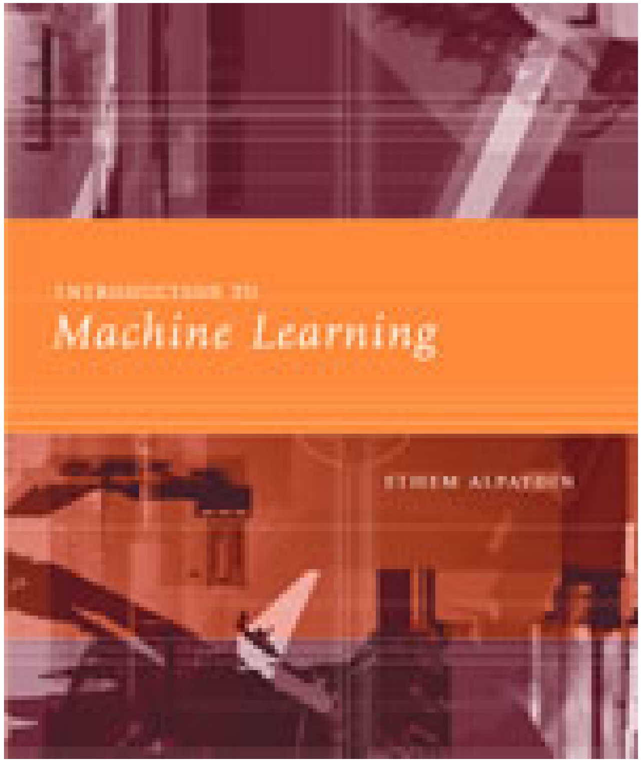
CS340 Machine learning
Lecture 3
Classification

Admin

- HW1 is due next Monday 18th
- Discussion section (optional, but recommended - the TAs will go over homework problems, etc.)
  - T1A, 3:00 - 4:00pm Thursdays, DMP101
  - T1B, 8:30 - 9:30am Tuesdays, DMP201
- This week only: extra Matlab tutorial by Prof Ian Mitchell on Wed 13th
  - 9 - 10am, CS x250
  - 5 - 6pm, DMP 301
- My office hours (changed)
  - Wed 1-2pm, CS 187

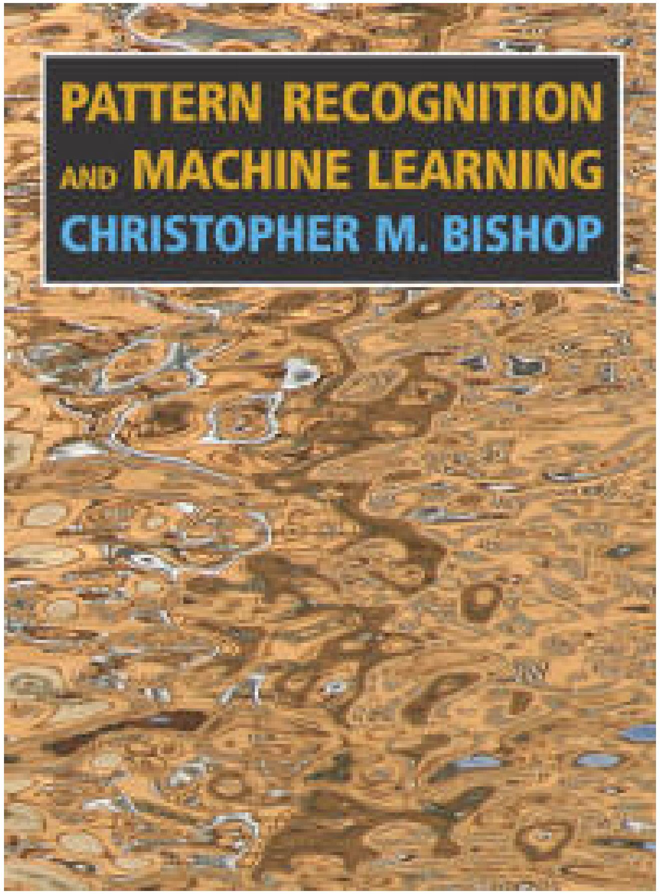
Textbook

- Required textbook "Introduction to machine learning", Ethem Alpaydin
- Has arrived in bookstore, $64

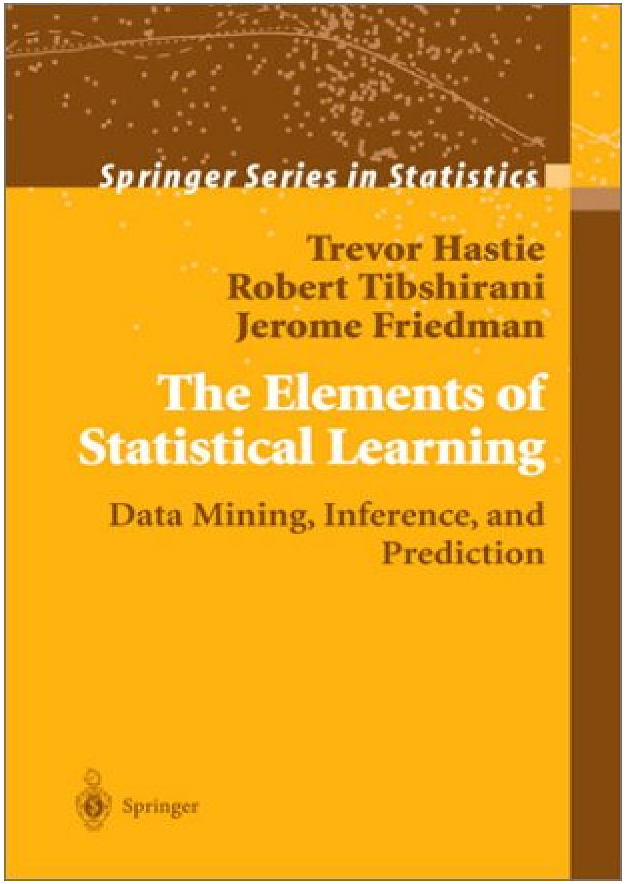


Other recommended books (more advanced)

20 copies on order ($90)



Order yourself from Amazon etc.



