CHAPTER 2:

SUPERVİSED LEARNİNG

Learning a Class from Examples

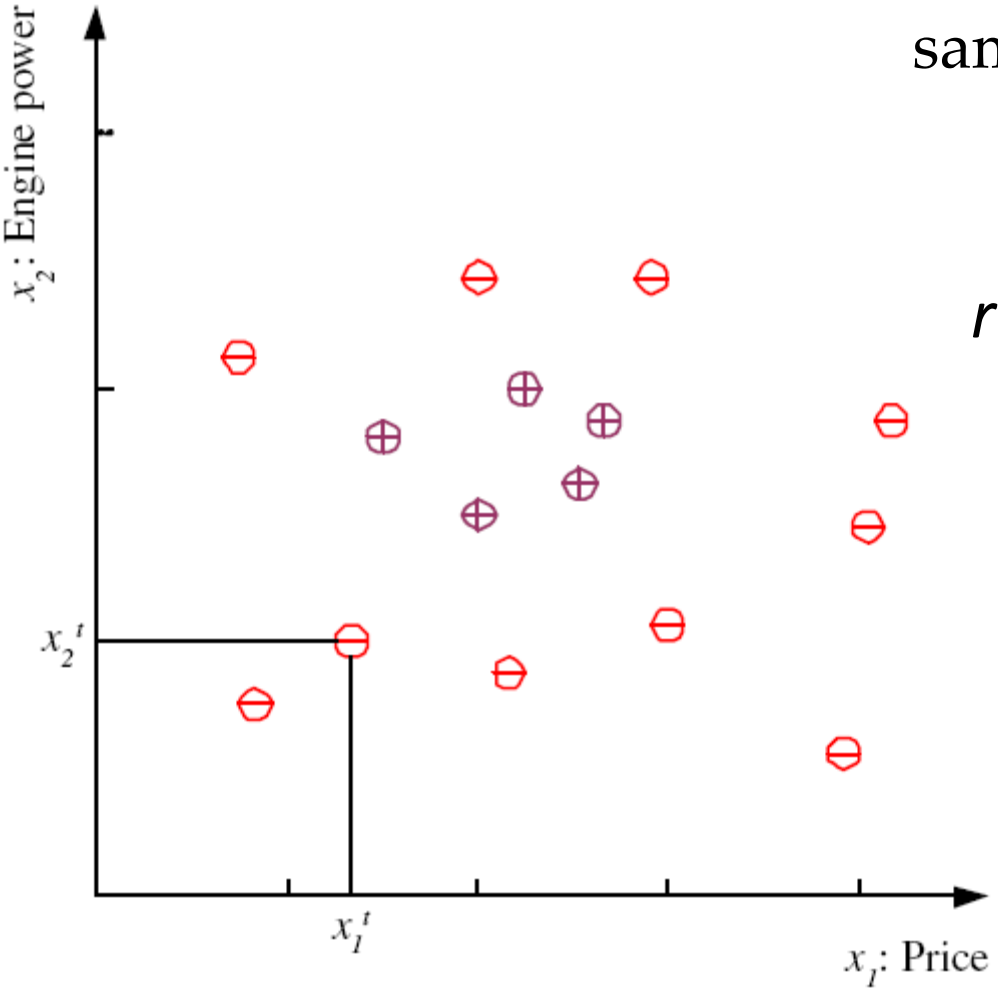
- Class C of a "family car"
  - Prediction: Is car *x* a family car?
  - Knowledge extraction: What do people expect from a family car?
- Output:

  Positive (+) and negative (–) examples

  Sometimes OK to say "doubt" (or abstain)
- Input representation:

