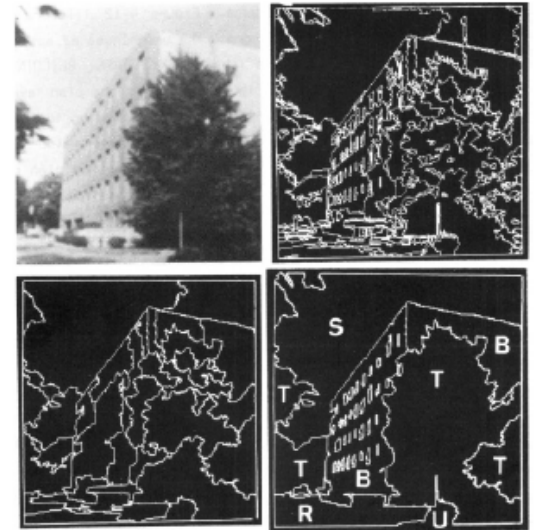
# Image Parsing



(a) Bottom-up process



(b) Top-down process

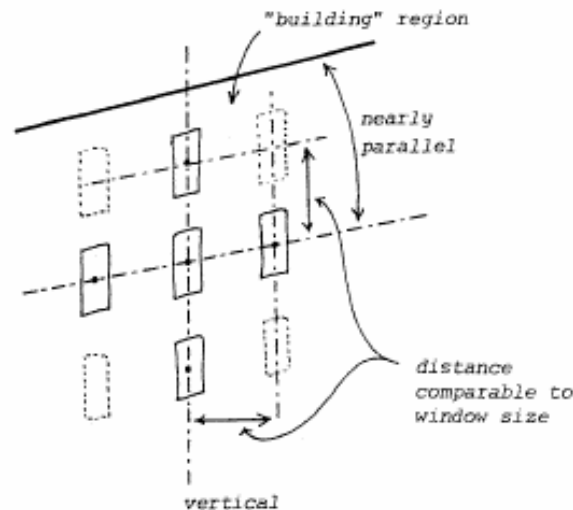


(c) Result

[Ohta & Kanade 1978]

- Guzman (*SEE*), 1968
- Hansen & Riseman (*VISIONS*), 1978
- Barrow & Tenenbaum 1978
- Brooks (*ACRONYM*), 1979
- Marr, 1982
- Ohta & Kanade, 1978
- Yakimovsky & Feldman, 1973

# What went wrong?



(a) "windows" and "building"

```
[ (ACT (IF (AND (IS-PLAN *PCH *MRGN) ..... (1)
              (*VERTICALLY-LONG *PCH))
  (THEN (GET-SET *PLSET (PLAN *MRGN) PATCHES) ..... (2)
        (AND (ALL-FETCH *WLIKE *PLSET ..... (3)
              (AND (IS (LABEL *WLIKE) NIL)
                    (*VERTICALLY-LONG *WLIKE)))
        (ALL-FETCH *WIND *WLIKE ..... (4)
          (THERE-IS *WK *WLIKE
            (*W-RELATION *WIND *WK))))))
  (THEN (CONCLUDE P-LABEL B-WINDOW)
        (FOR-EACH *WIND (AND (MUST-BE *WIND P-LABEL B-WINDOW)
                              (DONE-FOR *WIND)))
        (SCORE-IS (ADD 2.1 (DIV (NUMBER-OF *WIND) 100.0))))
    (*PCH *MRGN)]
```

(b) listing of the to-do rule for "windows" detection



# What went wrong?

## Appendix-B Complete Listing of the Model

```

aSCENE knowledge-block-of-scene

(OBJECTS (aSKY aTREE aBUILDING aROAD aUNKNOWN)
 SUB-OBJECTS (aB-WINDOW aCAR aC-SHADOW)
 KEY-PATCH-IS ( (GREATERP (AREA aPOCH) 380) (aPOCH) )

PLAN-IMAGE-GENERATION ( (DIV (BOUNDARY-LENGTH aPOCH aPOCH)
 (MULT (R-G-B-DIFFERENCE aPOCH aPOCH)
 (BOUNDARY-CONTRAST aPOCH aPOCH)))
 (aPOCH aPOCH) )

IF-PLAN-IS-MODIFIED (IF-DONE (
  rule-for-horizon-detection
  ( (ACT (IF (IS (OF HORIZON (SCENE)) NIL)
    (ALL-FETCH aHORN aPLAN-REGIONS
      (IF (AND (NOT (PROBABLY aROAD aHORN))
        (NOT (TOUCHING aHORN LOW-SIDE))
        (ALL-FETCH aHORN aPLAN-REGIONS
          (IF (AND (MAY-BE aROAD aHORN)
            (ABOVE aHORN aHORN)
            (NOT (aSAME-COLOR aHORN aHORN))
            (FACING HORIZONTALLY aHORN aHORN)
            (MULT (SUB (FACING HORIZONTALLY aHORN aHORN) 2.5)
              (SUB (MIN (ASK-VALUE aROAD aHORN) 8.6)
                (ASK-VALUE aROAD aHORN) ))))
            (VALUE aHORN aHORN) ))
        (THEN (MEMO (SCENE) ROAD-ZONE
          (WITH (MSR-LOW-SIDE aHORN) 256 1 256))
          (MEMO (SCENE) HORIZON (MSR-LOW-SIDE aHORN))
          (EXECUTE PLAN-EVALUATION) )) ) )
  P-SELECT (TO-DO (
    rule-for-initial-start
    ( (ACT (AND (PROBABLY BUILDING aPOCH (NOTFOUND BUILDING))

```

```

  rule-for-tree-occlusion
  ( (ACT (AND (aDARK aPOCH (aUPPER aPOCH)
    (OR (TOUCHING aPOCH UP-SIDE) (TOUCHING aPOCH SIDE))
    (THERE-IS aTR aREGIONS
      (AND (IS (LABEL aTRI) TREE)
        (ABOVE aPOCH aTRI)
        (TOUCHING aTR SIDE)
        (aWITH-IN2 aPOCH (Y-ZONE aTRI))))
    (THEN (CONCLUDE P-LABEL TREE)
      (CONCLUDE O-MERGE (WITH OCCLUDE aTR FRAME))
      (SCORE-IS 1.8))) (aPOCH) )
  rule-for-tree-garbage
  ( (ACT (PROBABLY TREE aPOCH)
    (THEN (CONCLUDE P-LABEL TREE)
      (SCORE-IS (ASK-VALUE TREE aPOCH))) (aPOCH) ) )
  P-LABEL (IF-DONE (
    if-done-rule-to-be-activated-when-keypatch-is-labeled
    ( (ACT (NOT (IS (OF PLAN aPOCH) NIL))
      (THEN (EXECUTE PLAN-EVALUATION)) (aPOCH) ) ) )