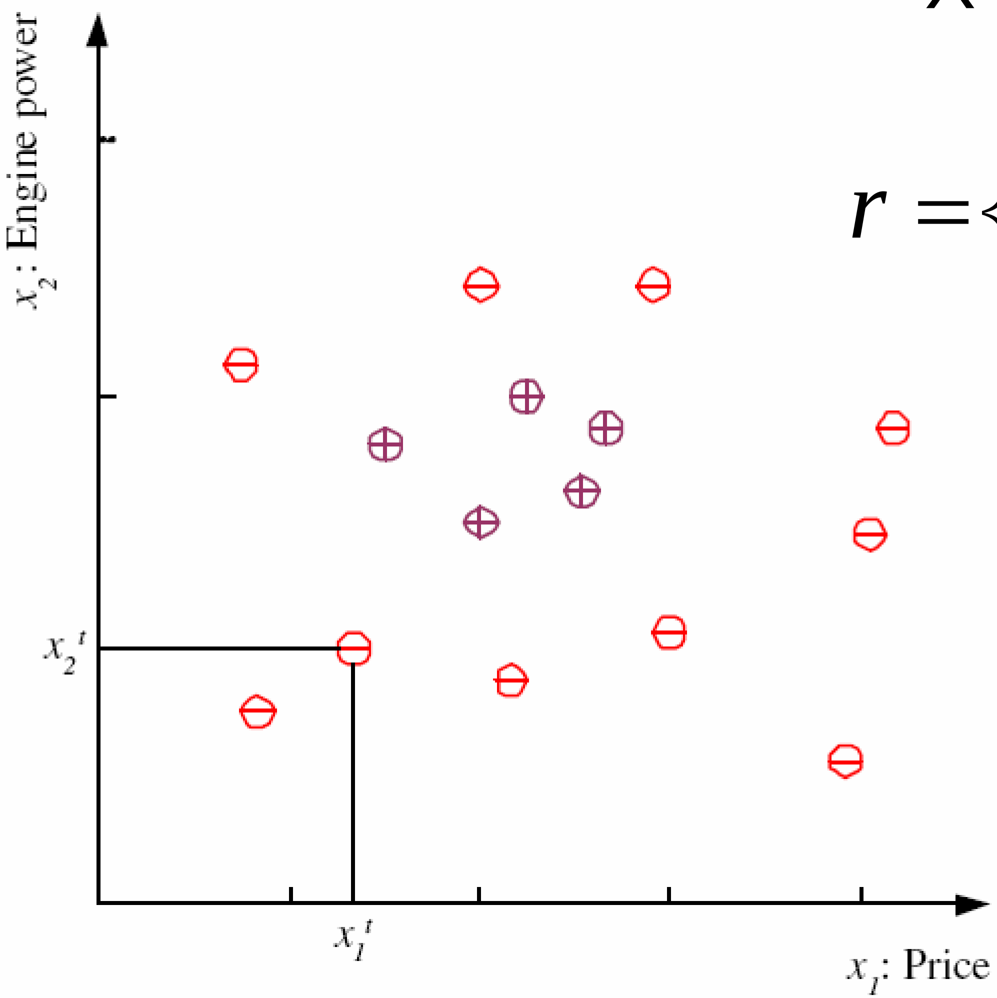
Learning a Class from Examples

- Class C of a "family car"
  - Prediction: Is car *x* a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output:

  Positive (+) and negative (–) examples
- Input representation:

  x_1: price, x_2 : engine power

Training set X



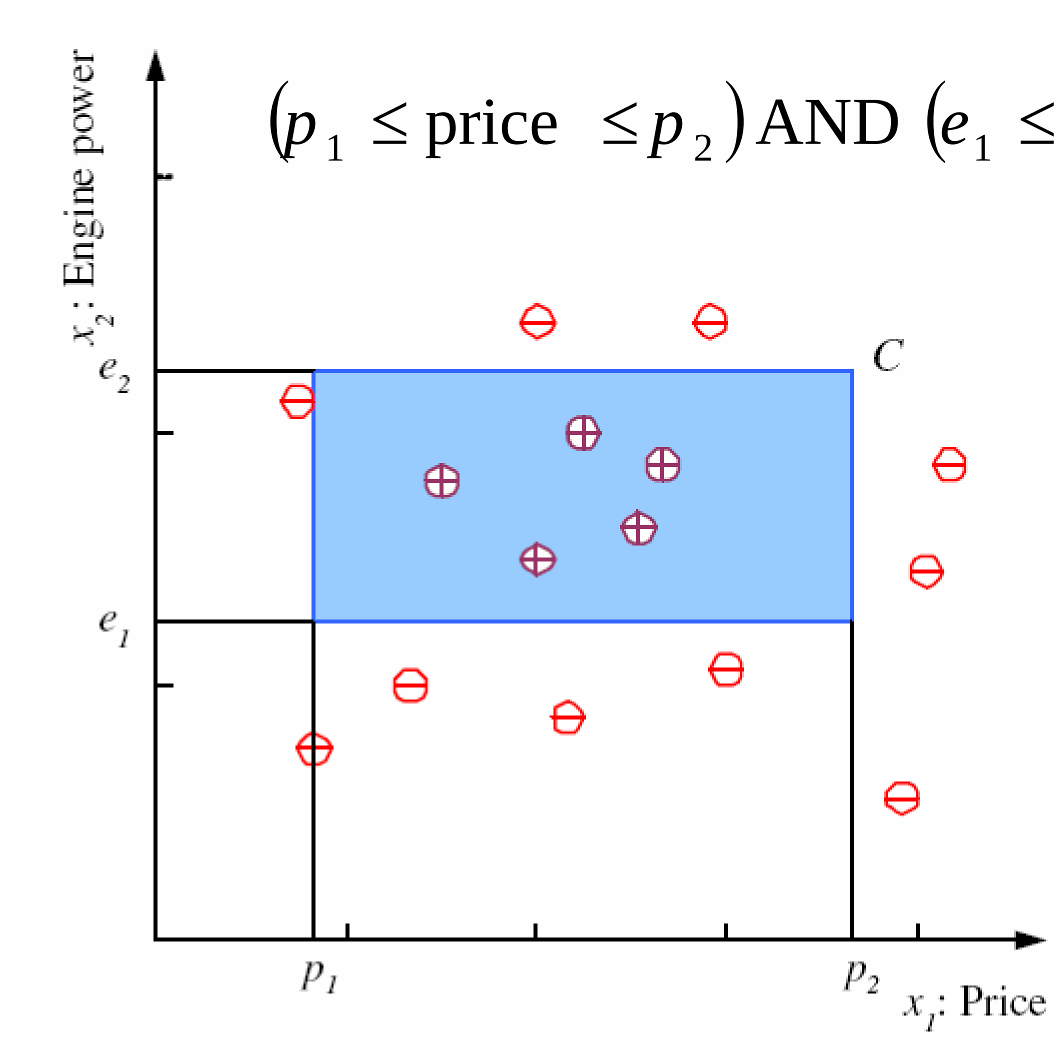
$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