  x_1: price, x_2 : engine power

Training set $\mathcal{X}$



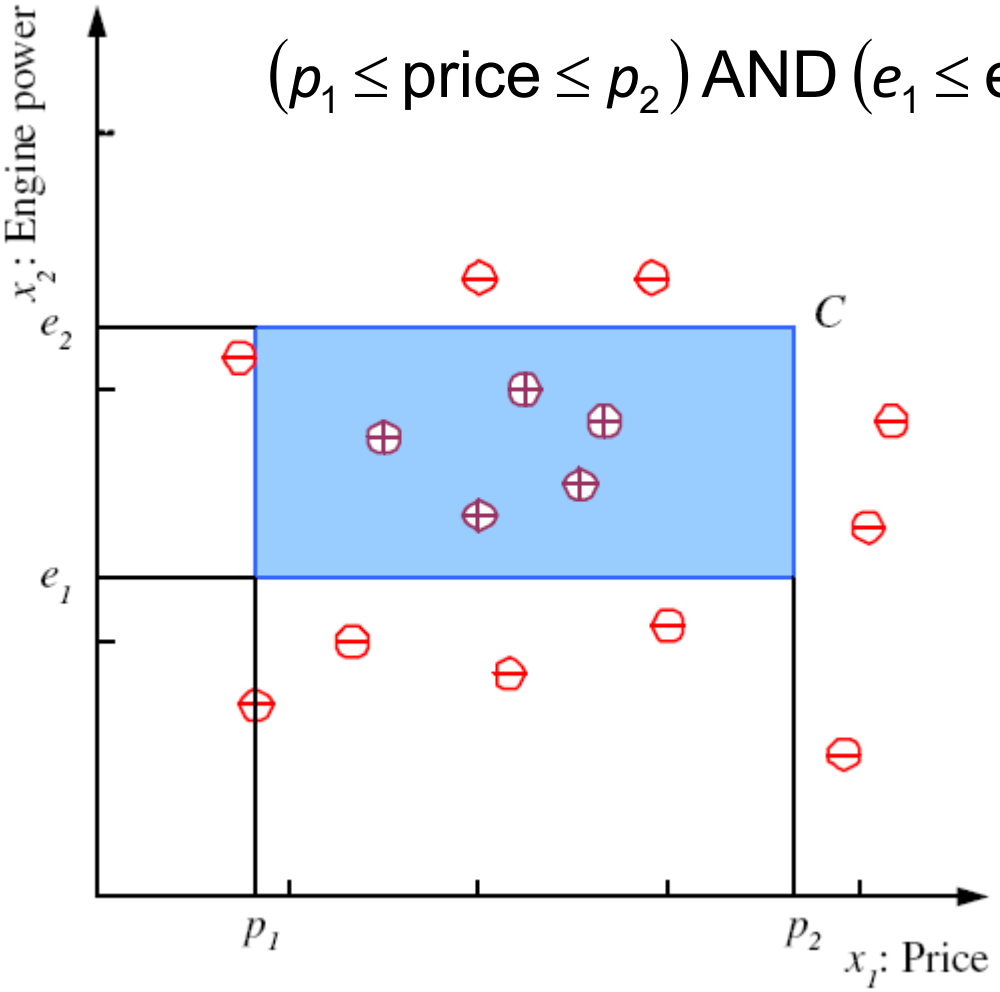
sample: $\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$

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

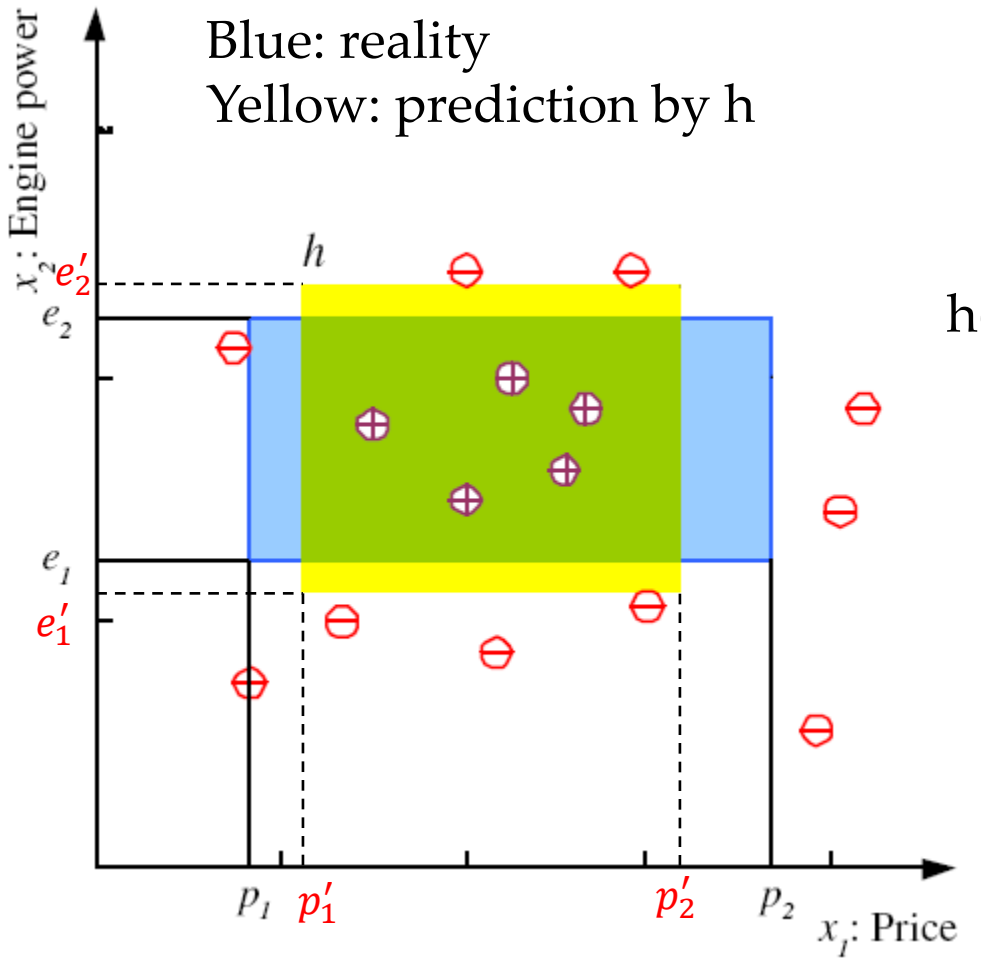
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Class C: the true class (concept)