```

```

aSKY knowledge-block-of-sky

(PROPERTY-RULES (
  (GEN (NOT (aLOWER aHORN)) (1.0 , 0.6)) (aHORN) )
  (GEN (aSHINING aHORN) (1.0 , 0.2)) (aHORN) )
  (GEN (OR (aBLUE aHORN) (aGREY aHORN)) (1.0 , 0.2)) (aHORN) )
  (GEN (NOT (aTEXTURAL aHORN)) (1.0 , 0.7)) (aHORN) )
  (ISTR (TOUCHING aHORN UP-SIDE) (0.7 , 0.2)) (aHORN) )
RELATION-RULES (
  (ISTR (AND (aLINEAR-BOUNDARY aHORN aHORN2)
    (IF (aLINEAR-BOUNDARY (POSITION DOWN aHORN aHORN2)))
      (0.8 , 0.5) FOR SKY) (aHORN aHORN2))
  (ISTR (IF (NOT (IS (OF BUILDING-ZONE (SCENE))) NIL))
    (FUZZY1 (O-RATIO aHORN (OF BUILDING-ZONE (SCENE))) 0.5 0.3)
      (0.0 , 0.5) FOR SCENE) (aHORN) )

```

```

  (THEN (CONCLUDE P-LABEL BUILDING))
  (SCORE-IS (ADD 4.0 (CONFIDENCE-VALUE aPOCH))) (aPOCH)
  ( (ACT (AND (PROBABLY aROAD aPOCH (NOTFOUND ROAD))
    (THEN (CONCLUDE P-LABEL ROAD)
      (SCORE-IS (ADD 4.0 (CONFIDENCE-VALUE aPOCH))) (aPOCH)
    ( (ACT (AND (PROBABLY SKY aPOCH (NOTFOUND SKY))
      (THEN (CONCLUDE P-LABEL SKY)
        (SCORE-IS (ADD 4.0 (CONFIDENCE-VALUE aPOCH))) (aPOCH)
    ( (ACT (AND (PROBABLY TREE aPOCH)
      (NOT (THERE-IS aTR aREGIONS
        (AND (IS (LABEL aTRI) TREE)
        (OR (TOUCHING (PLAN aPOCH) (PLAN aTRI)
          (aWITH-IN2 (PLAN aPOCH)
            (Y-ZONE2 38 (PLAN aTRI))))))
      (THEN (CONCLUDE P-LABEL TREE)
        (SCORE-IS (ADD 4.0 (CONFIDENCE-VALUE aPOCH))) (aPOCH)
    rule-for-adjacent-wall-of-building
    ( (ACT (AND (MAY-BE BUILDING aPOCH)
      (THERE-IS aBL aREGIONS
        (AND (IS (LABEL aBL) BUILDING)
          (NOT (IS (OF SHAPE VIEW (OBJECT aBL) ))
            (IS (OF ADJACENT (OBJECT aBL)) NIL)
            (DIFFERENT-ZONE aPOCH aBL)))
      (THEN (CONCLUDE P-LABEL BUILDING)
        (CONCLUDE O-MERGE (WITH ADJACENT aBL)
          (SCORE-IS (ADD 5.0 (ASK-VALUE BUILDING aPOCH))) (aPOCH)
    rule-for-building-occlusion
    ( (ACT (AND (MAY-BE BUILDING aPOCH)
      (THERE-IS aBL aREGIONS
        (AND (IS (LABEL aBL) BUILDING)
          (aSAME-ZONE aPOCH aBL)
          (aSAME-COLOR aPOCH aBL)
          (THERE-IS aTR aKEYPATCHES
            (AND (BETWEEN aTR aPOCH aBL)
              (OR (IS (LABEL aTRI) TREE)
                (AND (IS (LABEL aTRI) BUILDING)
                  (NOT (IS (OBJECT aBL)
                    (OBJECT aTRI))))))
          (THEN (CONCLUDE P-LABEL BUILDING)
            (CONCLUDE O-MERGE (WITH OCCLUDE aBL (REGION aTRI))
              (SCORE-IS (ADD 1.0 (ASK-VALUE BUILDING aPOCH))) (aPOCH)

```

```

  P-SELECT (
    TO-DO (
      (ACT (MAY-BE SKY aPOCH)
        (THEN (SCORE-IS (ADD 2.0 (ASK-VALUE SKY aPOCH))) (aPOCH)
      ( (ACT (AND (IS-PLAN aPOCH aHORN) (aBRIGHT aPOCH)
        (THEN (SCORE-IS 3.0)) (aPOCH aHORN))
      ( (ACT (aBRIGHT aPOCH) (THEN (SCORE-IS 8.05)) (aPOCH) )
    IF-DONE (
      (ACT aTa (THEN (CONCLUDE P-LABEL SKY)
        (CONCLUDE R-MERGE (MASTER aPOCH))) (aPOCH) ) )
    APRIORI-VALUE-IS 0.1

```

```

aTREE knowledge-block-of-tree

(MADE-OF (aLEAVES)
PROPERTY-RULES (
  (GEN (aMIDDLE aHORN) (0.6 , 0.3)) (aHORN) )
  (ISTR (aHEAVY-TEXTURE aHORN) (0.8 , 0.2)) (aHORN) )
P-SELECT (
  TO-DO (
    (ACT (MAY-BE TREE aPOCH)
      (THEN (SCORE-IS (ADD 2.0 (ASK-VALUE TREE aPOCH))) (aPOCH)
    ( (ACT (AND (IS-PLAN aPOCH aHORN) (NOT (aSHINING aPOCH))
      (THEN (SCORE-IS 3.0)) (aPOCH aHORN))
    IF-DONE (
      (ACT aTa (THEN (CONCLUDE P-LABEL TREE)
        (CONCLUDE R-MERGE (MASTER aPOCH))) (aPOCH) ) )
    APRIORI-VALUE-IS 0.2