$$r = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is positive} \\ 0 \text{ if } \mathbf{x} \text{ is negative} \end{cases}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

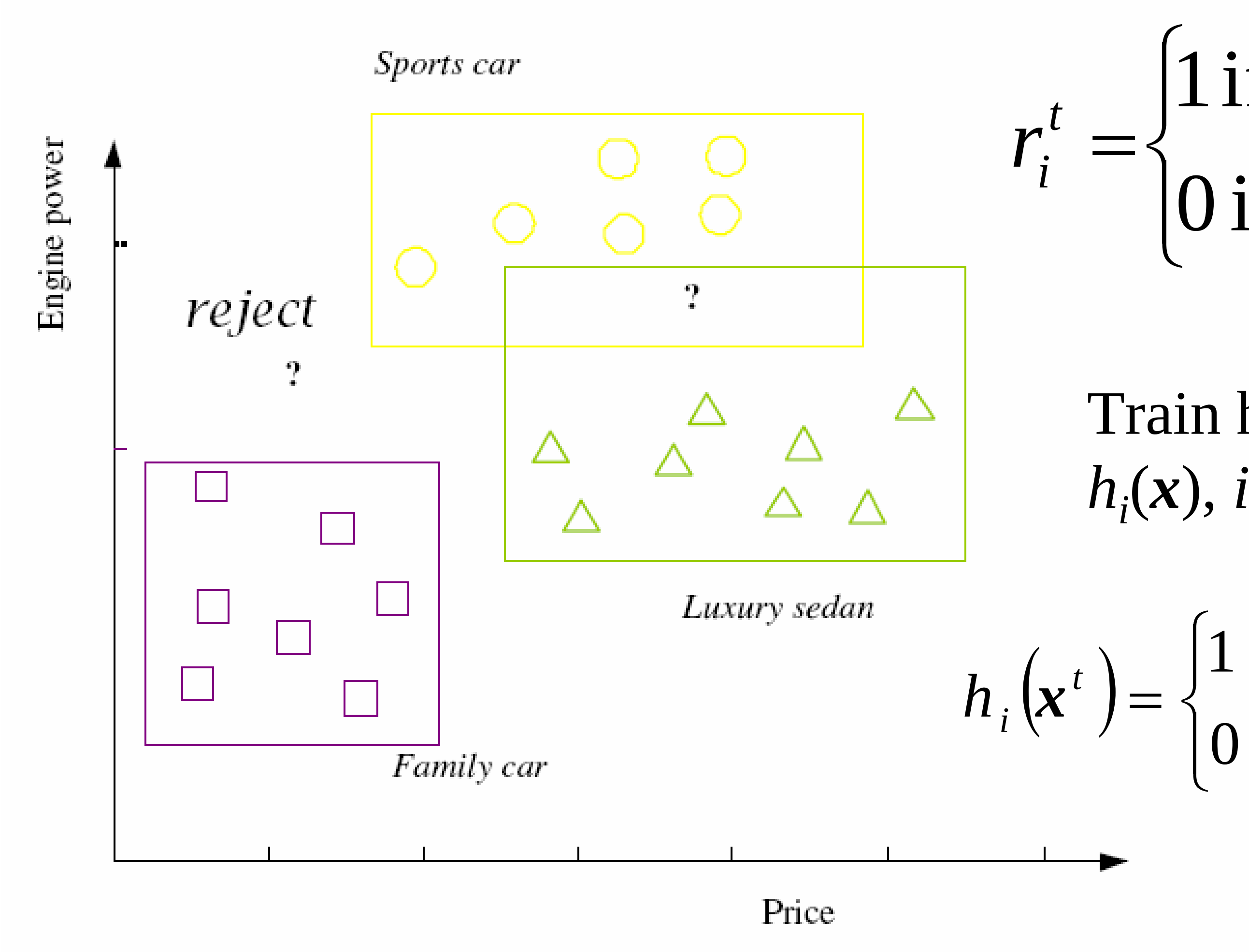
Class C

$$(p_1 \le \text{price} \le p_2) \text{ AND } (e_1 \le \text{engine power} \le e_2)$$