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



Hypothesis class $\mathcal{H}$



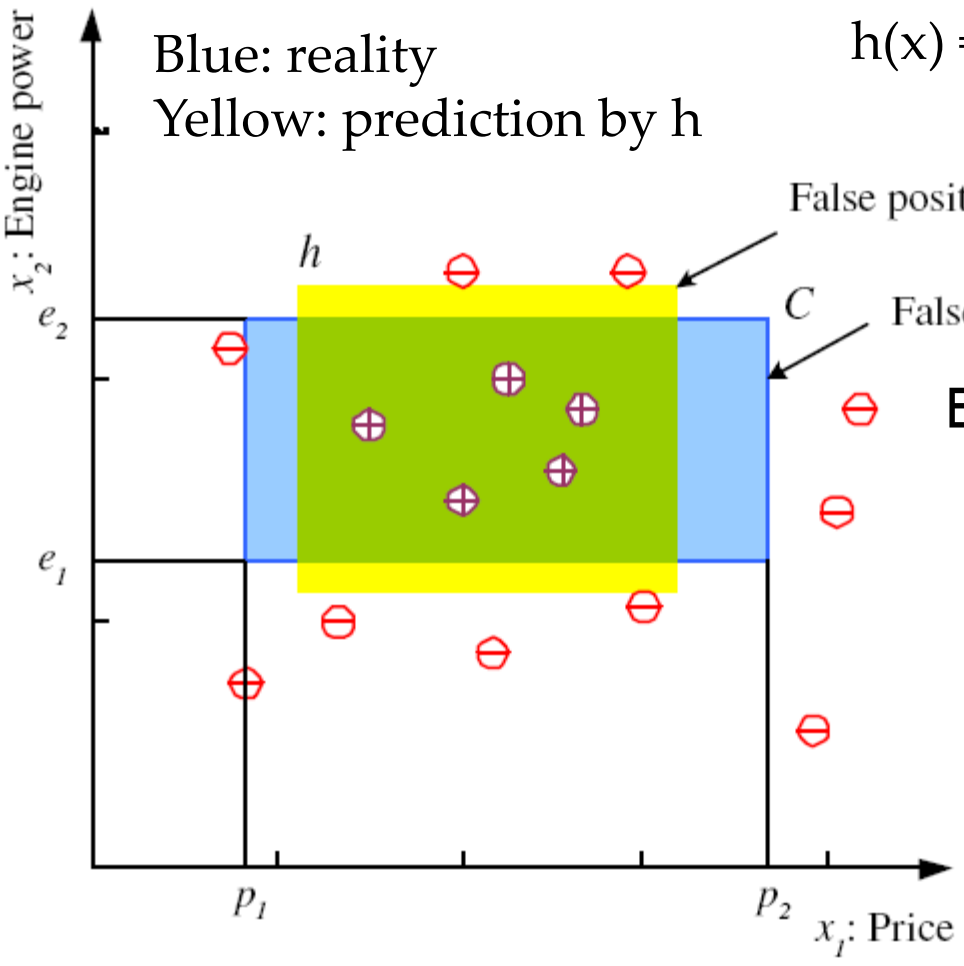
Fixing p_1', p_2', e_1', e_2',
we get a hypothesis h

$$h(x) = \begin{cases} 1 & \text{if x is in the yellow area} \\ 0 & \text{otherwise} \end{cases}$$

We get a **set of hypotheses**
if we vary p_1', p_2', e_1', e_2'

This is called a hypothesis
class $\mathcal{H}$
(not necessarily contains C,
flexibility)

Empirical error of a hypothesis



$$h(x) = \begin{cases} 1 & \text{if x is in the yellow area} \\ 0 & \text{otherwise} \end{cases}$$

