Machine Learning

Amirkabir University of Technology
(Tehran Polytechnic)

Lecture 2.
Supervised learning
kNN

Alireza Rezvanian

Fall 2023

Last update: Oct. 19, 2022

Outline

- Supervised learning
- Basic Steps of Supervised Learning
- Cross-Validation
- Instance-based Learning
- k-Nearest Neighbors
- Norms
- Lazy vs Eager Learning

Supervised Learning

- Given
  - examples of a function (x, f(x))
- Predict function f(x) for new examples x
  - Discrete f(x): Classification
  - Continuous f(x): Regression
  - f(x) = Probability(x): Probability estimation

Supervised Learning

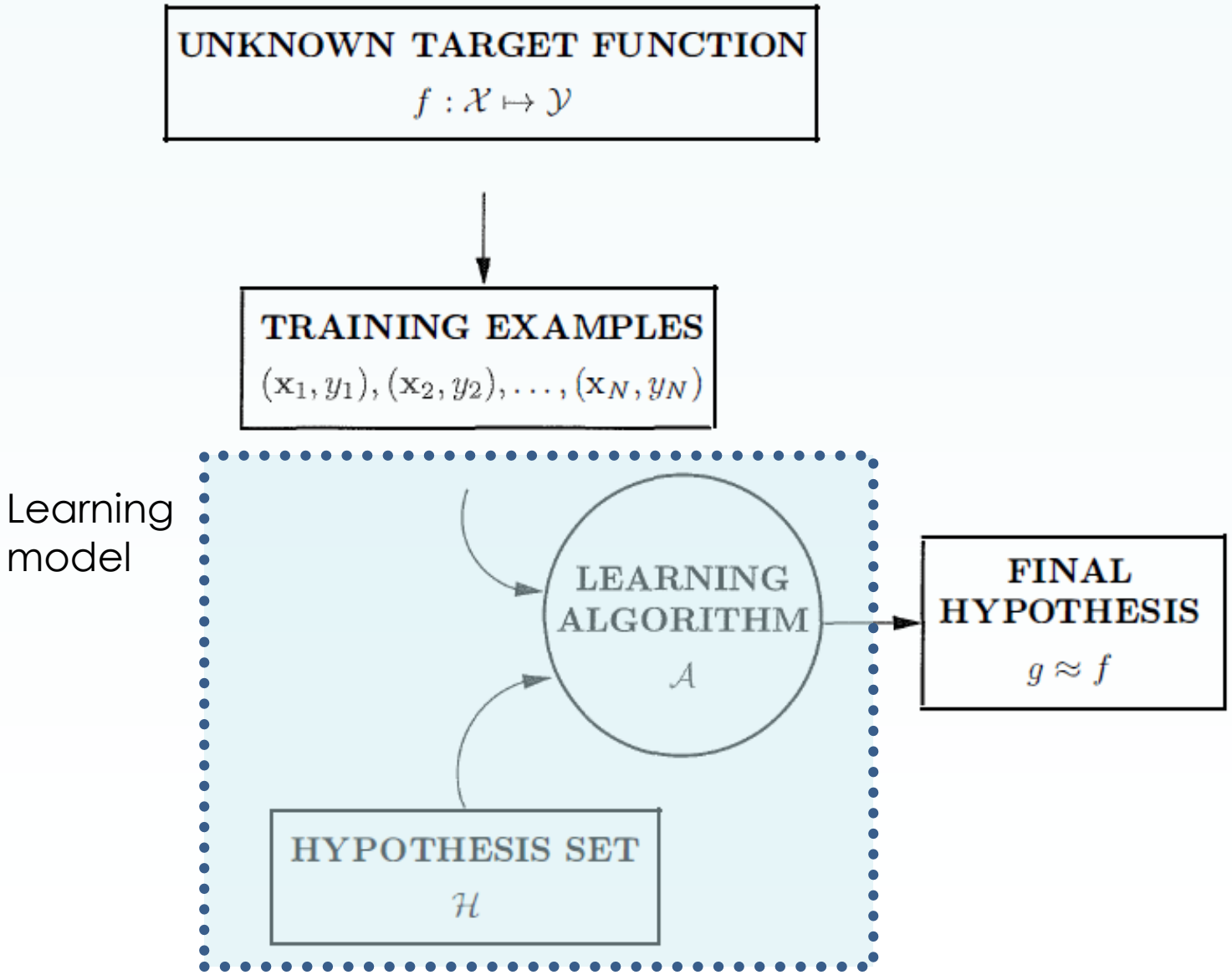
- Input: x (images, text, emails…)
- Output: y (spam or non-spam…)
- (Unknown) Target Function
  - f: X → Y (the "true" mapping / reality)
- Data
  - (x_1,y_1), (x_2,y_2), …, (x_N,y_N)
- Model / Hypothesis Class
  - g: X → Y
  - $y = g(x) = \text{sign}(w^T x)$
- Learning = Search in hypothesis space
  - Find best g in model class.

General concepts

- Hypothesis
  - The hypothesis is noted h_θ and is the model that we choose. For a given input data x^i the model prediction output is $h_\theta(x^i)$
- Loss function
  - A loss function is a function $L: (z, y) \in \Re \times Y \rightarrow L(z, y) \in \Re$ that takes as inputs the predicted value z corresponding to the real data value y and outputs how different they are.
- Cost function (Objective function)
  - The cost function J is commonly used to assess the performance of a model, and is defined with the loss function $\bar{L}$ as follows:

$$J(\theta) = \sum_{i=1}^{m} L\left(h_\theta(x^i, y^i)\right)$$

Supervised Learning



Basic Steps of Supervised Learning

- **Set up** a supervised learning problem
- **Data collection**
  - Start with training data for which we know the correct outcome provided by a teacher or oracle.
- **Representation**
  - Choose how to represent the data.
- **Modeling**
  - Choose a hypothesis class: $H = \{g: X \rightarrow Y\}$
- **Learning/Estimation**
  - Find best hypothesis you can in the chosen class.
- **Model Selection**
  - Try different models. Picks the best one. (More on this later)
- If happy stop
  - Else refine one or more of the above

The Design Cycle in supervised learning

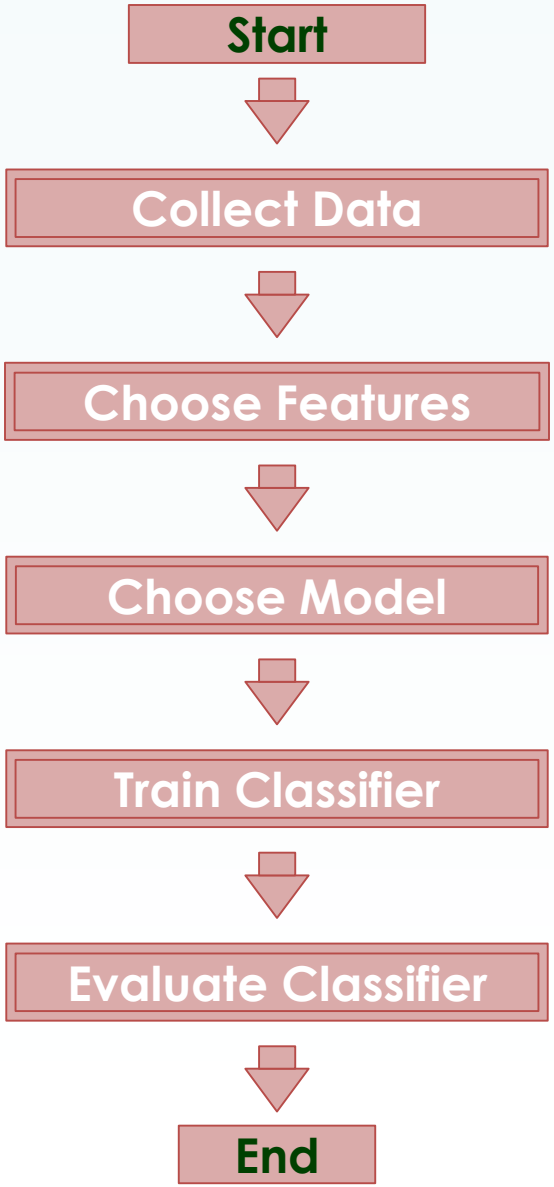
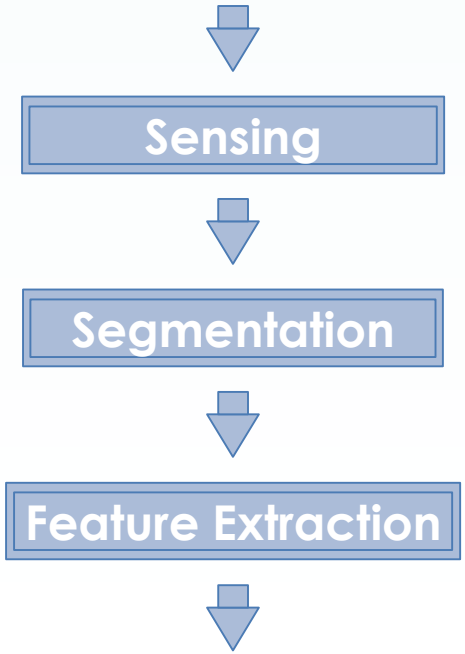
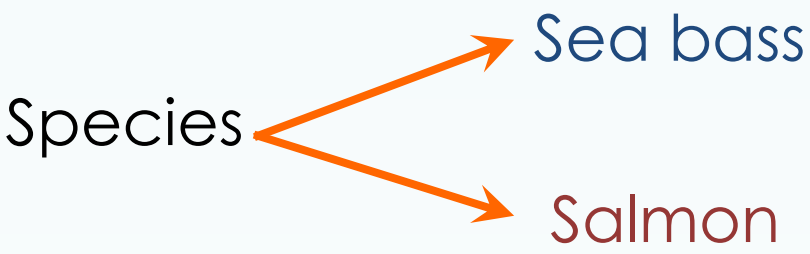


Image Processing Example

- **Sorting Fish:** incoming fish are sorted according to species using optical sensing (sea bass or salmon?)
- Problem Analysis:
  - set up a camera and take some sample images to extract features
  - Consider features such as length, lightness, width, number and shape of fins, position of mouth, etc.