Multiple Classes, C_i i=1,...,K

$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^N$$



$$r_i^t = \begin{cases} 1 \text{ if } \boldsymbol{x}^t \in C_i \\ 0 \text{ if } \boldsymbol{x}^t \in C_j, j \neq i \end{cases}$$

Train hypotheses
$h_i(\boldsymbol{x})$, $i = 1,...,K$:

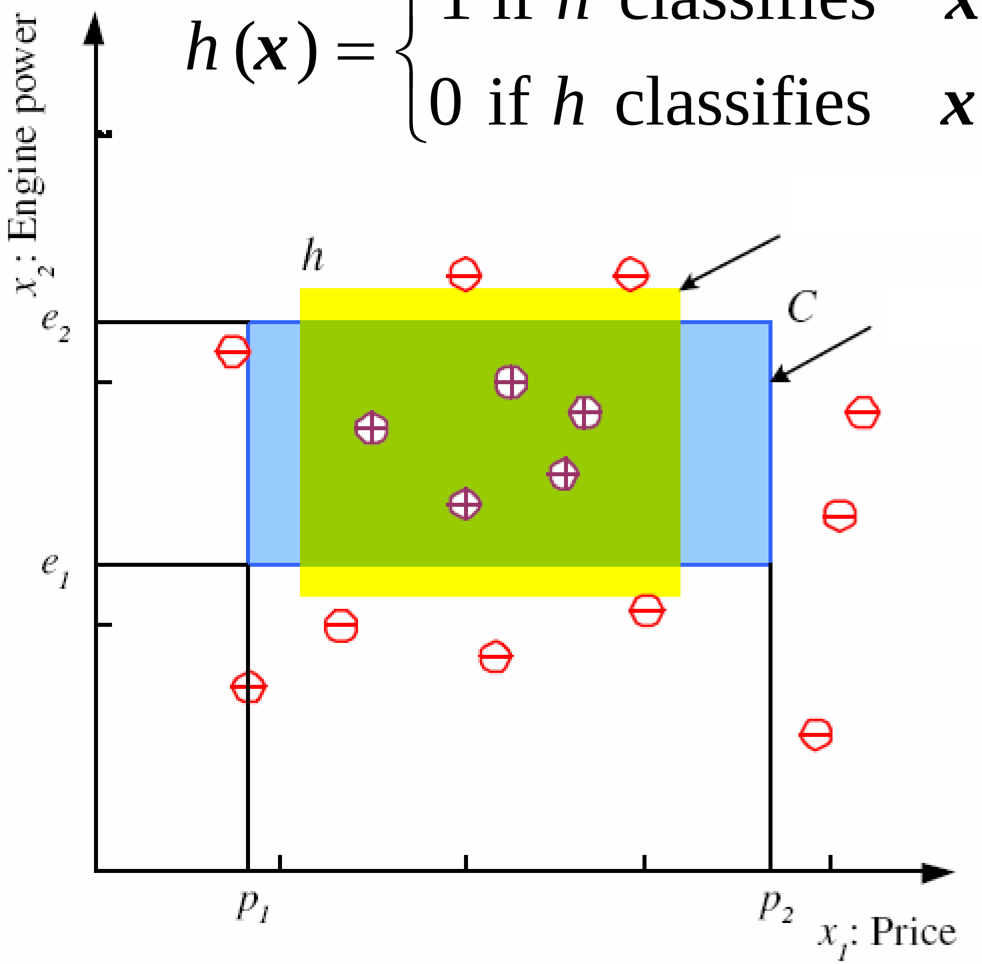
$$h_i(\boldsymbol{x}^t) = \begin{cases} 1 \text{ if } \boldsymbol{x}^t \in C_i \\ 0 \text{ if } \boldsymbol{x}^t \in C_j, j \neq i \end{cases}$$

Today we will focus on binary classification problems

Hypothesis class *H*

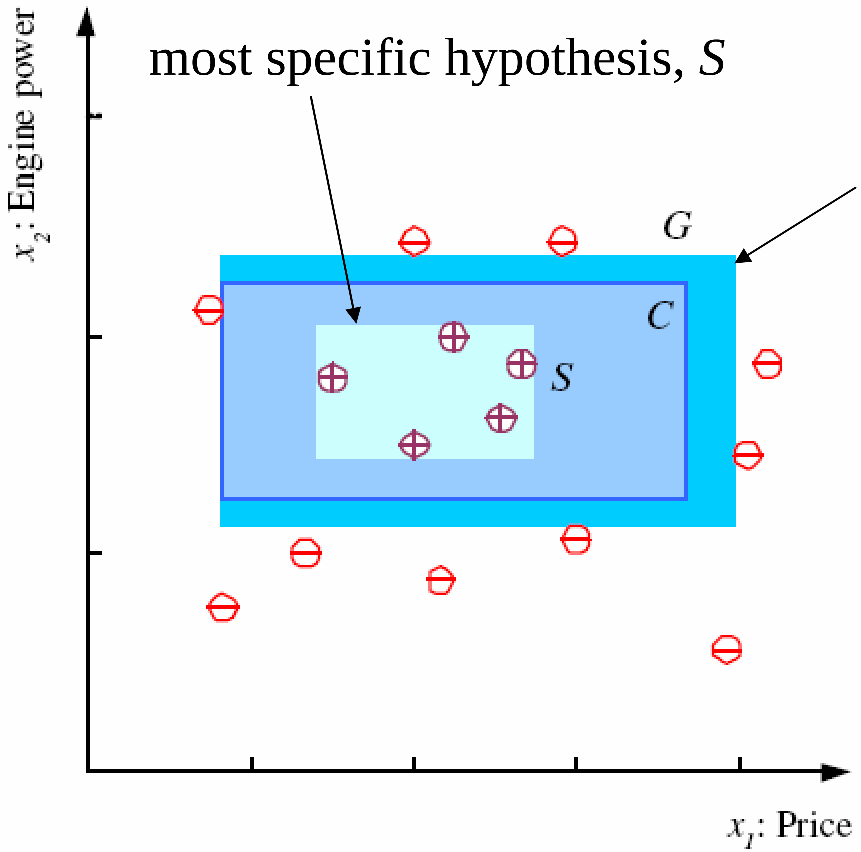
Hypothesis = yellow rectangle, Truth = blue rectangle

$$h(\boldsymbol{x}) = \begin{cases} 1 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as positive} \\ 0 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as negative} \end{cases}$$