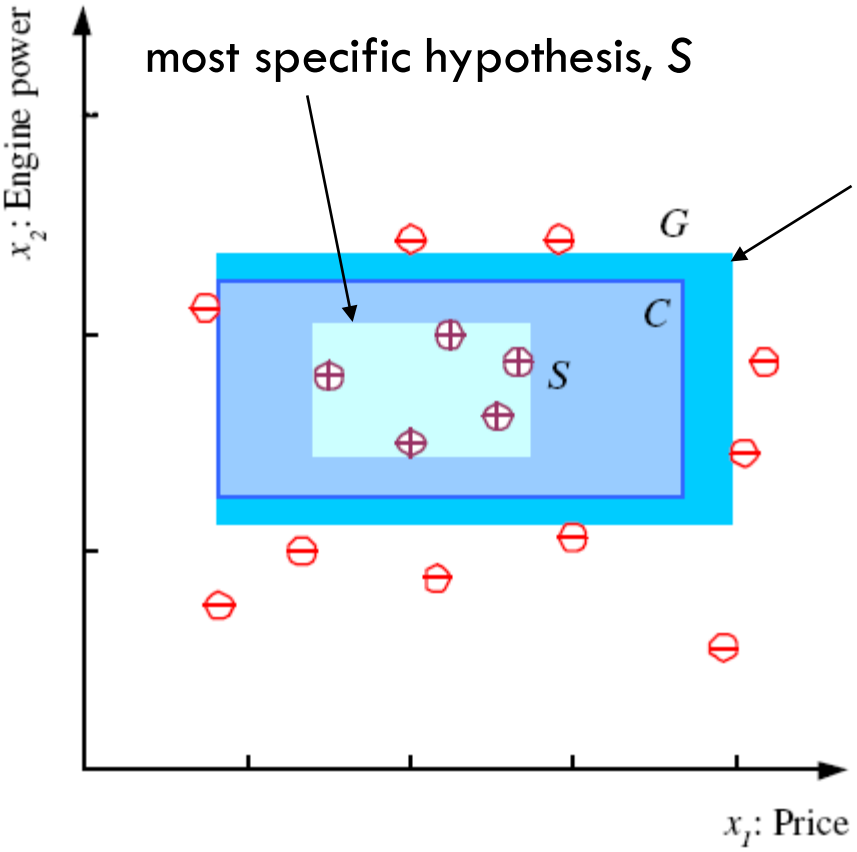
Empirical error of h

$$E(h \mid \mathcal{X}) = \sum_{t=1}^{N} 1\left(h\left(\mathbf{x}^t\right) \neq r^t\right)$$

seems like infinitely many h achieving 0 empirical error!

But differ in generalization

S, G, and the Version Space



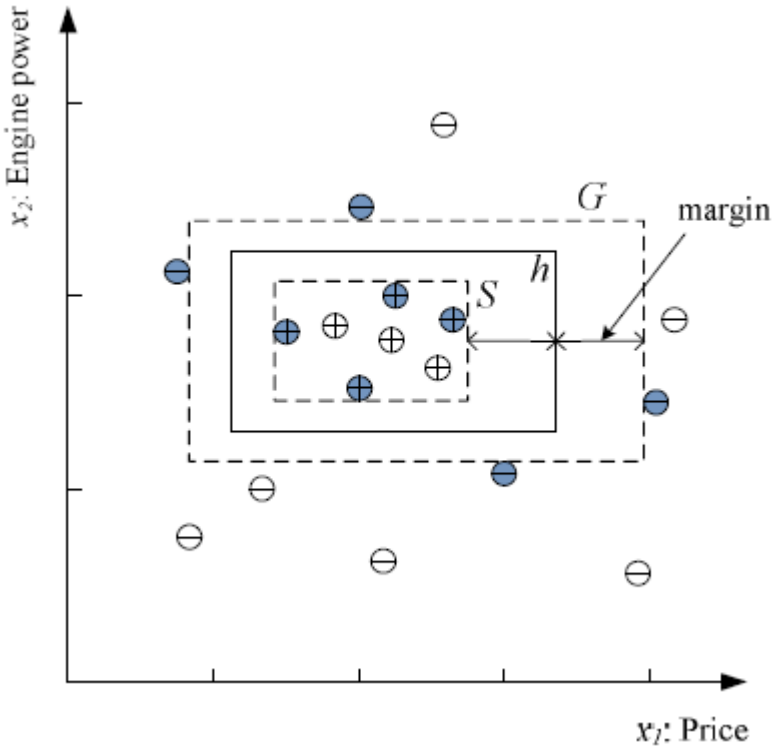
$h \in \mathcal{H}$, between S and G is **consistent** and make up the **version space** (Mitchell, 1997)

Change training set, change S, G, version space, and learned hypothesis.

Margin

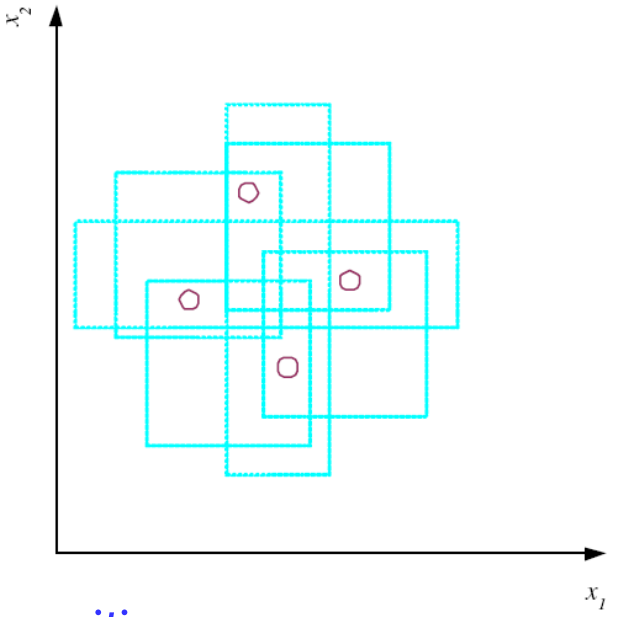
- Choose *h* with largest margin
  - Intuitive to choose *h* halfway between *S* and *G*
  - *Margin:* the distance between the boundary and the instances closest to it
  - Need a way to compute it

- But how to choose the hypothesis space $\mathcal{H}$ in the first place?