```

```

aROAD knowledge-block-of-road

(MADE-OF (OR (aASPHALT aCONCRETE)
SUB-OBJECTS (aCAR aC-SHADOW)
PROPERTY-RULES (
  (GEN (aMIDDLE aHORN) (0.8 , 0.4)) (aHORN) )
  (GEN (aHORIZONTALLY-LONG aHORN) (0.7 , 0.2)) (aHORN) )
  (ISTR (TOUCHING aHORN LOWER-SIDE) (0.9 , 0.2)) (aHORN) )
RELATION-RULES (
  (ISTR (AND (aSAME-COLOR aHORN aHORN2) (TOUCHING aHORN aHORN2))
    (0.9 , 0.2) FOR ROAD) (aHORN aHORN2))
  (GEN (IF (NOT (IS (OF HORIZON (SCENE))) NIL))
    (O-RATIO aHORN (OF ROAD-ZONE (SCENE)))
      (1.0 , 0.3) FOR SCENE) (aHORN) )
  APRIORI-VALUE-IS 0.2

```

```

aBUILDING knowledge-block-of-building

(MADE-OF (OR aCONCRETE aTILE aBRICK)
SUB-OBJECTS (aB-WINDOW)
PROPERTY-RULES (
  (GEN (aMIDDLE aHORN) (0.6 , 0.3)) (aHORN) )
  (ISTR (aMAYHOLE aHORN) (0.8 , 0.2)) (aHORN) )
  (ISTR (aMAYHOLE aHORN) (0.4 , 0.2)) (aHORN) )
  (GEN (aHOLELINE aHORN) (0.9 , 0.5)) (aHORN) )
RELATION-RULES (
  (GEN (AND (aLINEAR-BOUNDARY aHORN aHORN2)
    (IF (aLINEAR-BOUNDARY (NOT (POSITION UP aHORN aHORN2)))
      (0.8 , 0.4) FOR SKY) (aHORN aHORN2))
  (ISTR (IF (NOT (IS (OF BUILDING-ZONE (SCENE)) NIL))
    (AND (O-RATIO aHORN (OF BUILDING-ZONE (SCENE)))
      (aMAYHOLE aHORN))
      (0.5 , 0.3) FOR SCENE) (aHORN) ) )
P-SELECT (
  TO-DO (
    (ACT (AND (MAY-BE BUILDING aPOCH (aSAME-ZONE aPOCH aHORN)
      (THEN (CONCLUDE P-LABEL BUILDING)
        (CONCLUDE R-MERGE aHORN)
        (SCORE-IS (ADD 2.0 (ASK-VALUE BUILDING aPOCH)))
        (aPOCH aHORN)
    ( (ACT (AND (NOT (IS-PLAN aPOCH aHORN)) (aSAME-ZONE aPOCH aHORN)
      (MAY-BE BUILDING (PLAN aPOCH))
      (THEN (CONCLUDE P-LABEL BUILDING)
        (CONCLUDE R-MERGE aHORN)
        (SCORE-IS (ADD 1.95 (ASK-VALUE BUILDING (PLAN aPOCH))))
        (aPOCH aHORN)
    rule-for-window-extraction
    ( (ACT (IF (AND (IS-PLAN aPOCH aHORN) (aSAME-ZONE aPOCH aHORN)
      (VERTICALLY-LONG aPOCH) (aCONTACT aPOCH (PLAN aHORN)))
      (THEN (GET-SET aPSET (PLAN aHORN) PATCHES)
        (AND (ALL-FETCH aLIKE aPSET
          (AND (IS (LABEL aLIKE) NIL)
            (aSAME-ZONE aLIKE aHORN)
            (VERTICALLY-LONG aLIKE)
            (aCONTACT aLIKE (PLAN aHORN)))

```

```

    (THERE-IS aHk aLIKE (aRELATION aPOCH aHk))
    (ALL-FETCH aHND aLIKE
      (THERE-IS aHk aLIKE
        (aRELATION aHND aHk))))
    (THEN (CONCLUDE P-LABEL B-WINDOW)
      (FOR-EACH aHND (AND (MUST-BE aHND P-LABEL B-WINDOW)
        (aHND) )
      (SCORE-IS (ADD 2.1 (DIV (NUMBER-OF aHND) 100.0))))
      (aPOCH aHORN)
    ( (ACT (AND (IS-PLAN aPOCH aHORN) (aSAME-ZONE aPOCH aHORN)
      (THEN (CONCLUDE P-LABEL BUILDING)
        (CONCLUDE R-MERGE aHORN)
        (SCORE-IS 2.0)) (aPOCH aHORN)

```

```

  O-MERGE (IF-DONE (
    (ACT aTa (DESCRIBE-BUILDING (REGION aPOCH))) (aPOCH) ) )
  O-CREATE (IF-DONE (
    (ACT aTa (THEN (EXTRACT-BUILDING-SHAPE (REGION aPOCH))
      (DESCRIBE-BUILDING (REGION aPOCH))
      (EXECUTE PLAN-EVALUATION)) (aPOCH) ) )
  APRIORI-VALUE-IS 0.2

```

```

aROAD knowledge-block-of-road

(MADE-OF (OR aASPHALT aCONCRETE)
SUB-OBJECTS (aCAR aC-SHADOW)
PROPERTY-RULES (
  (GEN (aLOWER aHORN) (0.8 , 0.4)) (aHORN) )
  (GEN (aHORIZONTALLY-LONG aHORN) (0.7 , 0.2)) (aHORN) )
  (ISTR (TOUCHING aHORN LOWER-SIDE) (0.9 , 0.2)) (aHORN) )
RELATION-RULES (
  (ISTR (AND (aSAME-COLOR aHORN aHORN2) (TOUCHING aHORN aHORN2))
    (0.9 , 0.2) FOR ROAD) (aHORN aHORN2))
  (GEN (IF (NOT (IS (OF HORIZON (SCENE))) NIL))
    (O-RATIO aHORN (OF ROAD-ZONE (SCENE)))
      (1.0 , 0.3) FOR SCENE) (aHORN) )
  APRIORI-VALUE-IS 0.2

```

[Ohta & Kanade 1978]