Sorting Fish

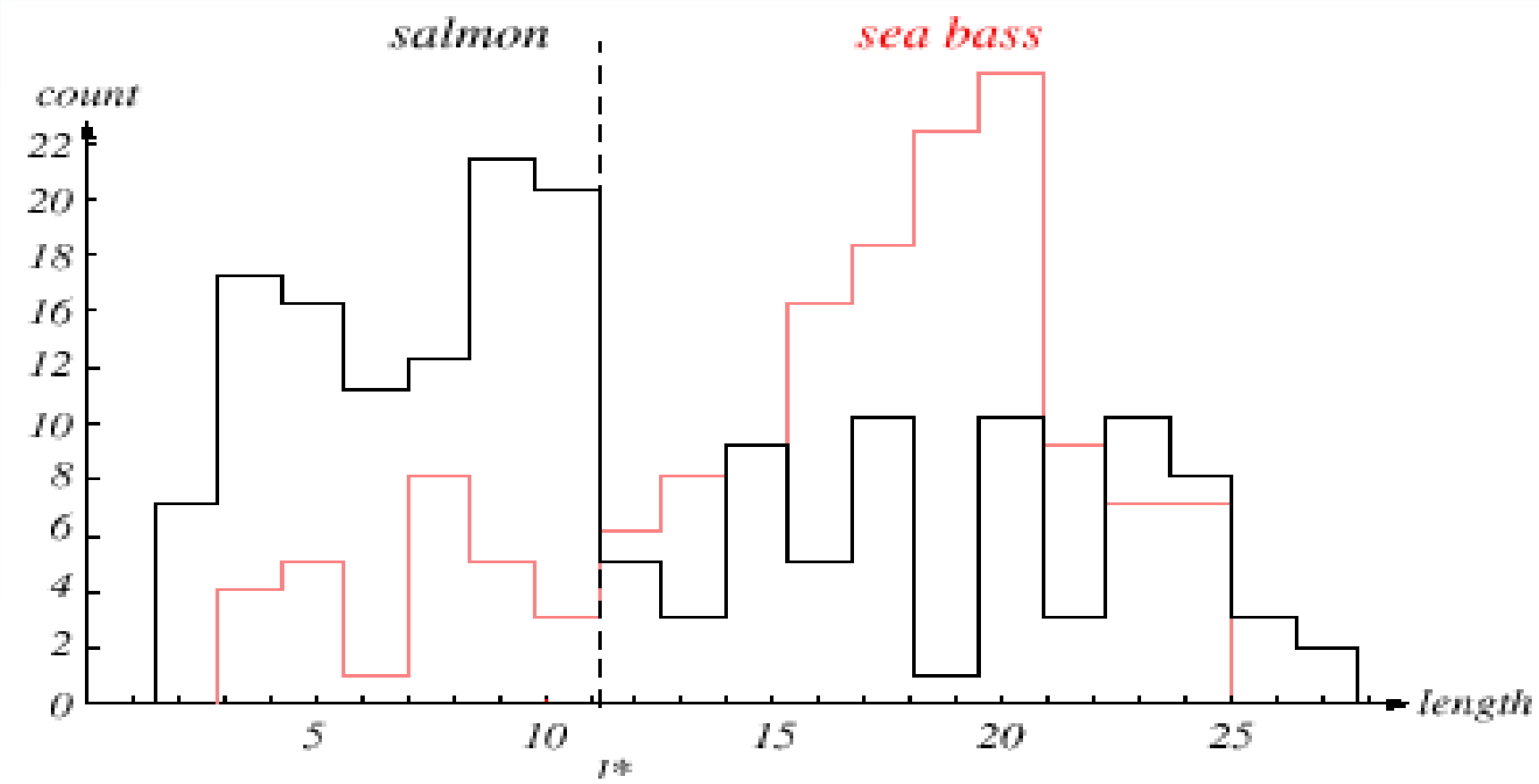


- Set up a camera and take some sample images to extract features
  - Length
  - Lightness
  - Width
  - Number and shape of fins
  - Position of the mouth, etc…

  - This is the set of all suggested features to explore for use in our classifier!

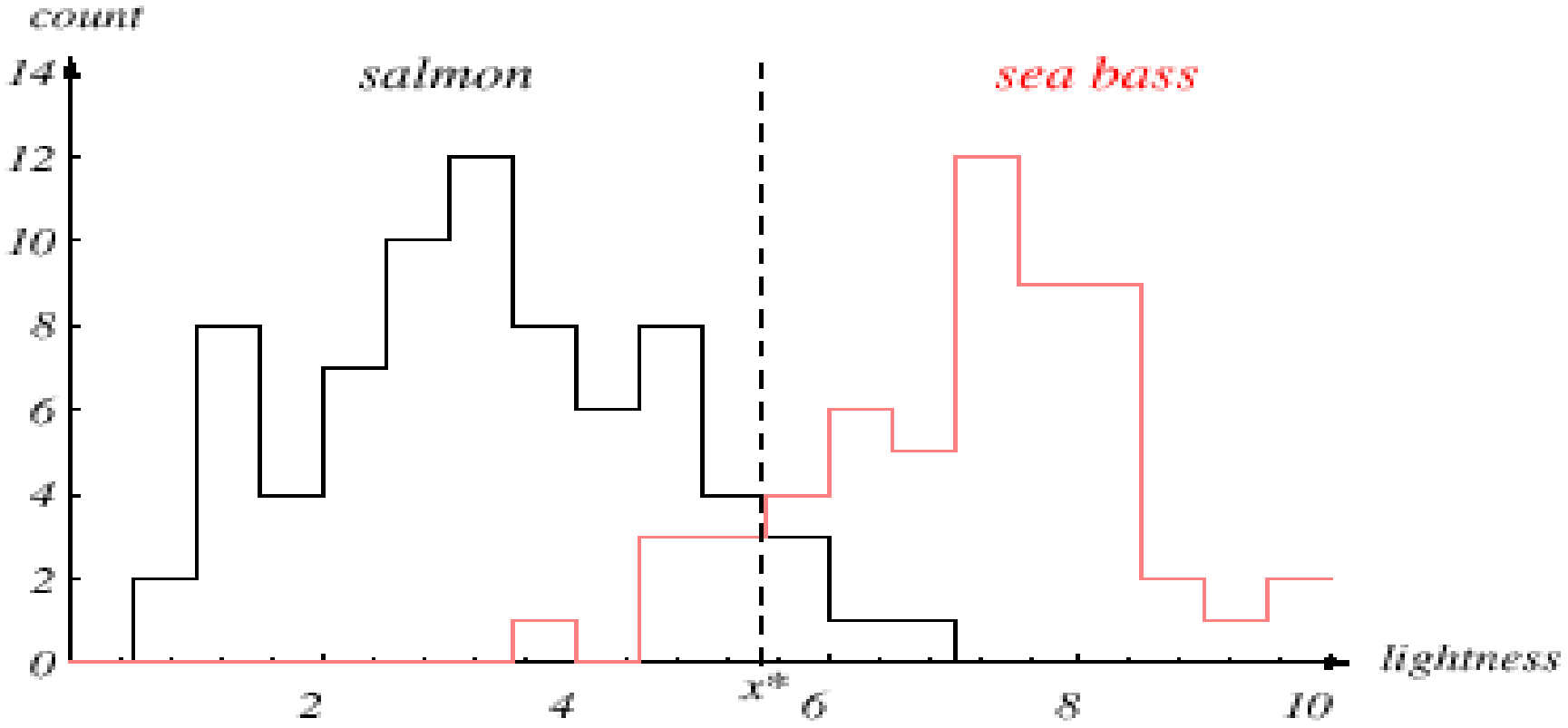
Length As A Discriminator

- Select the length of the fish as a possible feature for discrimination
- Distribution of the length of the fish



- **Conclusion: Length is a poor discriminator**

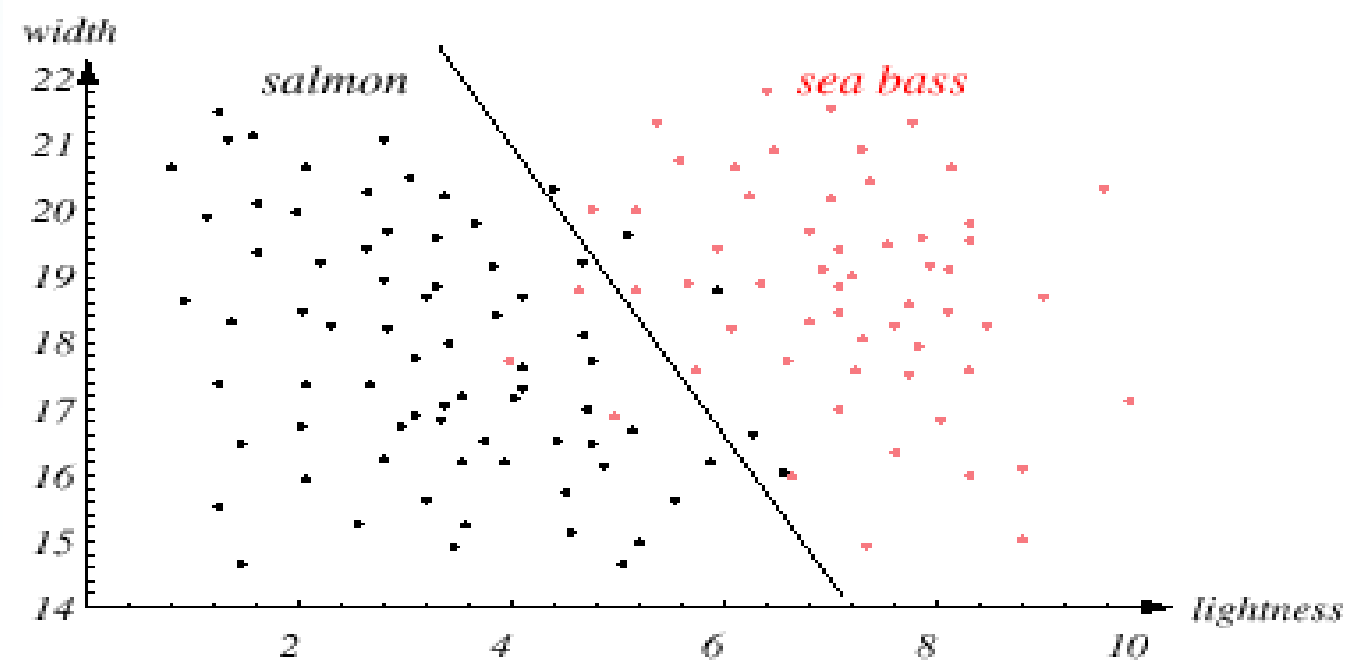
Add Another Feature



- Lightness is a better feature than length because it reduces the misclassification error.
- Can we combine features in such a way that we improve performance?
  (Hint: correlation)