VC Dimension

- $\mathcal{H}$ needs to be large/flexible enough (how to quantify?)
- Just count the number of parameters? Deep learning.
- N points can be labeled in 2^N ways as +/–
- $\mathcal{H}$ shatters N if there exists $h \in \mathcal{H}$ consistent for any of these labeling
- VC($\mathcal{H}$) = the maximum such N
- Oblivious to data distribution
- The bigger $\mathcal{H}$ the better?



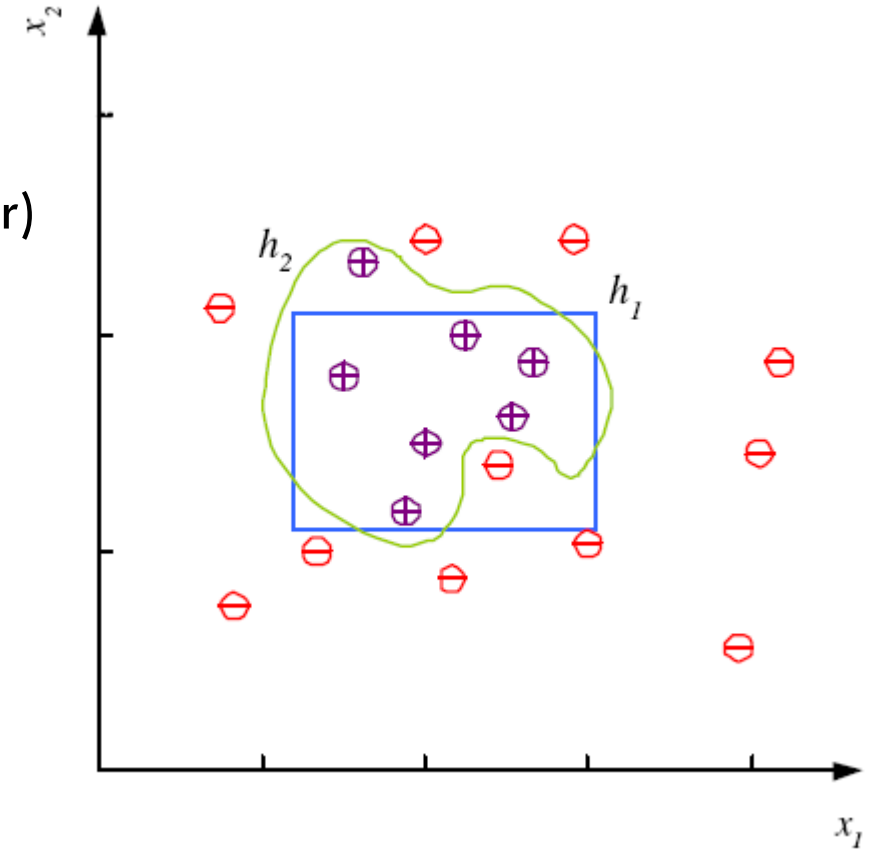
*An **axis-aligned** rectangle shatters 4 points only !*
*No need to shatter **any** four points.*
But no way to shatter 5 points regardless of their positions

Noise and Model Complexity

Noise: any unwanted anomaly in the data, e.g. imprecision in feature/label, hidden feature (XOR without X_2)

Use the simpler one because

- Generalizes better (less affected by single instances: Occam's razor)
- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)