S, G, and the Version Space

S is the smallest rectangle that contains all the +ve's.
G is the largest rectangle that excludes all the -ve's.
The version space is the set of consistent hypotheses (zero training error).



$h \in$ H, between S and G is consistent

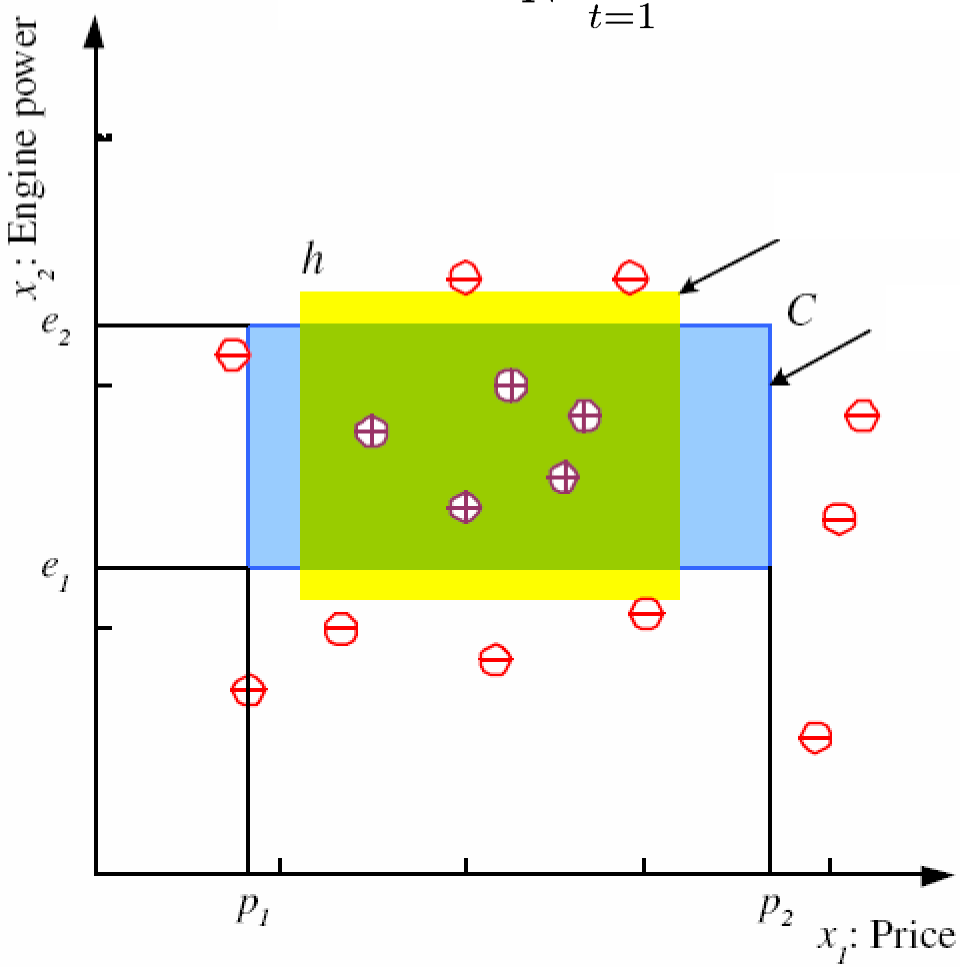
and make up the version space

(Mitchell, 1997)