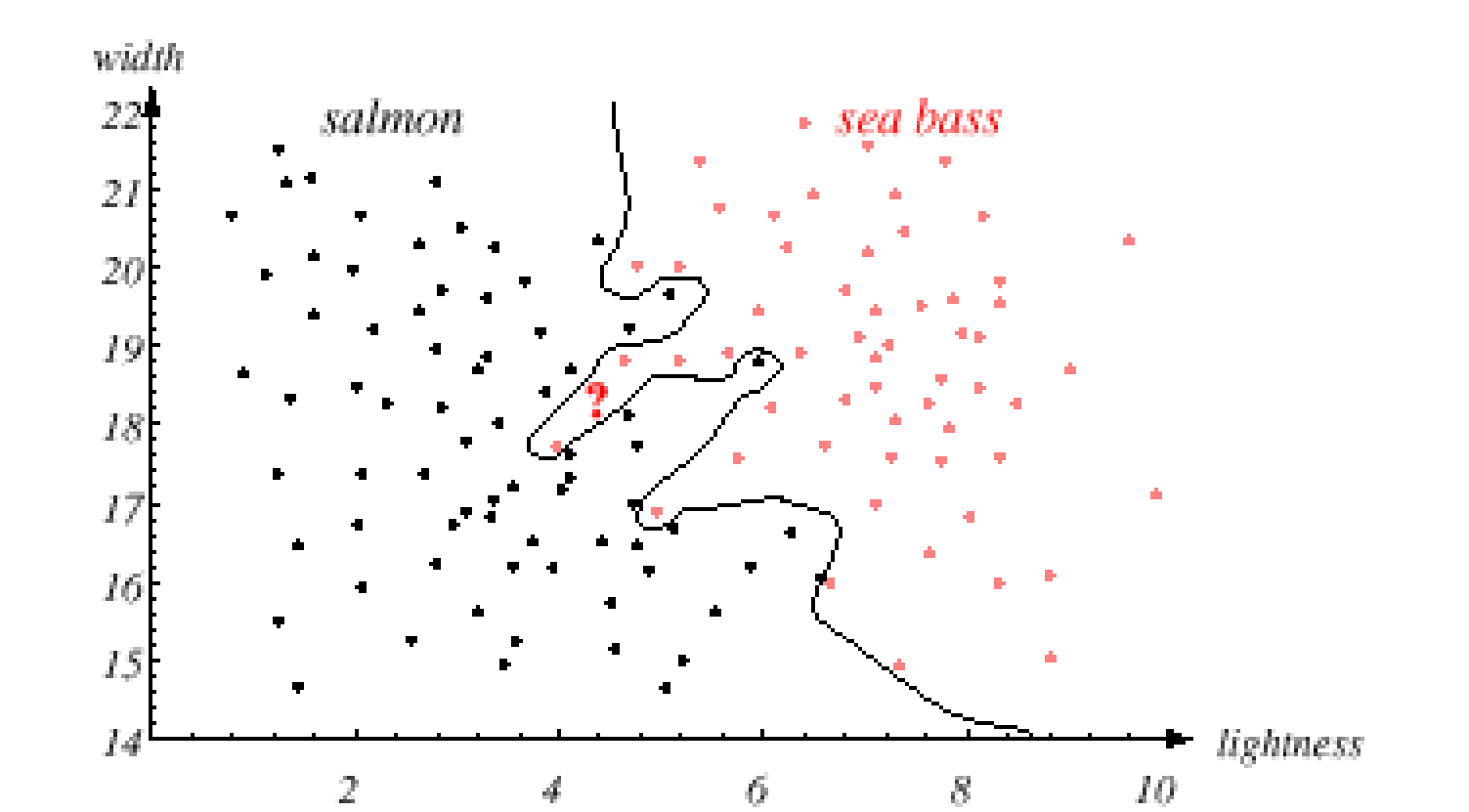
Width And Lightness

- Treat features as a N-tuple (two-dimensional vector)
- Create a scatter plot
- Draw a line (regression) separating the two classes



Decision Theory

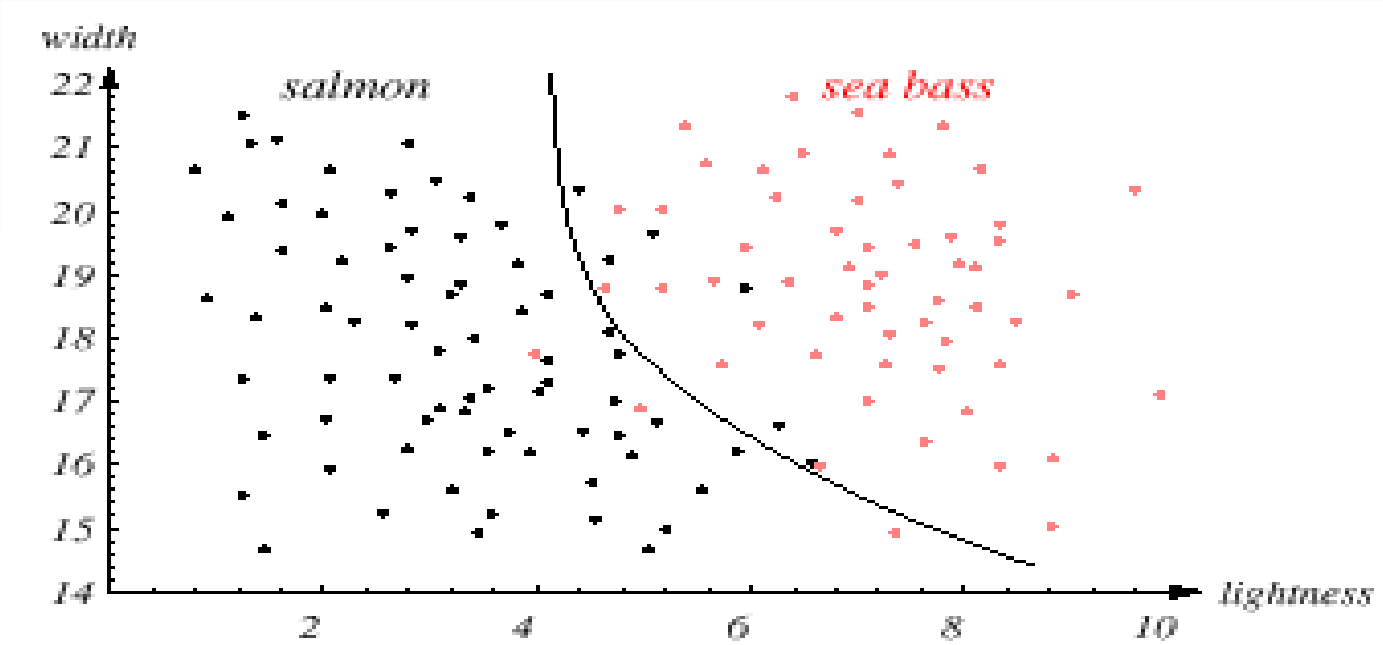
- Can we do better than a linear classifier?



- **What is wrong with this decision surface? (Hint: generalization)**

Generalization and Risk

- Why might a smoother decision surface be a better choice?
  (Hint: Occam's Razor).



- **This course investigates how to find such "optimal" decision surfaces and how to provide system designers with the tools to make intelligent trade-offs.**

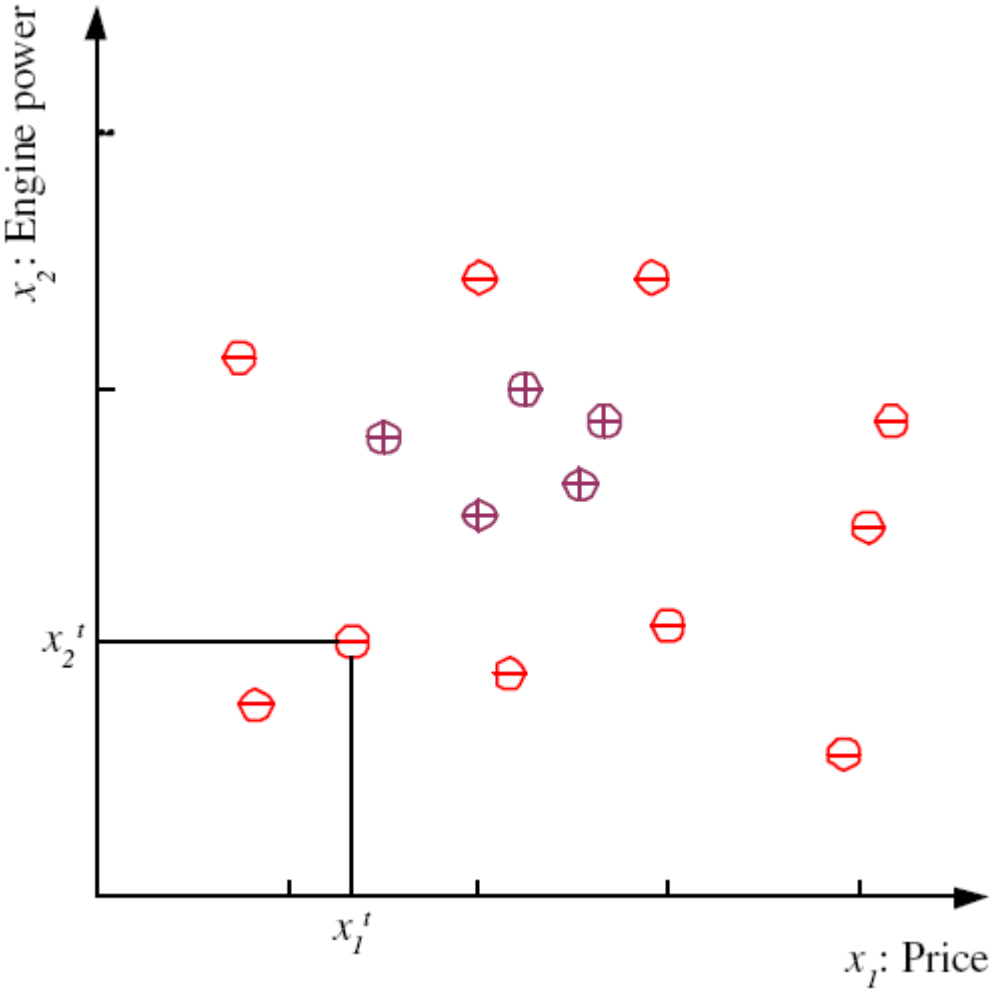
Learning a Class from Examples

- 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
- **Class learning** is finding a description that is shared by all positive examples and none of the negative examples.