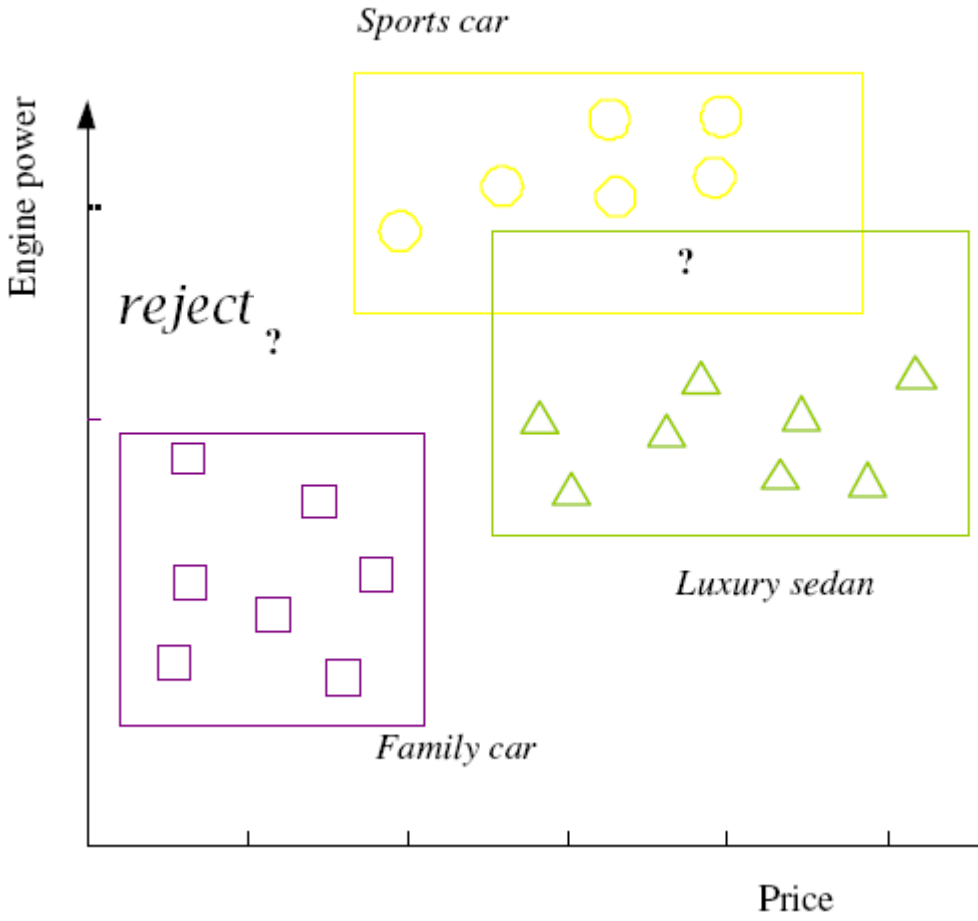


Multiple Classes, C_i $(i=1,\ldots,K)$

An input instance belongs to one and exactly one of them.
K two-class problems?

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

$$r_i^t = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \text{ (i.e., } \mathbf{x}^t \notin C_i) \end{cases}$$



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

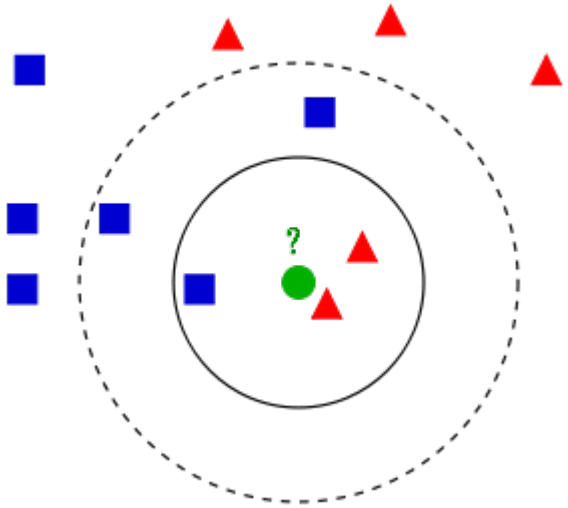
$$h_i(\mathbf{x}^t) = \begin{cases} 1 \text{ if } \mathbf{x}^t \in C_i \\ 0 \text{ if } \mathbf{x}^t \in C_j, j \neq i \end{cases}$$

$$E(\{h_i\}_{i=1}^K | \mathcal{X}) = \sum_{t=1}^N \sum_{i=1}^K 1(h_i(\mathbf{x}^t) \neq r_i^t)$$

still has some consistency issues

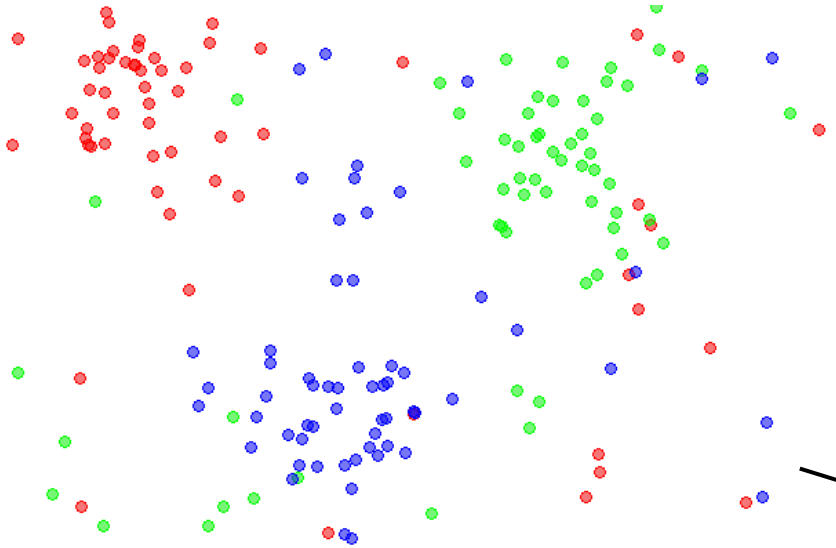
Nonparametric: k-nearest neighbor

- Given a new query point x
- Find the k training examples nearest to x
- Output the most frequent label in the k examples