Training set (empirical) error

$$err(D) = \frac{1}{N}\sum_{t=1}^{N} I(h(x^t) \neq y^t)$$

Zero training errors

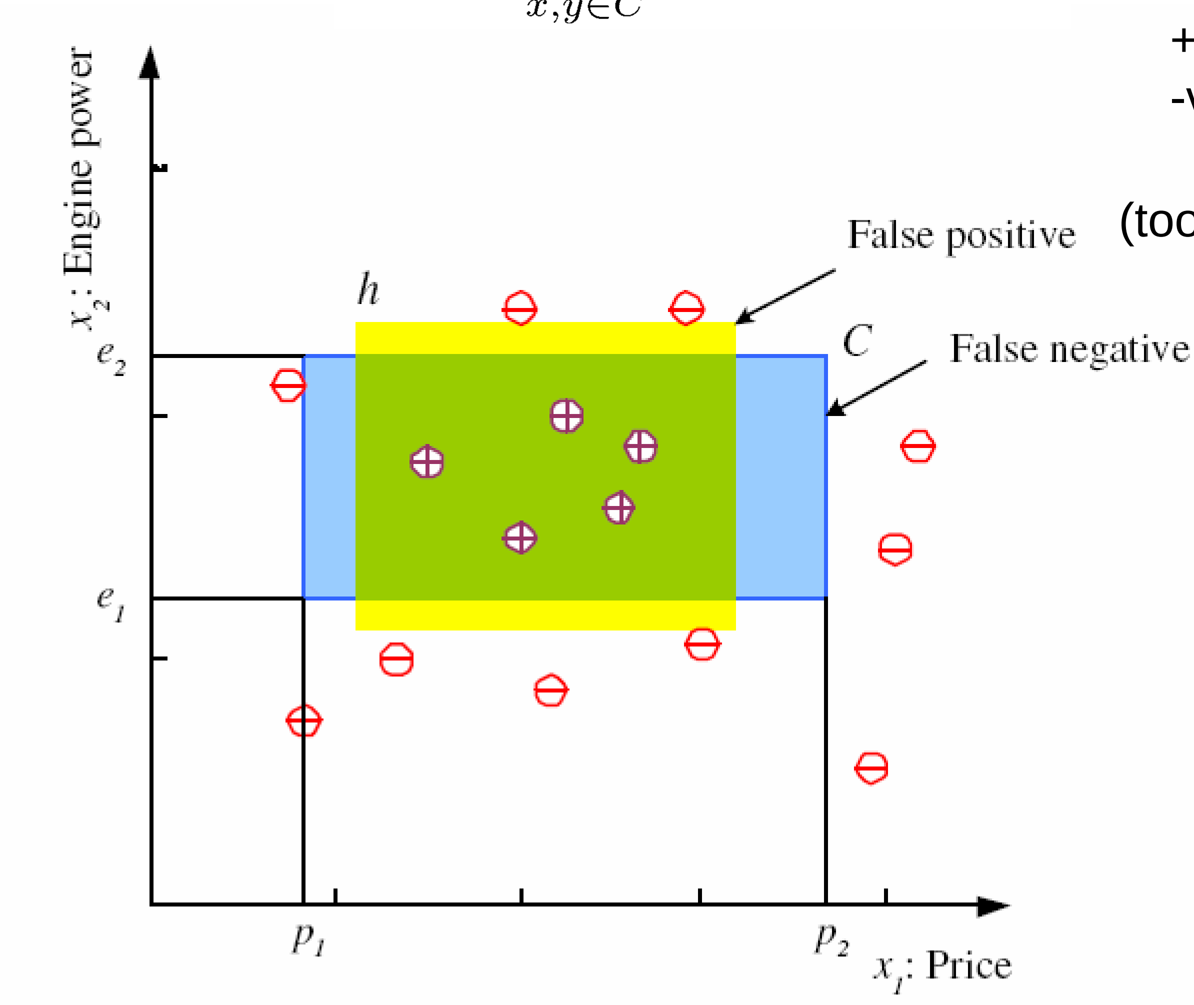


Notation: Alpaydin uses E for error, I'll use err (since E is for expectation)

Generalization error

$$Eerr = \sum_{x,y \in C} p(x,y) I(h(x) \neq y)$$

Error rate on points sampled from R^2 -
+ve inside blue rectangle
-ve outside blue rectangle



(too tall)

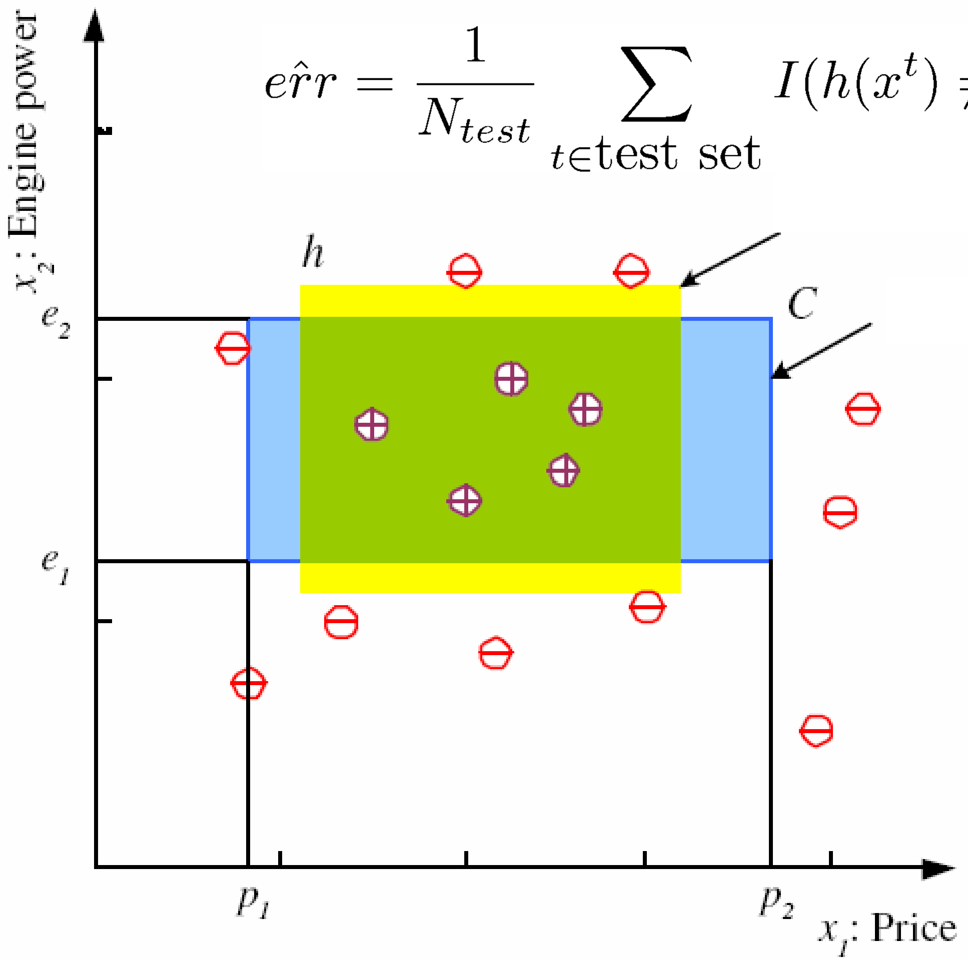
(too narrow)