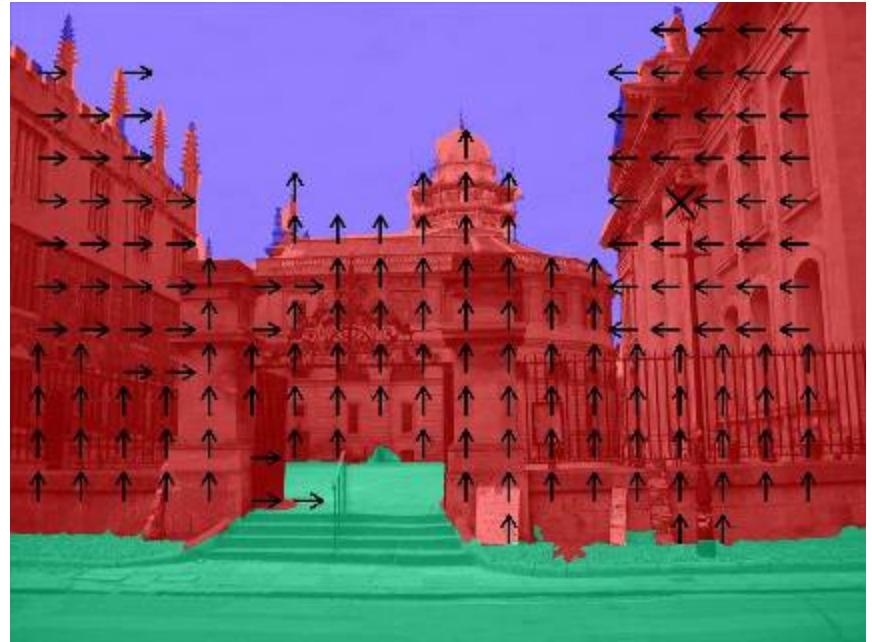
# Learn from data



...



# Scene Layout



**Goal:** learn labeling of image into 7 Geometric Classes:

- **Support (ground)**
- **Vertical**
  - Planar: facing **Left** ( $\leftarrow$ ), **Center** ( ), **Right** ( $\rightarrow$ )
  - Non-planar: **Solid** (X), **Porous** or wiry (O)
- **Sky**

# “Automatic Photo Pop-up”, SIGGRAPH 2005



# Examples of learning problems

- Recognizing hand-written digits in ZIP code
- Classifying email as spam or not
- Predicting the price of a stock 6 months from now
- Netflix problem – predict rating of a movie  $j$  by a customer  $i$
- Determine credit-worthiness for a mortgage or a credit card transaction

# Simple Warm-up Example

# Credit Approval

Let's use a conceptual example to crystallize the issues.

- Using salary, debt, years in residence, etc., approve for credit or not.
- No magic credit approval formula.
- Banks have lots of data.
  - customer information: salary, debt, etc.
  - whether or not they defaulted on their credit.

age	32 years
gender	male
salary	40,000
debt	26,000
years in job	1 year
years at home	3 years
...	...

Approve for credit?

A pattern exists. We don't know it. We have data to learn it.



# The Key Players

- Salary, debt, years in residence, ...
- Approve credit or not
- True relationship between  $\mathbf{x}$  and  $y$
- Data on customers

*input*  $\mathbf{x} \in \mathbb{R}^d = \mathcal{X}$ .

*output*  $y \in \{-1, +1\} = \mathcal{Y}$ .

*target function*  $f : \mathcal{X} \mapsto \mathcal{Y}$ .

(The target  $f$  is *unknown*.)

*data set*  $\mathcal{D} = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ .

( $y_n = f(\mathbf{x}_n)$ .)

$\mathcal{X}$   $\mathcal{Y}$  and  $\mathcal{D}$  are *given* by the learning problem;  
The target  $f$  is fixed but unknown.

We learn the function  $f$  from the data  $\mathcal{D}$ .



# Learning

- Start with a set of candidate hypotheses  $\mathcal{H}$  which you think are likely to represent  $f$ .

$$\mathcal{H} = \{h_1, h_2, \dots, \}$$

is called the hypothesis set or *model*.

- Select a hypothesis  $g$  from  $\mathcal{H}$ . The way we do this is called a *learning algorithm*.
- Use  $g$  for new customers. We hope  $g \approx f$ .

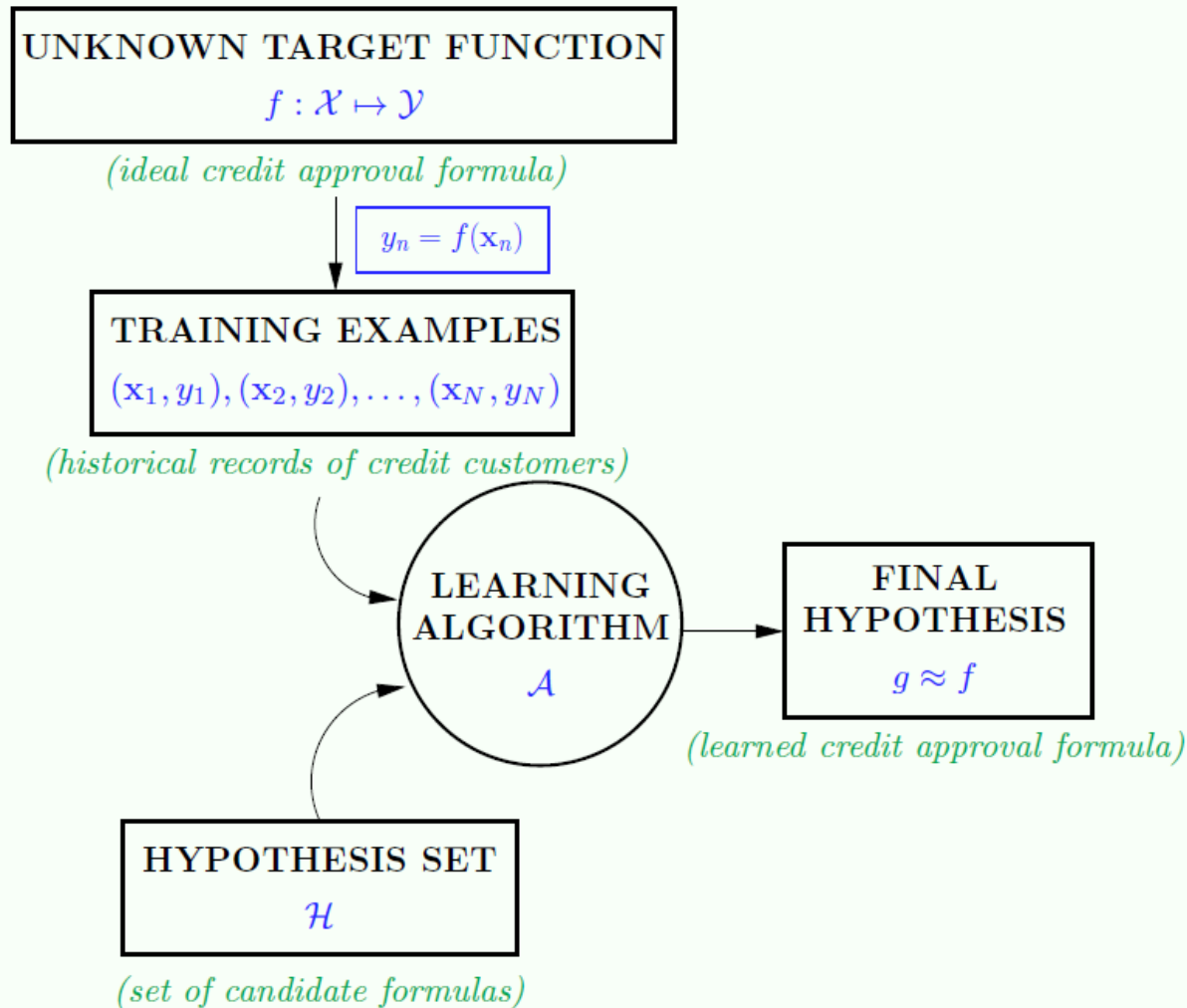
$\mathcal{X}$   $\mathcal{Y}$  and  $\mathcal{D}$  are *given* by the learning problem;