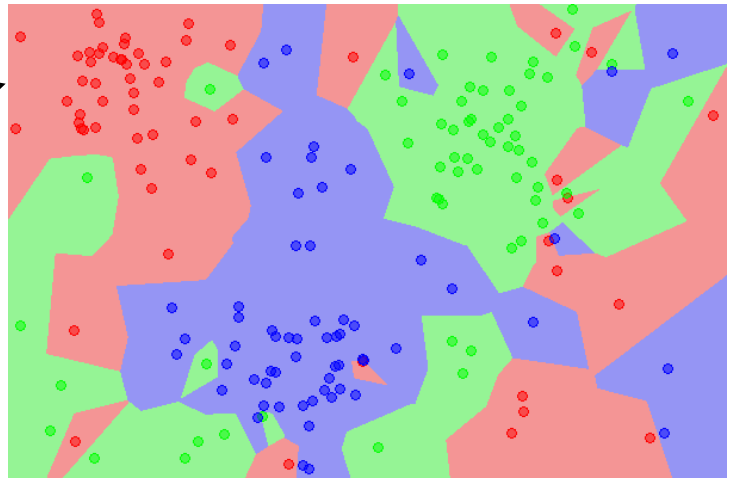


- k is a hyperparameter specified by user
- Trivially extends to multiple classes

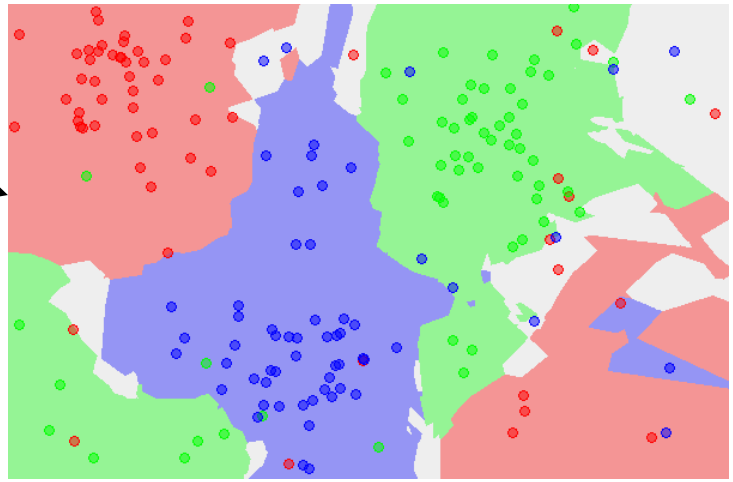
k-Nearest Neighbor Classifier



1-NN



5-NN



fewer small islands
(white for tie)

k-Nearest Neighbor Classifier Code


```
def get_neighbors(training_set,   # n training examples
                  labels,         # label of the n training examples
                  test_instance):

    n = length(training_set)
    distance = [0, 0, …, 0]   # n zeros
    for i = 1 … n
        distance(i) = Euclidean_distance(test_instance, training_set[i])

    index = numpy.argsort(distance)

    # compute the most frequent label among the k-nearest neighbors
    count = [0, 0]     # for two classes. Suppose labels can be either 0 or 1
    for j = 1 … k   (or from 0 to k – 1 if you use Python)
        l = labels[ index[ j ] ]
        count[l] += 1
    return 0 if count[0] > count[1] else 1
```

numpy.array([2, 3, -1, 0])
returns [2, 3, 0, 1]