Training set $\mathcal{X}$



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

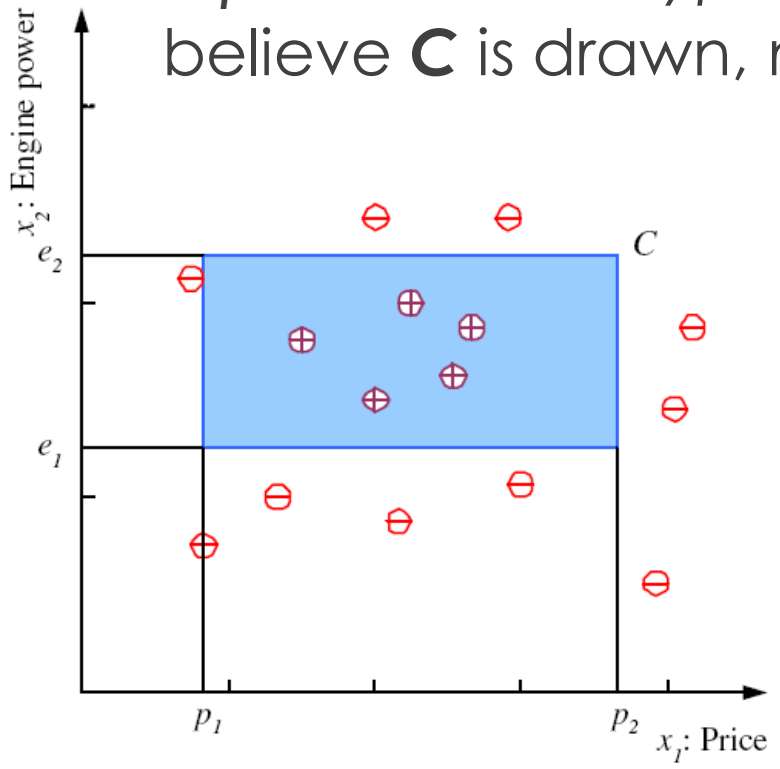
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

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

Class C

$$H_1: (p_1 \leq \text{price} \leq p_2) \text{ AND } (e_1 \leq \text{engine power} \leq e_2)$$

- H_1 fixes $\mathcal{H}$, the *hypothesis class* from which we believe **C** is drawn, namely, the set of rectangles.



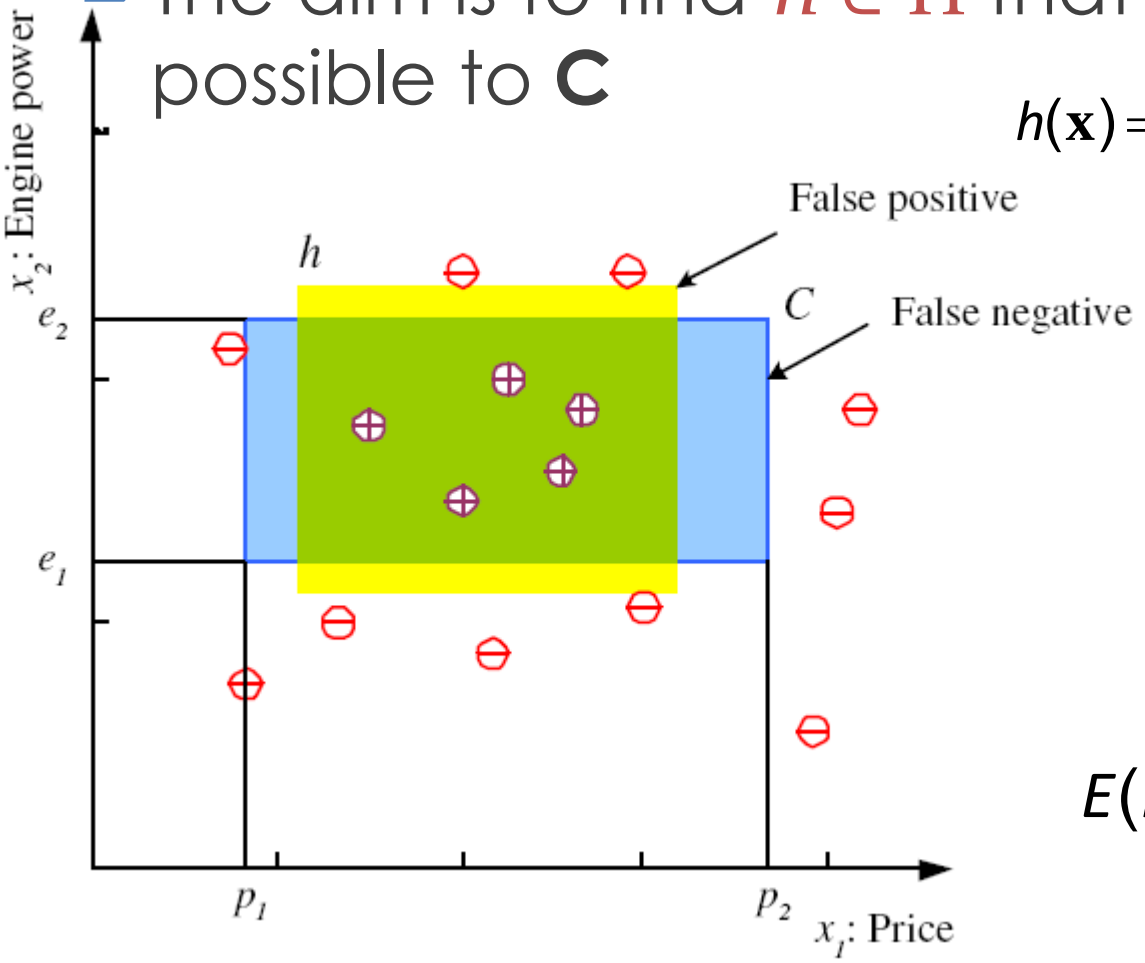
The **learning algorithm** then finds ***hypothesis*** the particular *hypothesis*, $h \in \mathrm{H}$, specified by a particular quadruple of $(p_1^h, p_2^h, p_1^h, p_2^h)$, to approximate **C** as closely as possible.

C: the actual class
h: our induced hypothesis

Hypothesis class $\mathcal{H}$

- The aim is to find $h \in \mathrm{H}$ that is as similar as possible to **C**

$$h(\mathbf{x}) = \begin{cases} 1 \text{ if } h \text{ says } \mathbf{x} \text{ is positive} \\ 0 \text{ if } h \text{ says } \mathbf{x} \text{ is negative} \end{cases}$$



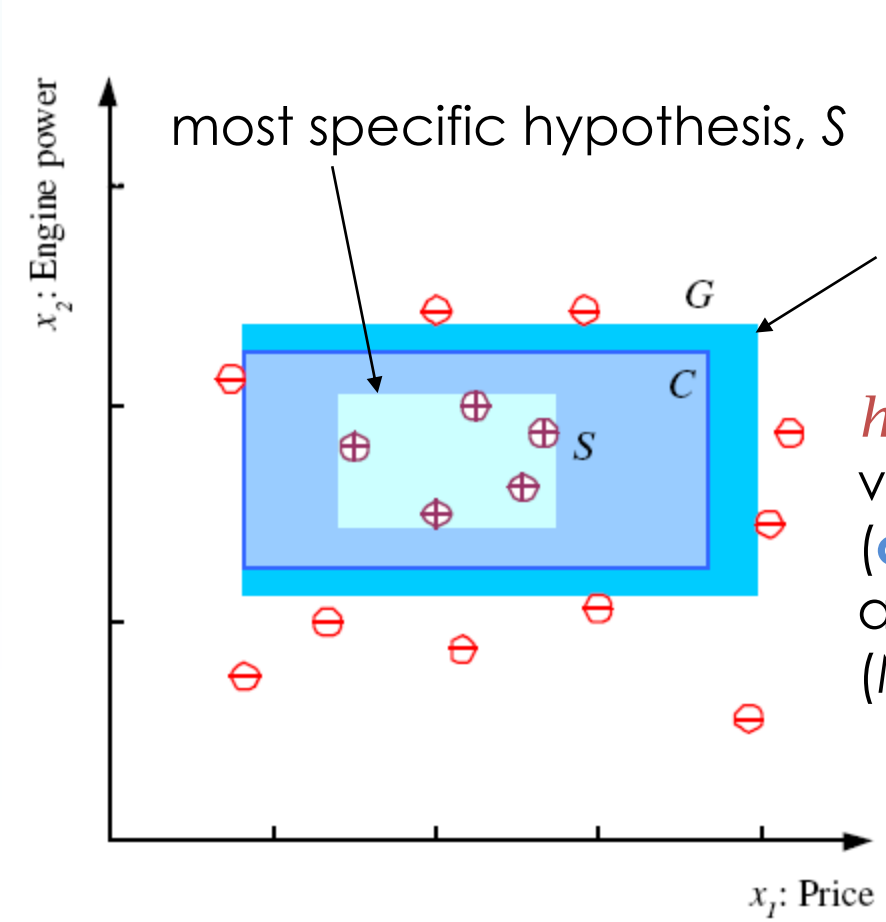
Error of h on $\mathcal{H}$

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

Empirical error (training error) is the proportion of training instances where *predictions* of $\boldsymbol{h}$ do not match the *required values* given in **X**.

Generalization, and the Version Space

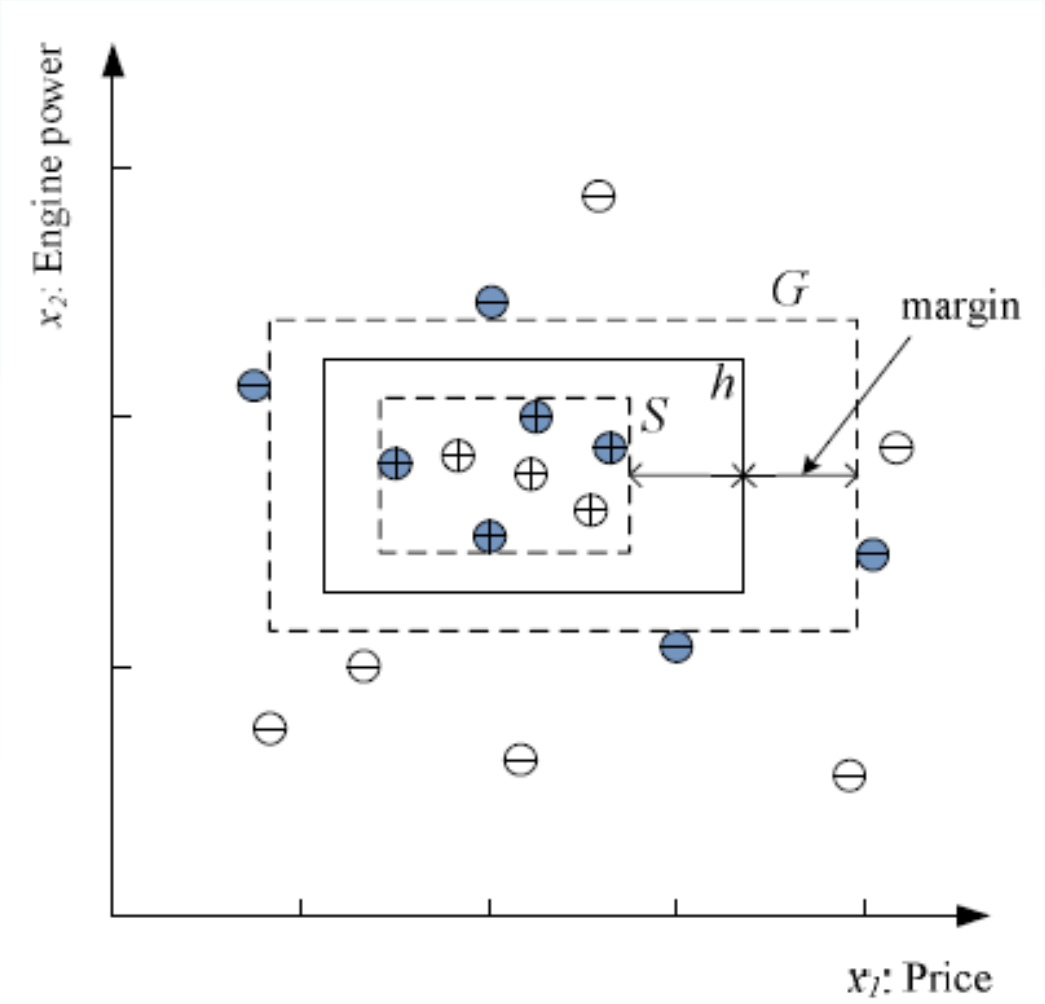
- **Generalization**, that is, how well our *hypothesis* will correctly classify future examples that are not part of the training set.