The target  $f$  is **fixed but unknown**.

*We choose  $\mathcal{H}$  and the learning algorithm*

This is a very general setup (eg. choose  $\mathcal{H}$  to be all possible hypotheses)

# Summary of the Learning Setup



# (Supervised) Learning Pipeline

- Collect training data
  - For each data point, define a **feature vector**  $x \in R^d$  and label  $y$
- Pick learning algorithm **A** / hypothesis set **H**
- At **training time**:
  - Train on the training data, to produce final hypothesis **g**
- At **test time**:
  - Apply **g** to new “test data”



Another Example:

# Binary Digit Classification



# Classification Pipeline

- Collect Training Images

- Positive: 
- Negative: 

- Training Time

- Compute **feature vectors** for positive and negative example images
- Train a **classifier**

- Test Time

- Compute feature vector on new test image:
- Evaluate classifier



# Creating feature vector

# Creating Feature Vector

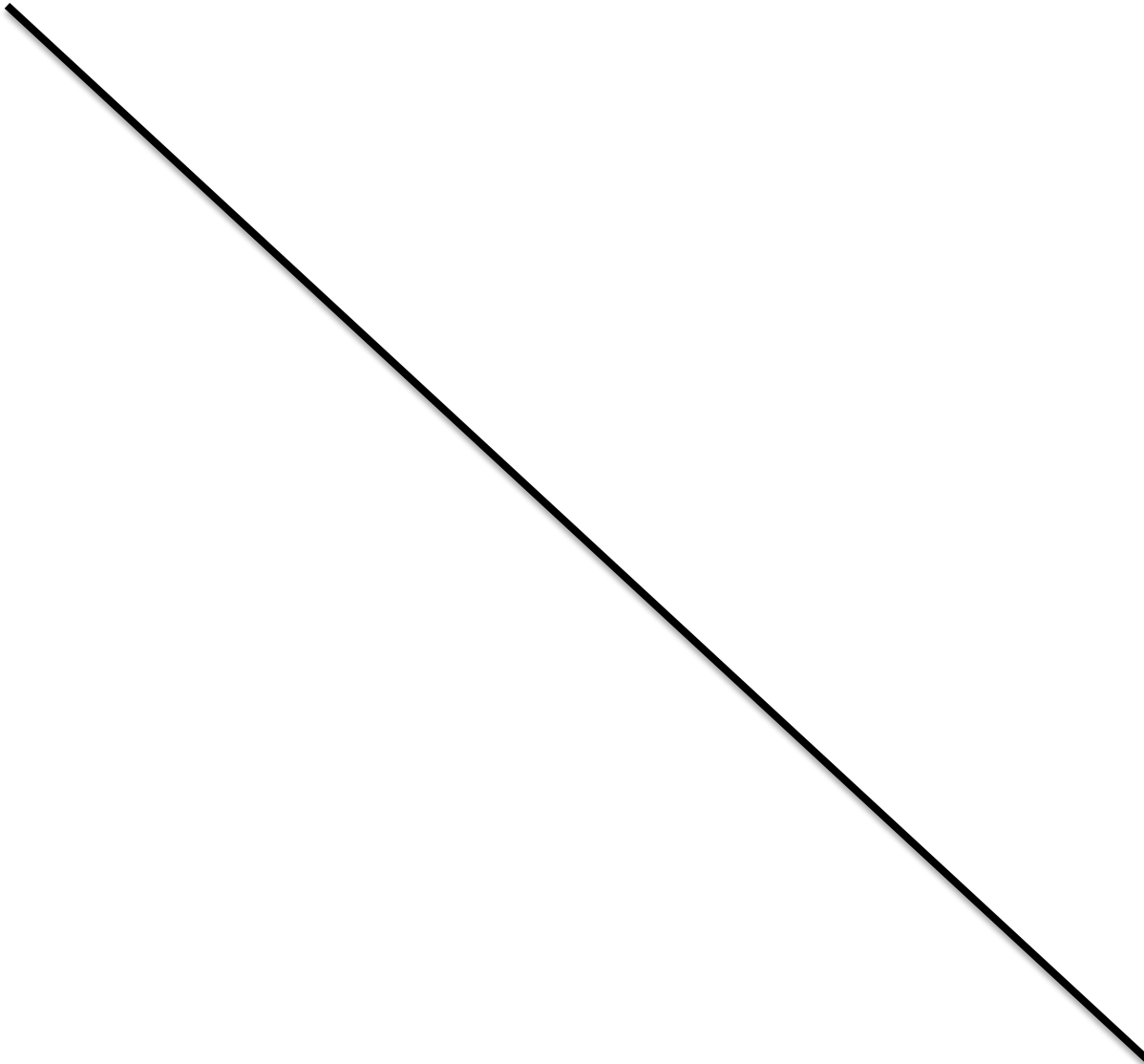
In feature space, positive and negative examples are just points...



How do we classify a new point?