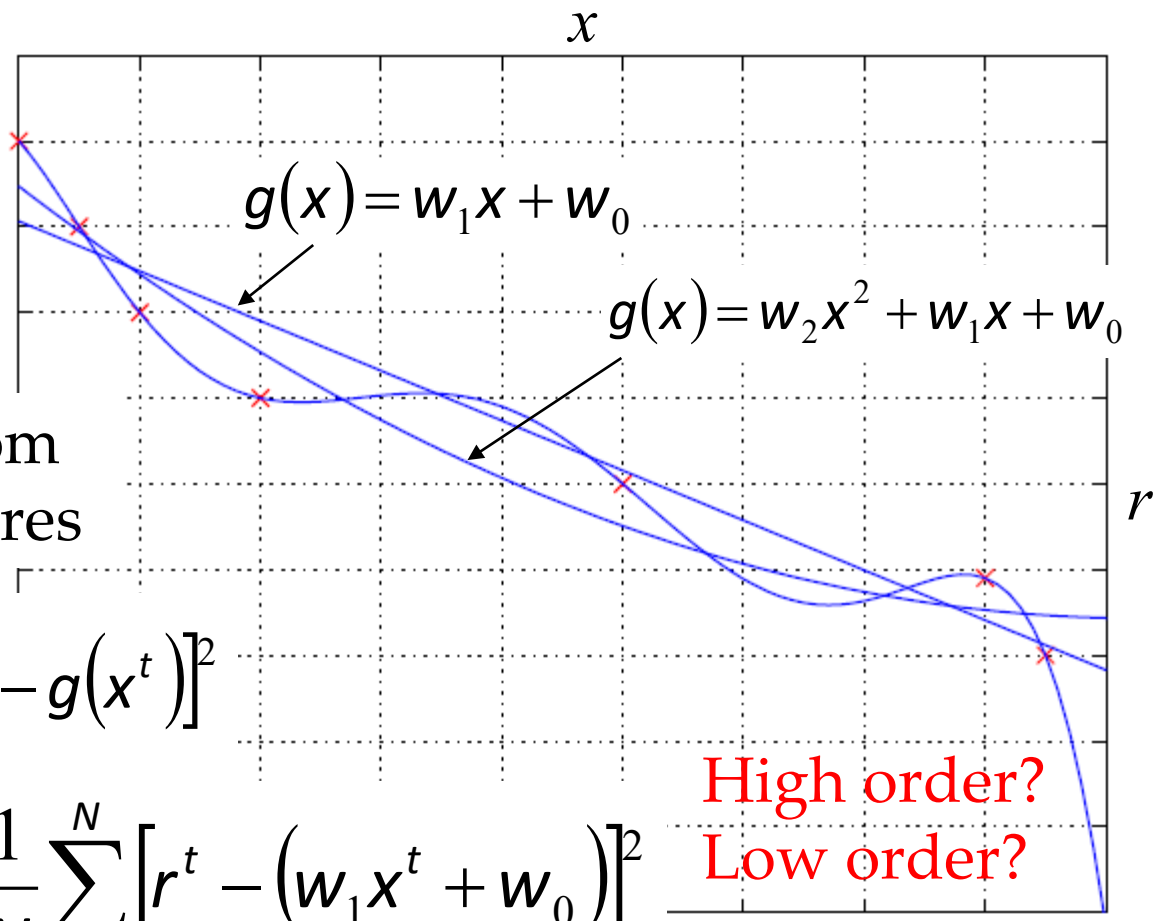
Regression (numeric value)

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

$$r^t \in \Re$$

$$r^t = f\left(x^t\right) + \varepsilon$$

noise possibly from unobserved features



$$E(g \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\left[r^t - g\left(x^t\right)\right]^2$$

$$E(w_1, w_0 \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\left[r^t - \left(w_1 x^t + w_0\right)\right]^2$$

Model Selection & Generalization

- Learning is an ill-posed problem; as data is not sufficient to find a unique solution

x_1	x_2	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}	h_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

  - Each distinct training example removes half the hypotheses
- So we need inductive bias, i.e., assumptions about $\mathcal{H}$
  - Rectangles, linear function, max margin, most specific
- Generalization: How well a model predicts on new data
- Underfitting: $\mathcal{H}$ less complex than C or *f*
- Overfitting: $\mathcal{H}$ more complex than C or *f* *(noise)*

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  1. Complexity of $\mathcal{H}$: c ($\mathcal{H}$)
  2. Training set size: N
  3. Generalization error, E, on new data
- As $N \uparrow$, $E \downarrow$ then plateaus
  - If c($\mathcal{H}$) is too high, E will be kept in check only up to a point
- As c ($\mathcal{H}$) $\uparrow$, first $E \downarrow$ and then $E \uparrow$

Cross-validation

- Measure the quality of a given inductive bias
  - To estimate generalization error, we need data unseen during training. We split the dataset as
    - Training set (50%)
    - Validation set (50%), report the error on it
- Measure the quality of the best inductive bias from a given set of inductive biases
  - Split the dataset as
    - Training set (50%)
    - Validation set (25%)

    } use them to select the best inductive bias
    - Test set (25%): apply the best inductive bias to it and report error
- Resampling when there is few data

Dimensions of a Supervised Learner

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

1. **Model**: $g(\mathbf{x} \mid \theta)$

   g: hypothesis space $\mathcal{H}$ (rectangles), θ: parameters (four corners)

   Should be large enough, have enough capacity, to include the unknown function that generated the data represented in $\mathbf{x}$ (with noise)

2. **Loss function**:

$$E(\theta \mid \mathcal{X}) = \sum_t L\left(r^t, g\left(\mathbf{x}^t \mid \theta\right)\right)$$

   There should be enough training data to allow us to pinpoint the correct (or a good enough) hypothesis

3. **Optimization procedure**:

$$\theta^* = \arg\min_\theta E(\theta \mid \mathcal{X})$$