Notation: Alpaydin uses E for error, I'll use err (since E is for expectation)

Test set error

We can approximate the generalization error by using a set of test points drawn from the true (blue) concept

$$\hat{err} = \frac{1}{N_{test}} \sum_{t \in \text{test set}} I(h(x^t) \neq y^t)$$

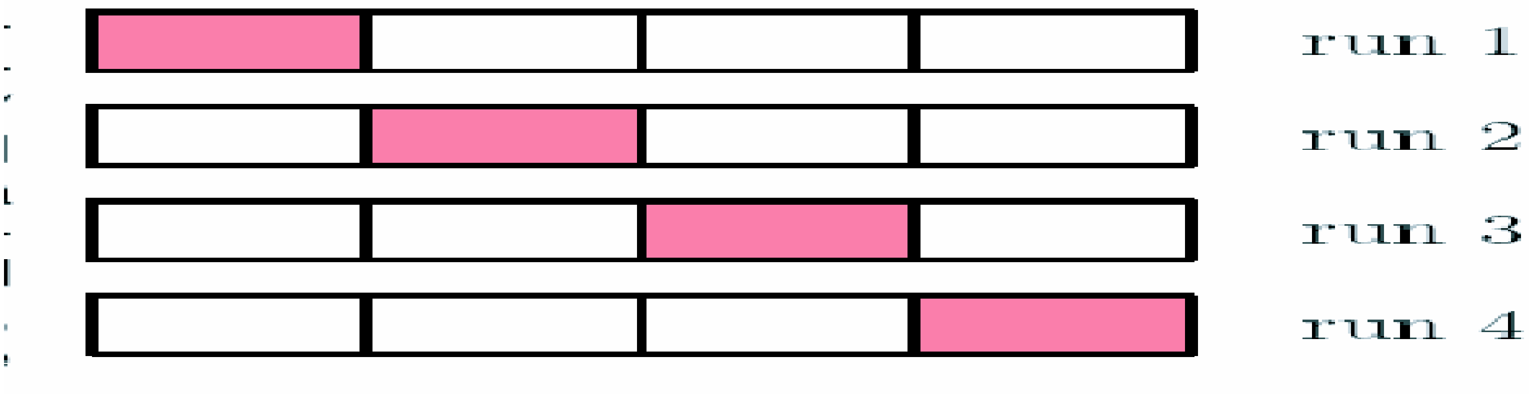


Cross validation

Since we don't have access to the test set (by assumption),
we hold back a fraction of the training data, called a validation set, and measure performances on that.
This gives us an estimate of the test set error E[err].
We can repeat this K times to get an average (K-fold CV).

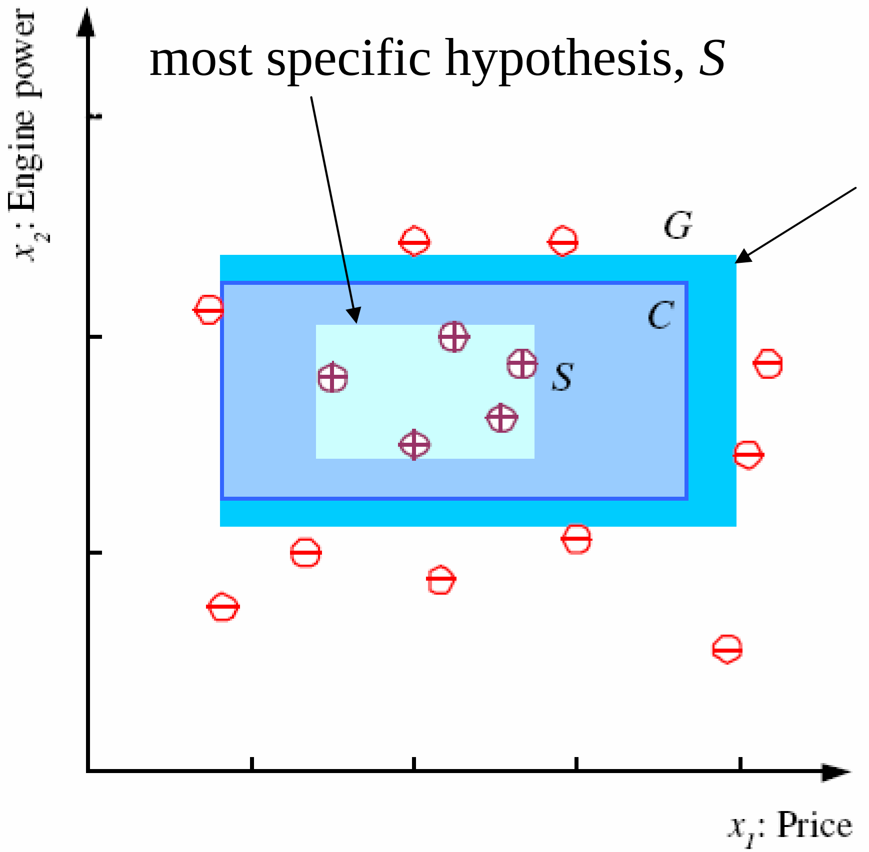
$$\hat{err}_k = \frac{1}{N_k} \sum_{t \in fold(k)} I(h(x^t) \neq y^t)$$

$$\hat{err} = \frac{1}{K} \hat{err}_k$$



FP/FN tradeoff

S and G both have zero training error, but make different errors on the test set. S has a lower false positive rate, and G has a lower false negative rate.