$h \in \mathrm{H}$, between ***S*** and ***G*** is a valid hypothesis with no error (**consistent** with the training set) and make up the **version space** (Mitchell, 1997)

Margin

- Choose h with largest margin (minimize risk) for best separation

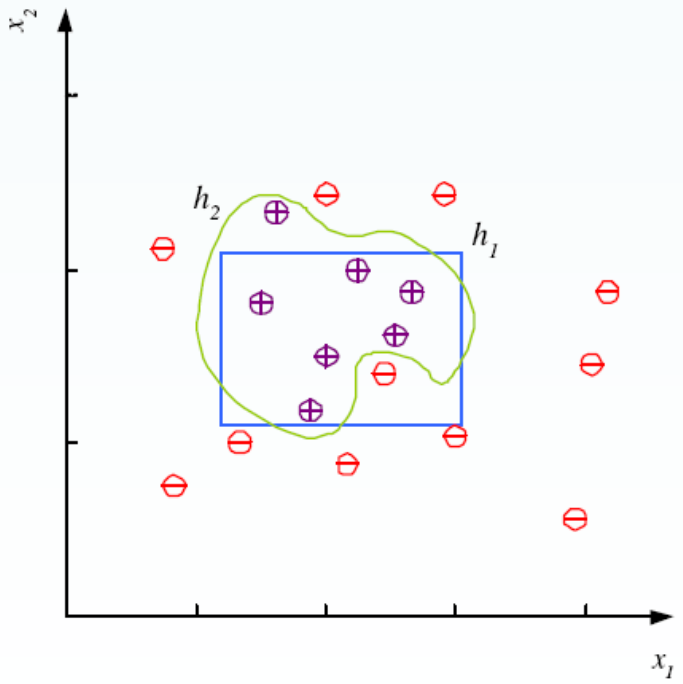


the ***margin***, is the distance between the boundary and the instances closest to it

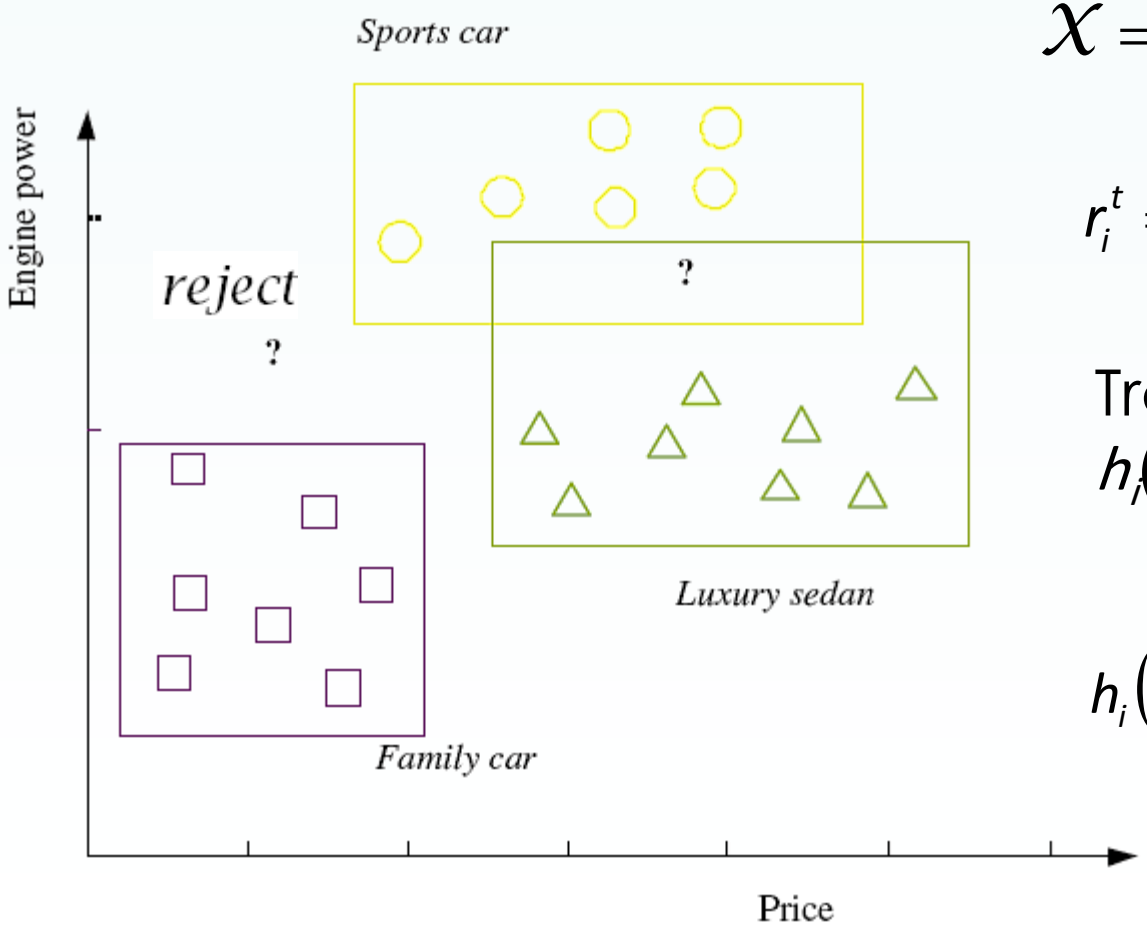
Noise and Model Complexity

Use the simpler one because

- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better (lower variance - Occam's razor)



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



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

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

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

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

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 $\mathcal{H}$
- Generalization: How well a model performs on new data
- Overfitting: $\mathcal{H}$ more complex than C or f
- Underfitting: $\mathcal{H}$ less complex than C or f

Dimensions of a Supervised Learner

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

2. Loss function: $E(\theta|\mathcal{X}) = \sum_t L(r^t, g(\mathbf{x}^t|\theta))$

3. Optimization procedure:

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

Loss/Error Functions

- How do we measure performance?
- Regression:
  - L_2 error
- Classification:
  - #misclassifications
  - Weighted misclassification via a cost matrix
  - For 2-class classification:
    - True Positive, False Positive, True Negative, False Negative
  - For k-class classification:
    - Confusion Matrix

Training vs Testing

- What do we want?
  - Good performance (low loss) on training data?
  - No, Good performance on *unseen test data!*

- Training Data:
  - $\{ (x_1,y_1), (x_2,y_2), \ldots, (x_N,y_N) \}$
  - Given to us for learning f

- Testing Data
  - $\{ x_1, x_2, \ldots, x_M \}$
  - Used to see if we have learnt anything

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (60%)
  - Validation set (10%)
  - Test (publication) set (30%)
- Resampling when there is few data

Procedural View

- Training Stage:
  - Raw Data → x (Feature Extraction)
  - Training Data { (x,y) } → f (Learning)
- Testing Stage
  - Raw Data → x (Feature Extraction)
  - Test Data x → f(x) (Apply function, Evaluate error)

Concepts

- Capacity
  - Measure how large hypothesis class H is.
  - Are all functions allowed?
- Overfitting
  - f works well on training data
  - Works poorly on test data
- Generalization
  - The ability to achieve low error on new test data

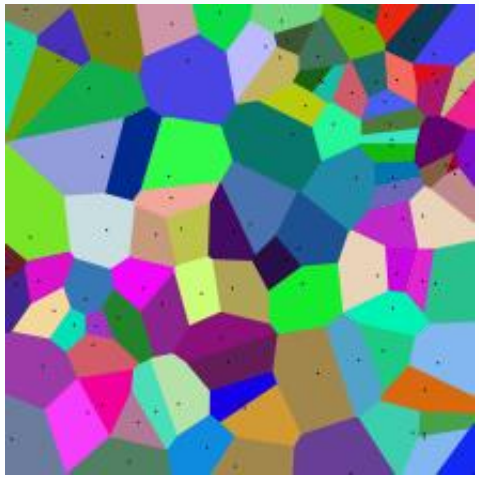
Guarantees

- 20 years of research in Learning Theory oversimplified:

- If you have:
  - Enough training data D
  - and H is not too complex
  - then *probably* we can generalize to unseen test data

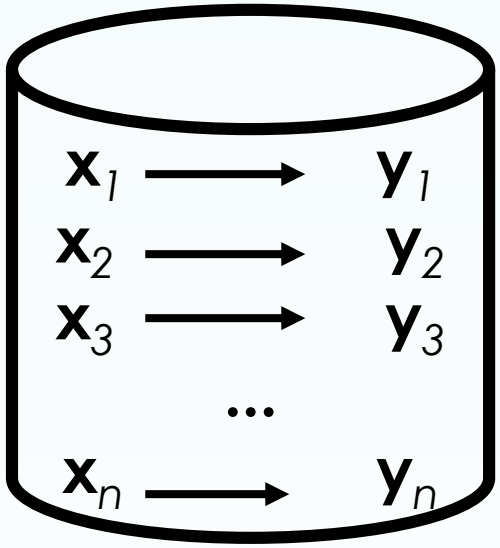
Nearest Neighbors

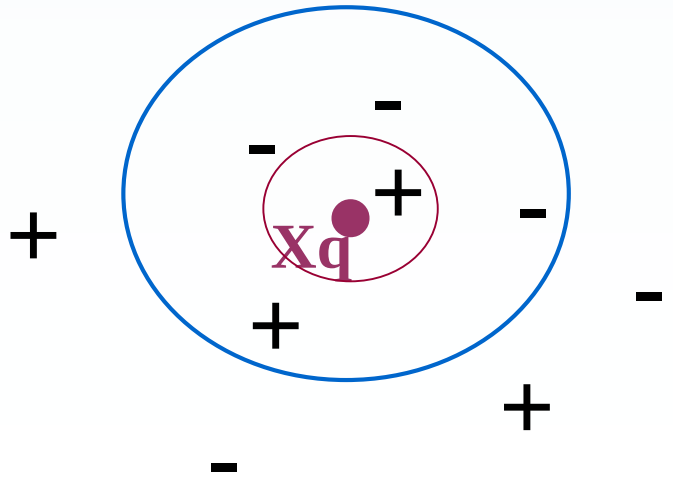
- Nearest Neighbors
- k-Nearest Neighbors
- Member of following families:
  - Instance-based Learning
  - Lazy learning (vs. eager)
  - Memory-based Learning
  - Exemplar methods
  - Non-parametric methods
- To predict y at a point x
  - Find the k nearest neighbors
  - $y^{est}(x)$ = the majority label or average of the y's of those points