Nearest neighbor rule  
“transfer label of nearest example”

# Linear classifier rule

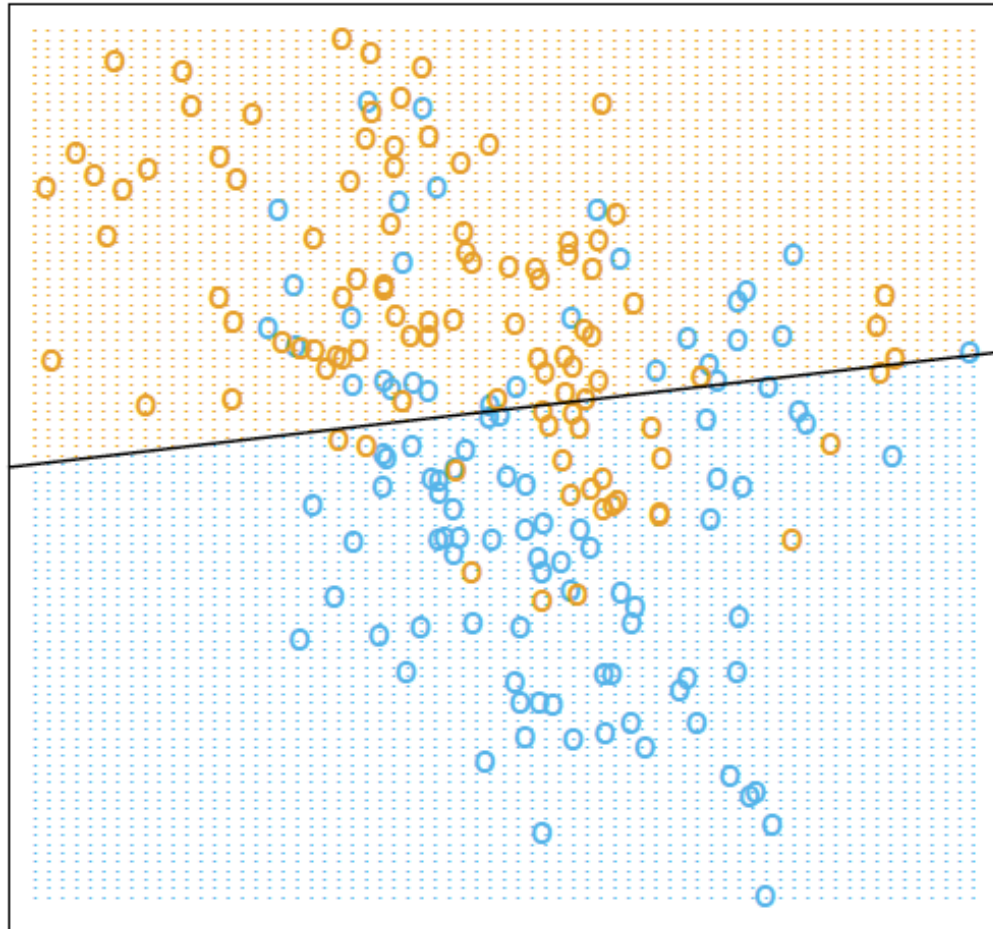


# Parametric vs. Non-Parametric

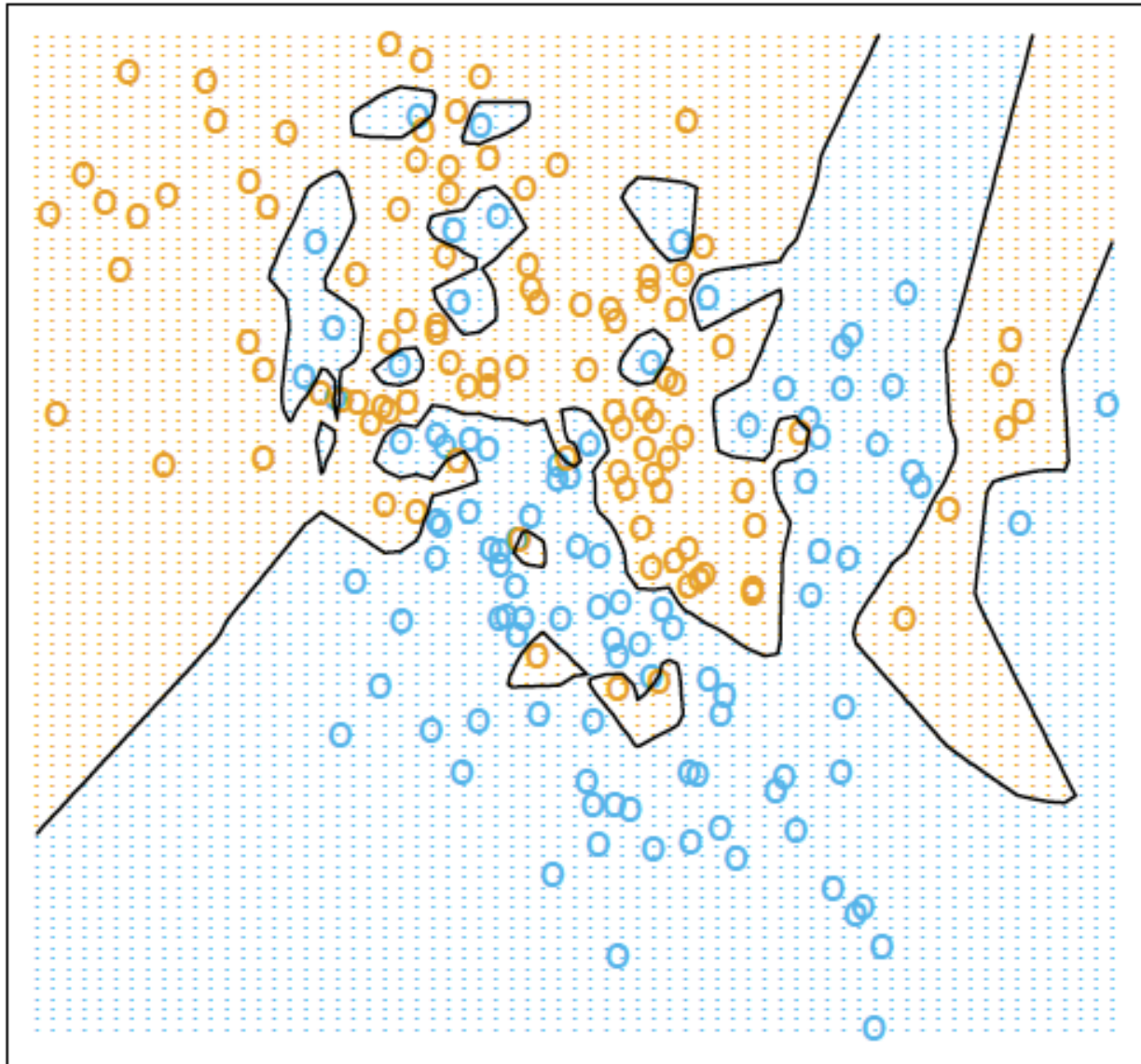
- Linear Classifier (Parametric)
  - small number of parameters, e.g.  $\mathbf{w}$  and  $b$
  - very fast at test time:  $\mathbf{w} \cdot \mathbf{x} + b$
  - after training is done, can throw away the data!
- Nearest-Neighbor (Non-Parametric)
  - Number of parameters grows with size of dataset
  - Very slow at test time --  $O(N)$
  - No training
- Which one works better?