$$p_{fp} = p(x \in h | x\text{-ve}) = p(h(x) = 1 | y = 0)$$

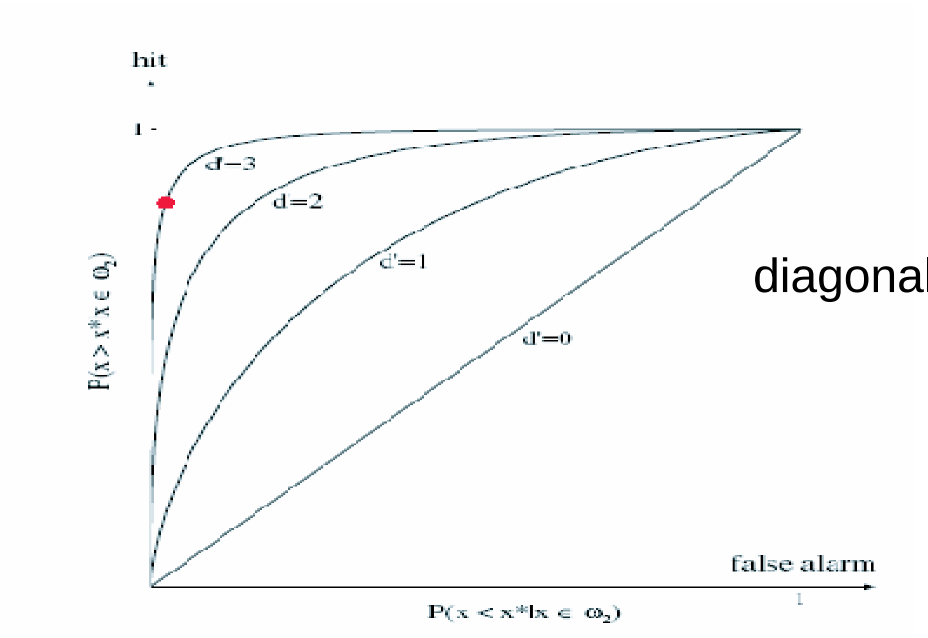
$$p_{fn} = p(x \notin h | x\text{+ve}) = p(h(x) = 0 | y = 1)$$

ROC curves

As we vary the size of the rectangle, we can change the FP/FN rate.
A receiver operating curve (ROC) plots hit rate vs false alarm rate and measures the discriminability between +ve and -ve examples.

$$\begin{aligned} p_{hit} &= p(x \in h | x\text{+ve}) = p(h(x) = 1 | y = 1) \\ p_{fa} &= p(x \in h | x\text{-ve}) = p(h(x) = 1 | y = 0) \end{aligned}$$

upper left
= perfect
performance



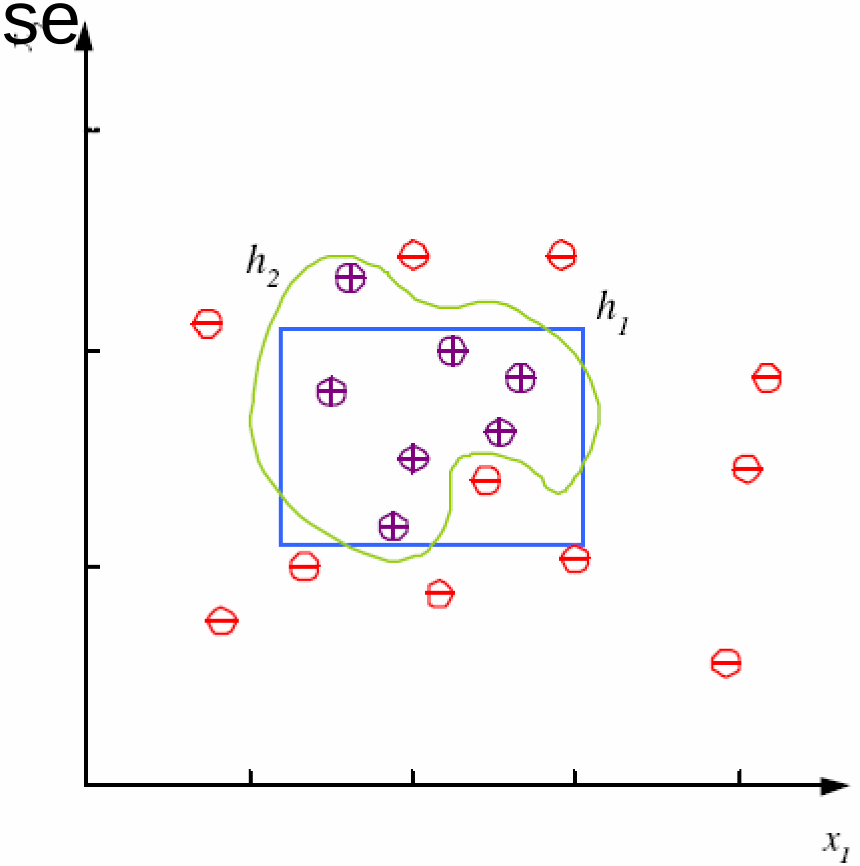
diagonal = chance level

Noise and Model Complexity

The true concept (green) may not be describable by a simple rectangle.

We may still prefer a simple rectangle hypothesis (blue) because

- Simpler to use (lower computational complexity)
- Easier to train (lower sample complexity)
- Easier to explain (more interpretable)
- Generalizes better



Model Selection & Generalization

- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about H
- Generalization: How well a model performs on new data
- Overfitting: H more complex than *C*
- Underfitting: H less complex than *C*
- Can use cross validation to estimate the generalization ability.

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  1. Complexity of H, *c* (H),
  2. Training set size, *N*,
  3. Generalization error, *Err*, on new data
- □ As *N*↑, *Err*↓
- □ As *c* (H)↑, first *Err*↓ and then *Err*↑