Nearest Neighbor is an example of…. Instance-based learning

- Has been around since about 1910.
- To make a prediction, search database for similar datapoints, and fit with the local points.
- Assumption: Nearby points behavior similarly wrt y





Components of ML - K-NN

- Representation: nonparametric
  - ~~$\hat{y} = f(x; w) = w^T x$~~
- Loss function
  - $L(y, \hat{y}) = \left|\left|y - \hat{y}\right|\right|_2$
- Optimization method: not required
  - $argmin_w L(y, \hat{y}(w))$
  - ~~*gradient descent*~~

Instance/Memory-based Learning

Four things make a memory based learner:

- *A distance metric*
- *How many nearby neighbors to look at?*
- *A weighting function (optional)*
- *How to fit with the local points?*

1-Nearest Neighbor

Four things make a memory based learner:

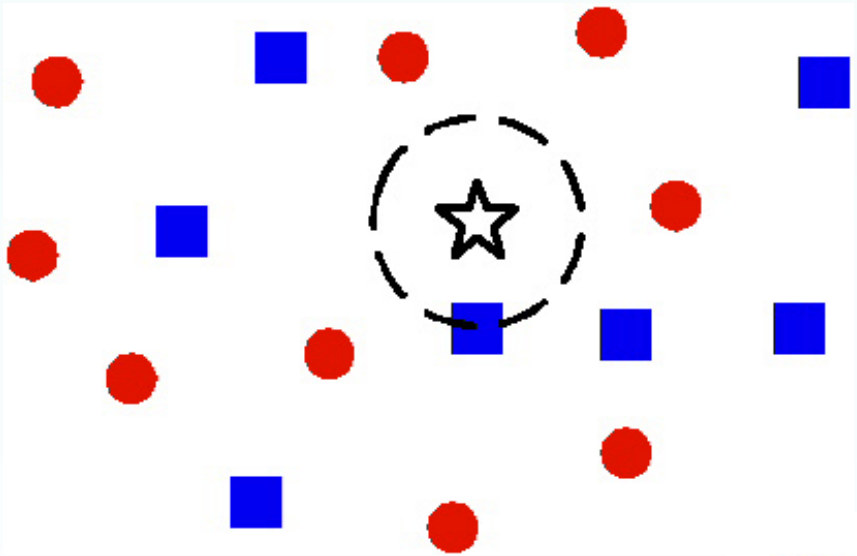
- *A distance metric*
  - **Euclidean (and others)**
  - $d(x, x') = \sqrt{\sum_i (x_i - x'_i)^2}$
- *How many nearby neighbors to look at?*
  - **1**
- *A weighting function (optional)*
  - **unused**
- *How to fit with the local points?*
  - **Just predict the same output as the nearest neighbour.**

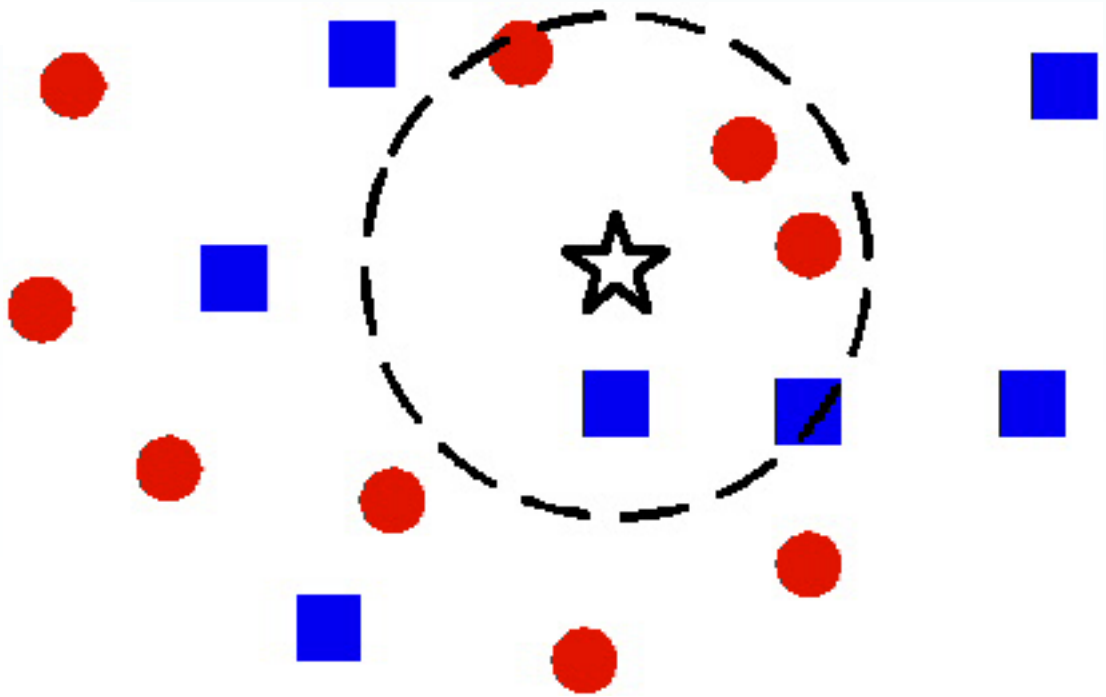
k-Nearest Neighbor

Four things make a memory based learner:

- *A distance metric*
  - **Euclidean (and others)**
- *How many nearby neighbors to look at?*
  - **k**
- *A weighting function (optional)*
  - **unused**
- *How to fit with the local points?*
  - **Just predict the average output among the nearest neighbours.**

1 vs k Nearest Neighbor





Norms

- L_p norm, Minkowski distance
  - $d(X,Y) = \left|\left|X,Y\right|\right|_p = (\sum_i |x_i - y_i|^p)^{1/p}$
- p=1
  - L_1 distance, L^1 distance or l_1 norm, Manhattan distance, city block distance
  - $d(X,Y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$
- p=2
  - L_2 distance, L^2 distance or l_2 norm, Euclidean distance
  - $d(X,Y) = \sqrt[2]{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$
- P=+∞
  - Chebyshev, $\lim_{n\to+\infty} (\sum_i |x_i - y_i|^p)^{1/p} = \max_i |x_i - y_i|$
- P=-∞
  - $\lim_{n\to-\infty} (\sum_i |x_i - y_i|^p)^{1/p} = \min_i |x_i - y_i|$

Handling Categorical Data: Similarity Measures

- Common

$$|\Gamma(X) \cap \Gamma(Y)|$$

- Jaccard

$$\frac{|\Gamma(X) \cap \Gamma(Y)|}{|\Gamma(X) \cup \Gamma(Y)|}$$

- Preferential attachment

$$|\Gamma(X)|.|\Gamma(Y)|$$

- Cosine

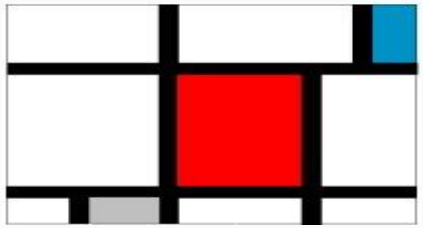
$$\frac{\Gamma(X).\Gamma(Y)}{\left|\left|\Gamma(X)\right|\right| \left|\left|\Gamma(X)\right|\right|}$$

Example

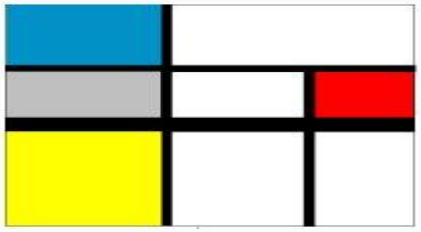
- Document classification
  - A **document** can be represented by thousands of attributes, each recording the *frequency* of a particular word (such as keywords) or phrase in the document.

Document	*team*	*coach*	*hockey*	*baseball*	*soccer*	*penalty*	*score*	*win*	*loss*	*season*
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

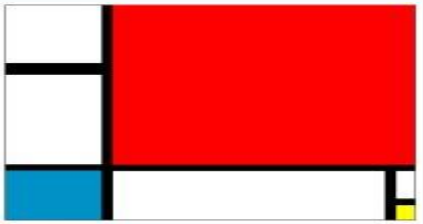
Example: Mondrian painting



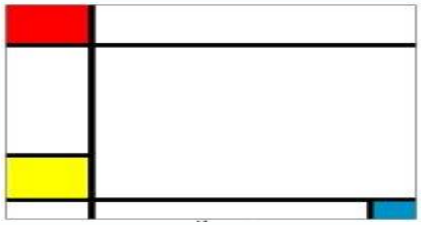
one



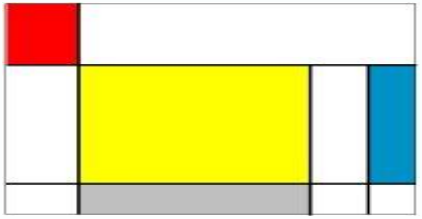
two



three

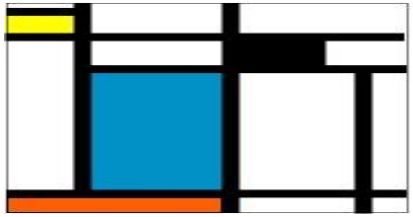


four

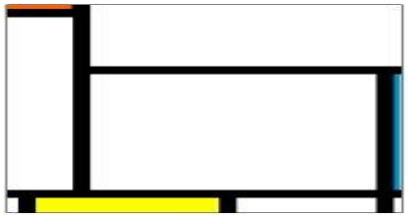


five

six



seven



Eight ?