# Linear Classifier

[Source : Hastie, Tibshirani, Friedman]



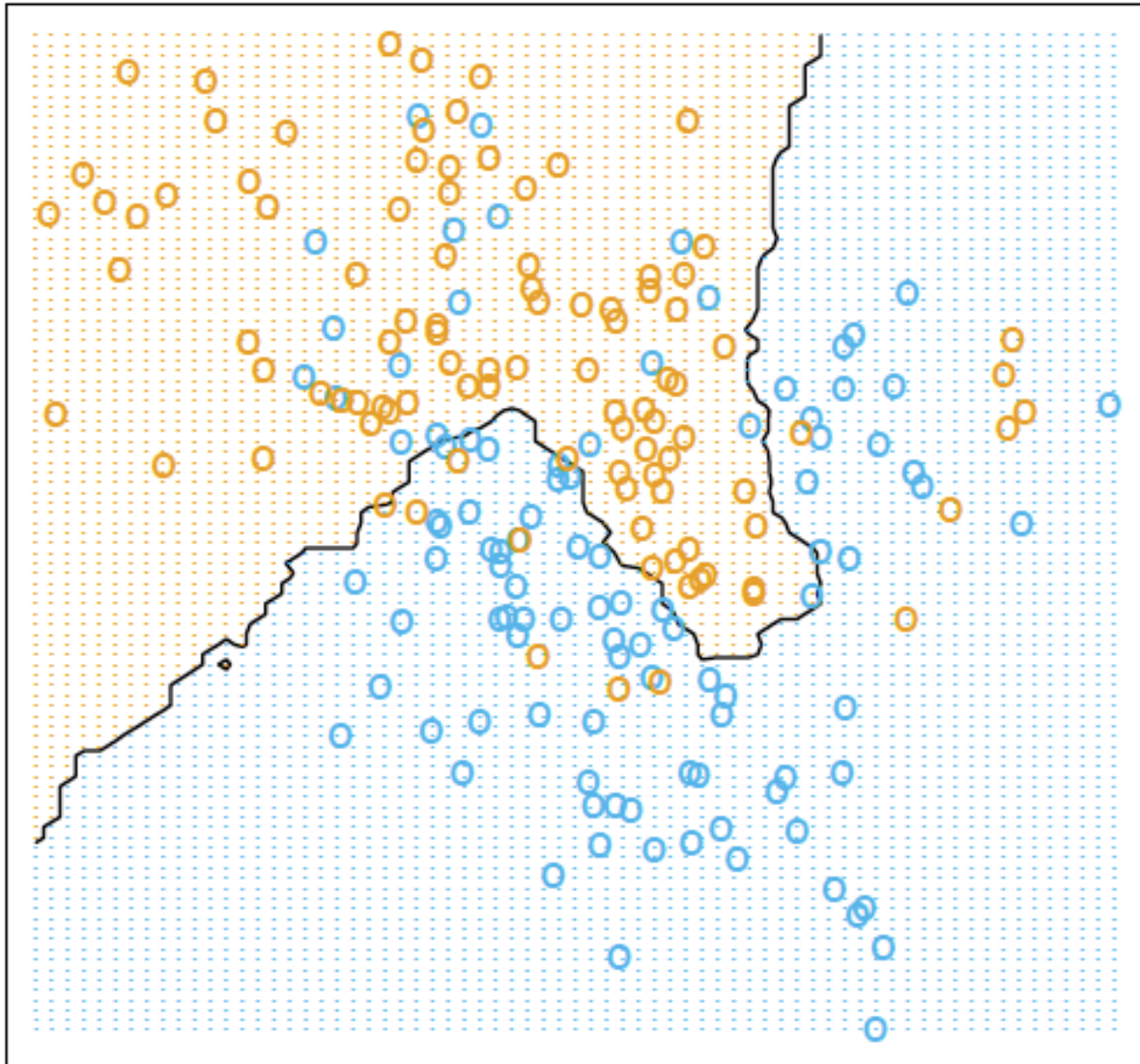
# Nearest Neighbor



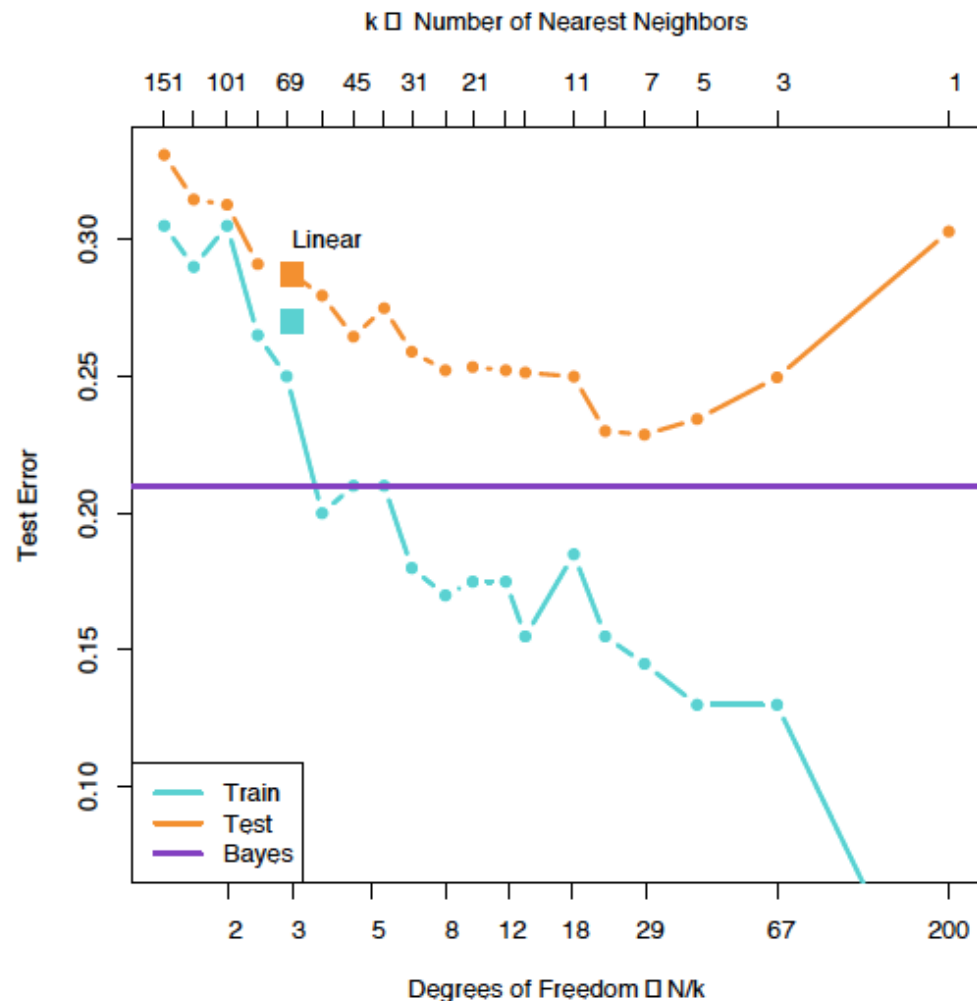
# Two kinds of error

- Training set error
  - We train a classifier to minimize training set error.
- Test set error
  - At run time, we will take the trained classifier and use it to classify previously unseen examples. The error on these is called test set error.

# 15-Nearest Neighbor







**FIGURE 2.4.** Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size 200 was used, and a test sample of size 10,000. The orange curves are test and the blue are training error for  $k$ -nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.

# Over-fitting

- If the test set error is much greater than training set error, that is called **over-fitting**.

# Historical Anecdote

- In the 1970s, DARPA wanted to find tanks in photographs...

Positive  
Set



Negative  
Set



# Validation and Cross-Validation

- To avoid over-fitting, we can measure error on a held-out set of training data, called the **validation set**.
- What if we can't spare so much data?
- We could divide the data into  $k$ -folds, use  $k-1$  of these to train and test on the remaining fold. This is **cross-validation**
  - In the limit, **leave-one-out-cross-validation**