Training data

Number	Lines	Line types	Rectangles	Colours	Mondrian?
1	6	1	10	4	*No*
2	4	2	8	5	*No*
3	5	2	7	4	Yes
4	5	1	8	4	Yes
5	5	1	10	5	*No*
6	6	1	8	6	Yes
7	7	1	14	5	*No*

Test instance

Number	Lines	Line types	Rectangles	Colours	Mondrian?
8	7	2	9	4	

Data Standardization

- Transform raw feature values into z-scores

$$z_{ij} = \frac{x_{ij} - m_j}{s_j}$$

  - x_{ij} is the value for the i^{th} sample and j^{th} feature
  - m_j is the average of all x_{ij} for feature j
  - s_j is the standard deviation of all x_{ij} over all input samples
- Range and scale of z-scores should be similar (providing distributions of raw feature values are alike)

Normalised training data

Number	Lines	Line types	Rectangles	Colours	Mondrian?
1	0.632	-0.632	0.327	-1.021	*No*
2	-1.581	1.581	-0.588	0.408	*No*
3	-0.474	1.581	-1.046	-1.021	Yes
4	-0.474	-0.632	-0.588	-1.021	Yes
5	-0.474	-0.632	0.327	0.408	*No*
6	0.632	-0.632	-0.588	1.837	Yes
7	1.739	-0.632	2.157	0.408	*No*

Test instance

Number	Lines	Line types	Rectangles	Colours	Mondrian?
8	1.739	1.581	-0.131	-1.021	

Distances of test instance from training data

Example	Distance of test from example	Mondrian?
1	2.517	*No*
2	3.644	*No*
3	2.395	Yes
4	3.164	Yes
5	3.472	*No*
6	3.808	Yes
7	3.490	*No*

Classification	
1-NN	Yes
3-NN	Yes
5-NN	*No*
7-NN	*No*

k-NN regression

- Simplest k-NN regression:
  - Let $x_1, x_2, \ldots, x_k$ be the k nearest neighbors of X and $y_1, y_2, \ldots y_k$ be their labels.

$$\hat{y} = \frac{1}{k} \sum_{j \in N_k(x)} y_j$$

  - Problems of kNN regression
    - discontinuities in the estimated function
    - 1NN: noise fitting problem
    - kNN (k>1): smoothes away noise

1-NN for Regression

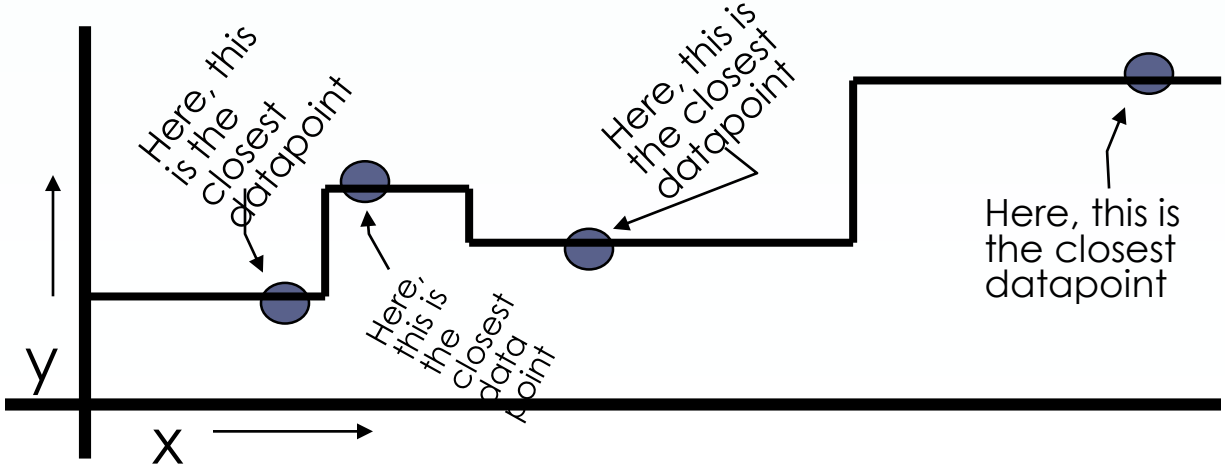
Given datapoints (x_1,y_1) $(x_2,y_2)..(x_N,y_N)$,where we assume $y_i=f(s_i)$ for some unknown function f.
Given query point x_q, your job is to predict
Nearest Neighbor:

1. Find the closest x_i in our set of datapoints

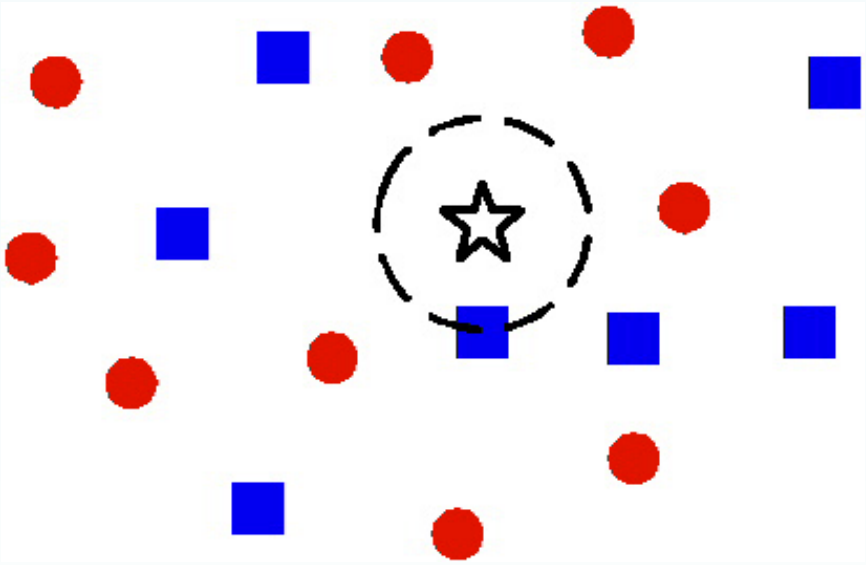
$$i(nn) = \underset{i}{\operatorname{argmin}} \left| x_i - x_q \right|$$

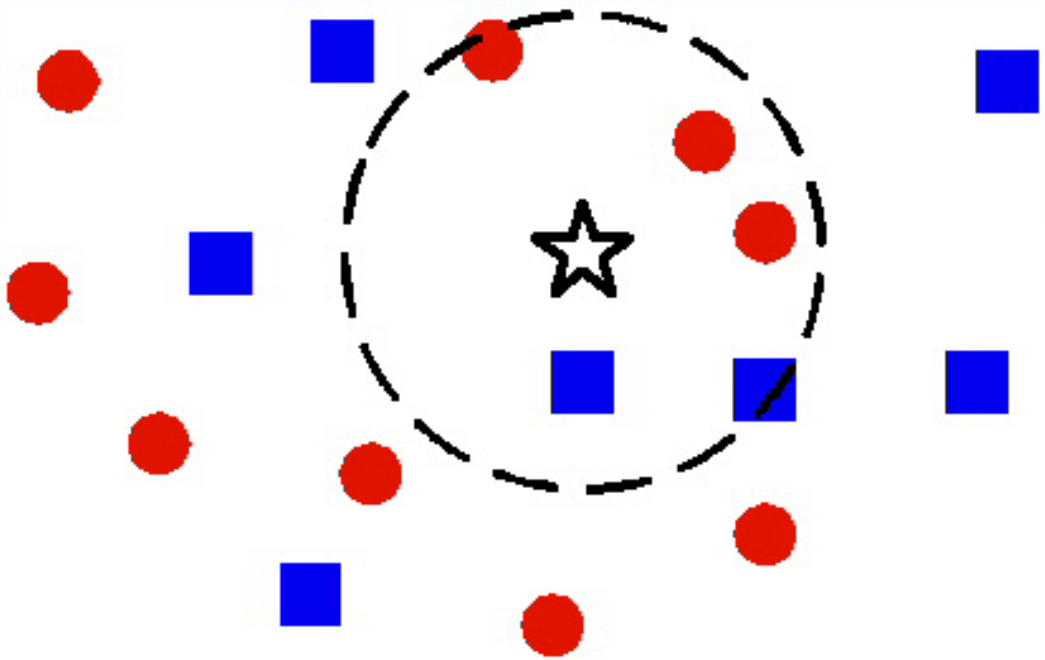
2. Predict

Here's a dataset with one input, one output and four datapoints.