We often have a choice of **hyper-parameters**. We can use cross-validation to pick them.

- The  $K$  in  $K$ -nearest neighbors
- $C$  in Support Vector Machines
- The order of polynomial basis in regression
- The coefficient of the regularization term
- The depth of a decision tree
- The number of layers and the number of hidden nodes in a neural network

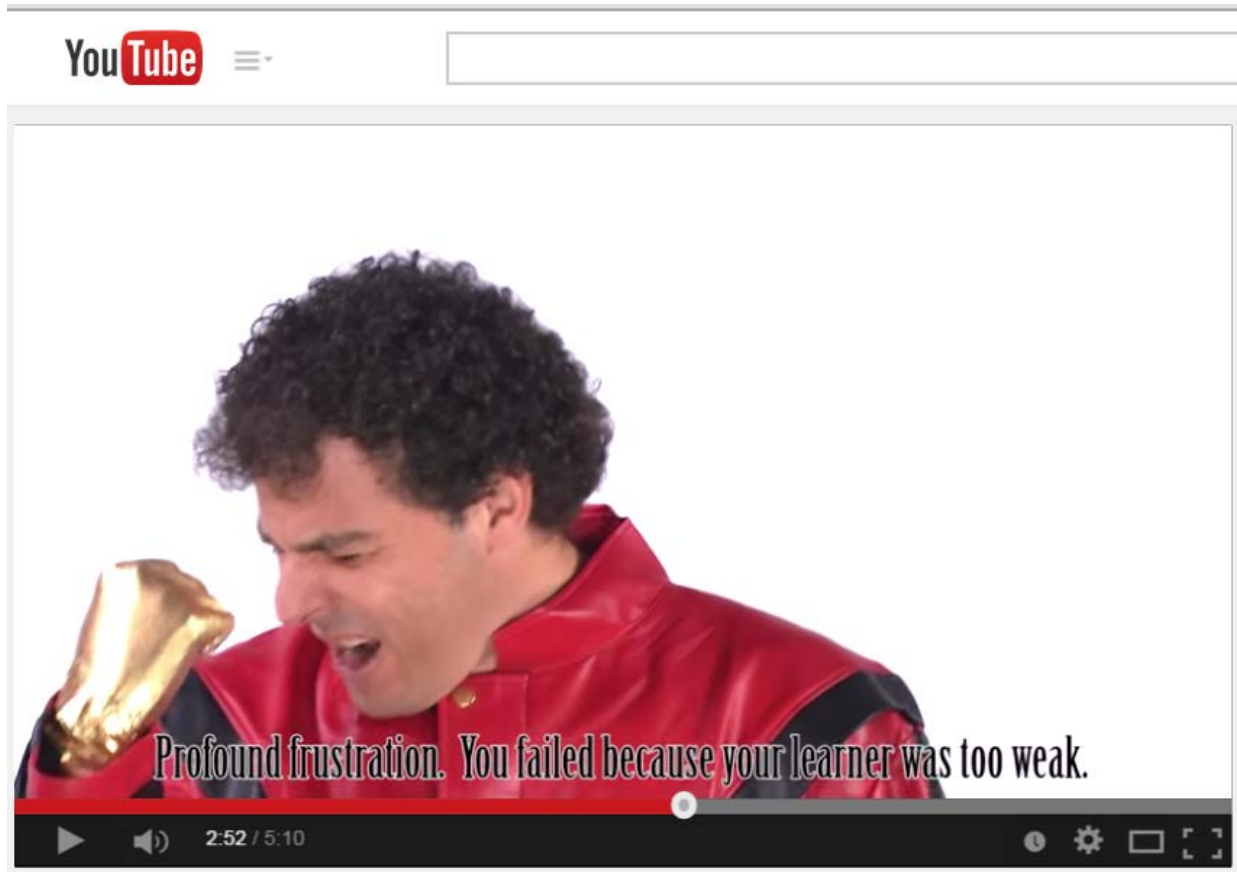
# IMPORTANT: Good ML hygiene

- Training set: to learn model parameters
- Validation set: to tune hyper-parameters, pick the best model, etc
- Test set: to get the final performance number only. Keep in a vault. **Run once**, at the very end!
- Data snooping:
  - Sadly common
  - See recent [Baidu fiasco](#) (NYTimes)
  - Scary when researchers in medicine, pharmaceuticals, etc peek into the test set!



- Kaggle runs many Machine Learning competitions
- We will (probably) use it in this class
  - Digits competition
  - Spam competition
- There will be two test sets
  - Public: results available while competition is running (limit: one submission per day)
  - Private: results only known once the competition is over
- If your results on Public are much better than on Private, we will know you **over-fitted** on test data – a capital sin!

# The Overfitting Song!



<https://www.youtube.com/watch?v=DQWI1kvmwRg>