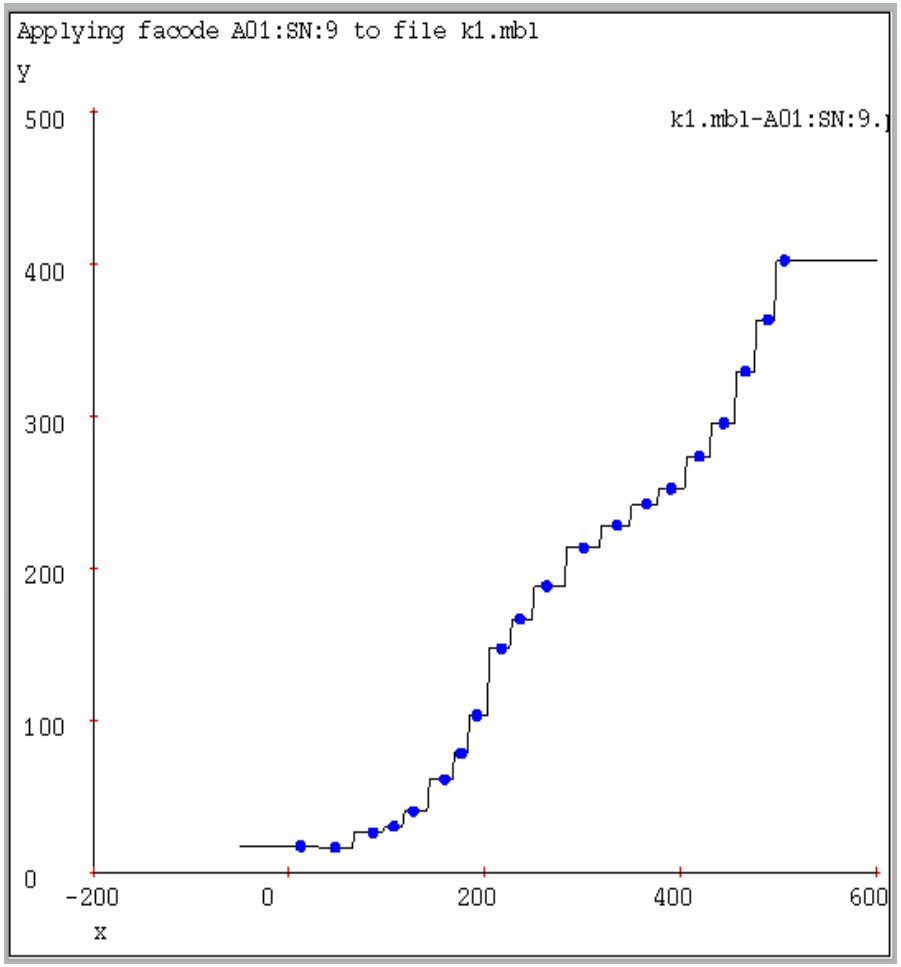
1 vs k Nearest Neighbor

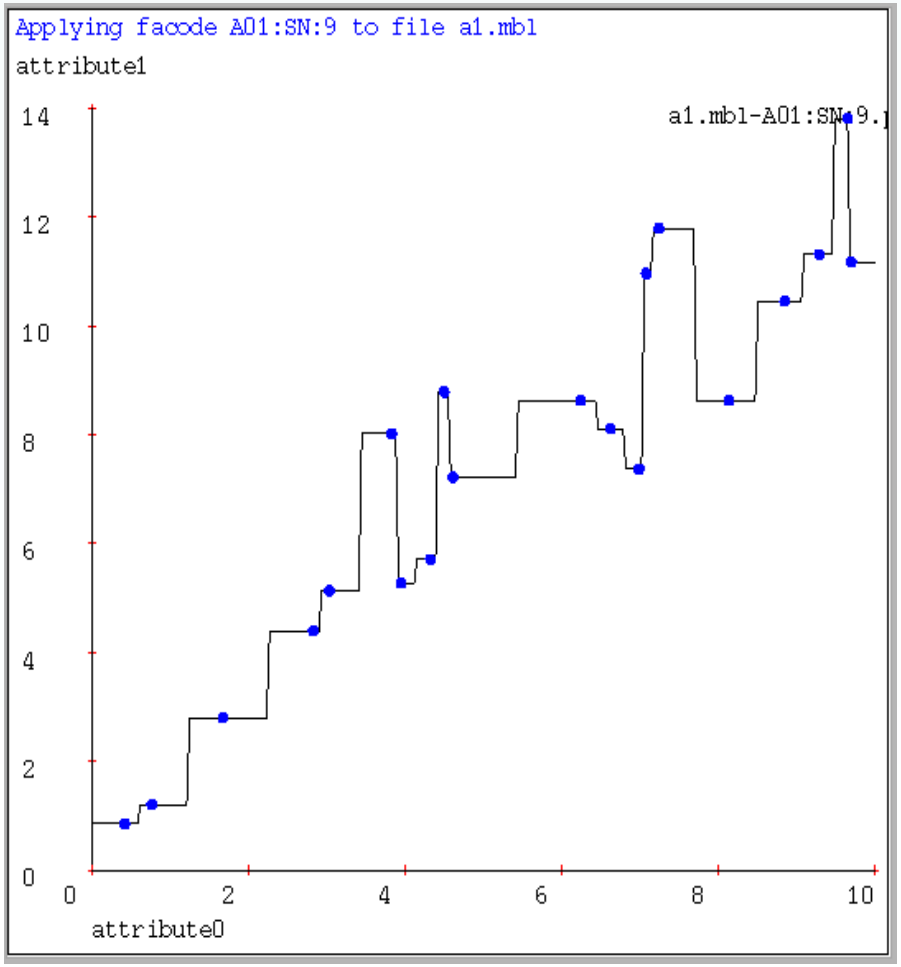




1-NN for Regression

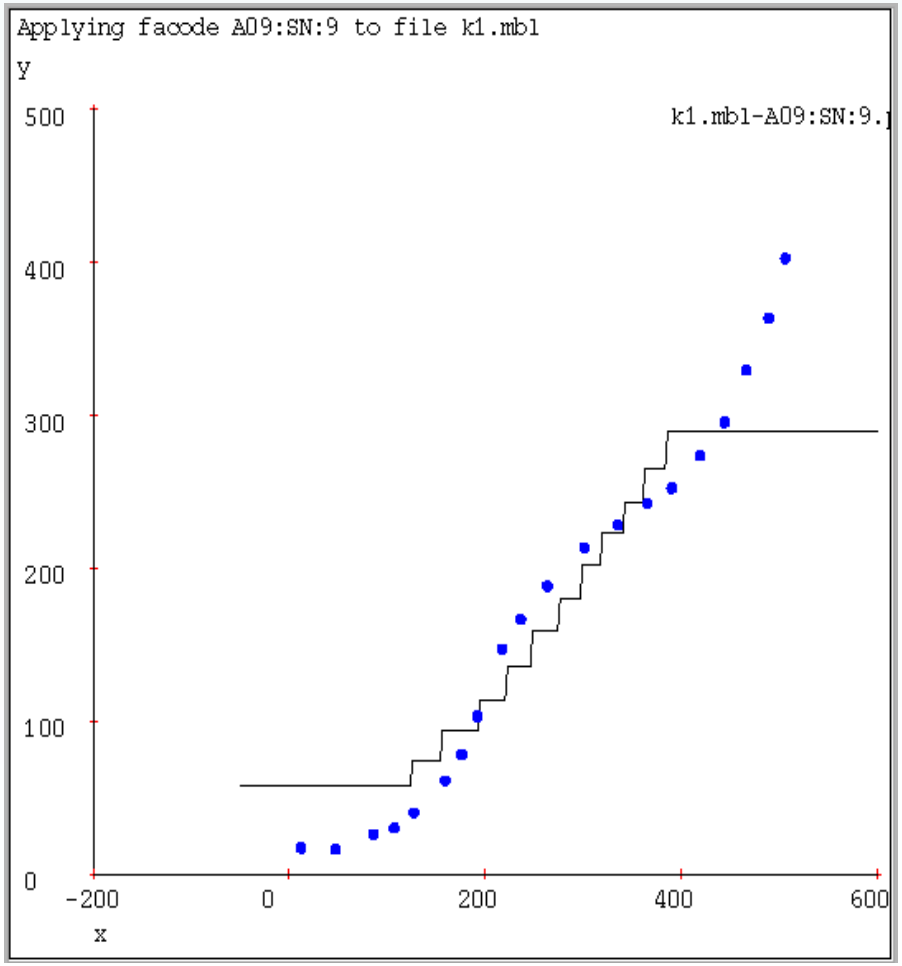
- Often bumpy (overfits)

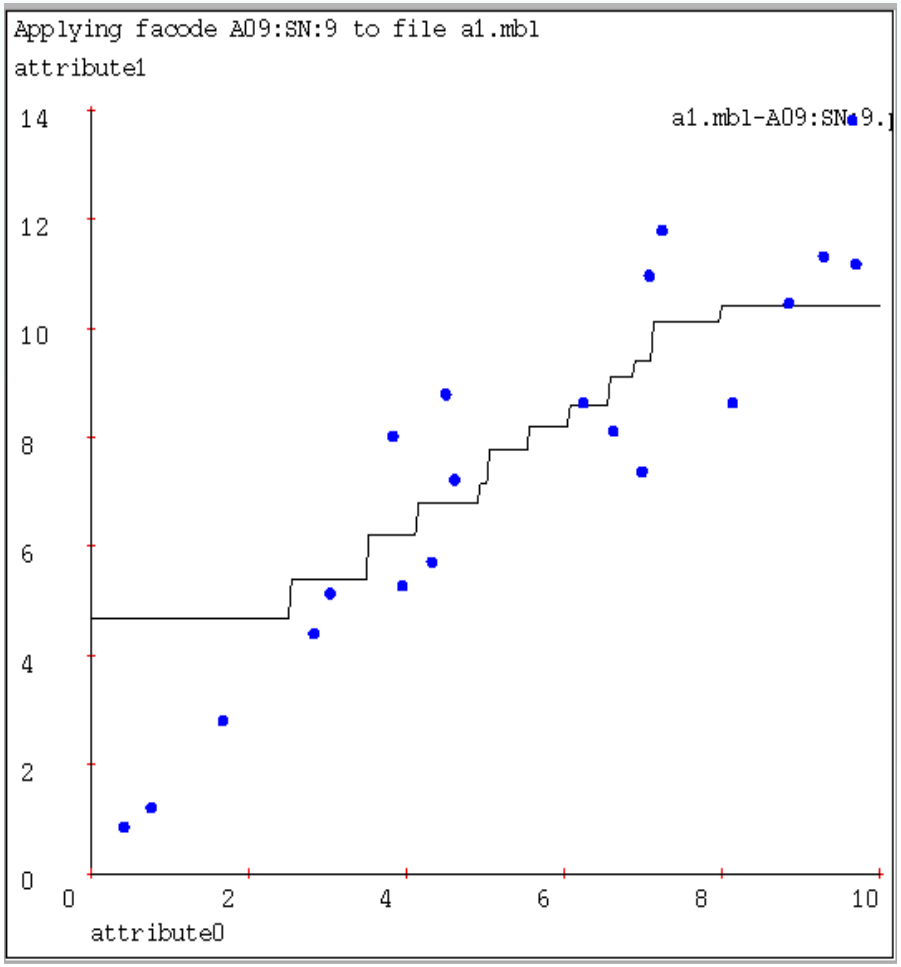




9-NN for Regression

- Often bumpy (overfits)





Pros and cons of KNN

- **Pros**
  - Easy to understand
  - No assumptions about data
  - Can be applied to both classification and regression
  - Works easily on multi-class problems
- **Cons**
  - Memory intensive / computationally expensive
  - Sensitive to scale of data
  - Not work well on rare event (skewed) target variable
  - Struggle when high number of independent variables
- A small value of *k* will lead to a large variance in predictions.
- Setting *k* to a large value may lead to a large model bias.

Curse of Dimensionality

- Prediction accuracy can quickly degrade when number of attributes grows.
  - Irrelevant attributes easily "swamp" information from relevant attributes
  - When many irrelevant attributes, similarity/distance measure becomes less reliable
- Remedy
  - Try to remove irrelevant attributes in pre-processing step
  - Weight attributes differently
  - Increase *k* (but not too much)

Lazy vs Eager Learning

- The lazy methods defer the decision of how to generalize beyond the training data until each new query instance is encountered.
- The eager method generalizes beyond the training data before observing the new query, committing at training time to the network structure and weights that define its approximation to the target function.

Parametric vs Non-Parametric Models

- Does the capacity (size of hypothesis class) grow with size of training data?
  - Yes = Non-Parametric Models
  - No = Parametric Models

- Example
  - http://www.theparticle.com/applets/ml/nearest_neighbor/

Reading

- C. M. Bishop, **Pattern recognition and machine learning**, Springer, 2006. (ch. 2)
- E. Alpaydin, **Introduction to Machine Learning**, 3rd ed., The MIT Press, 2014. (ch. 1)
- R. O. Duda, P. E. Hart. D. G. Stork, **Pattern classification**, 2nd ed., John Wiley & Sons, 2006. (ch. 4)