CS57300
PURDUE UNIVERSITY

SEPTEMBER 20, 2021

DATA MINING

PREDICTIVE MODELING

DATA MINING COMPONENTS

- Task specification: **Prediction**
- Knowledge representation
- Learning technique
- Prediction and/or interpretation

PREDICTIVE MODELING

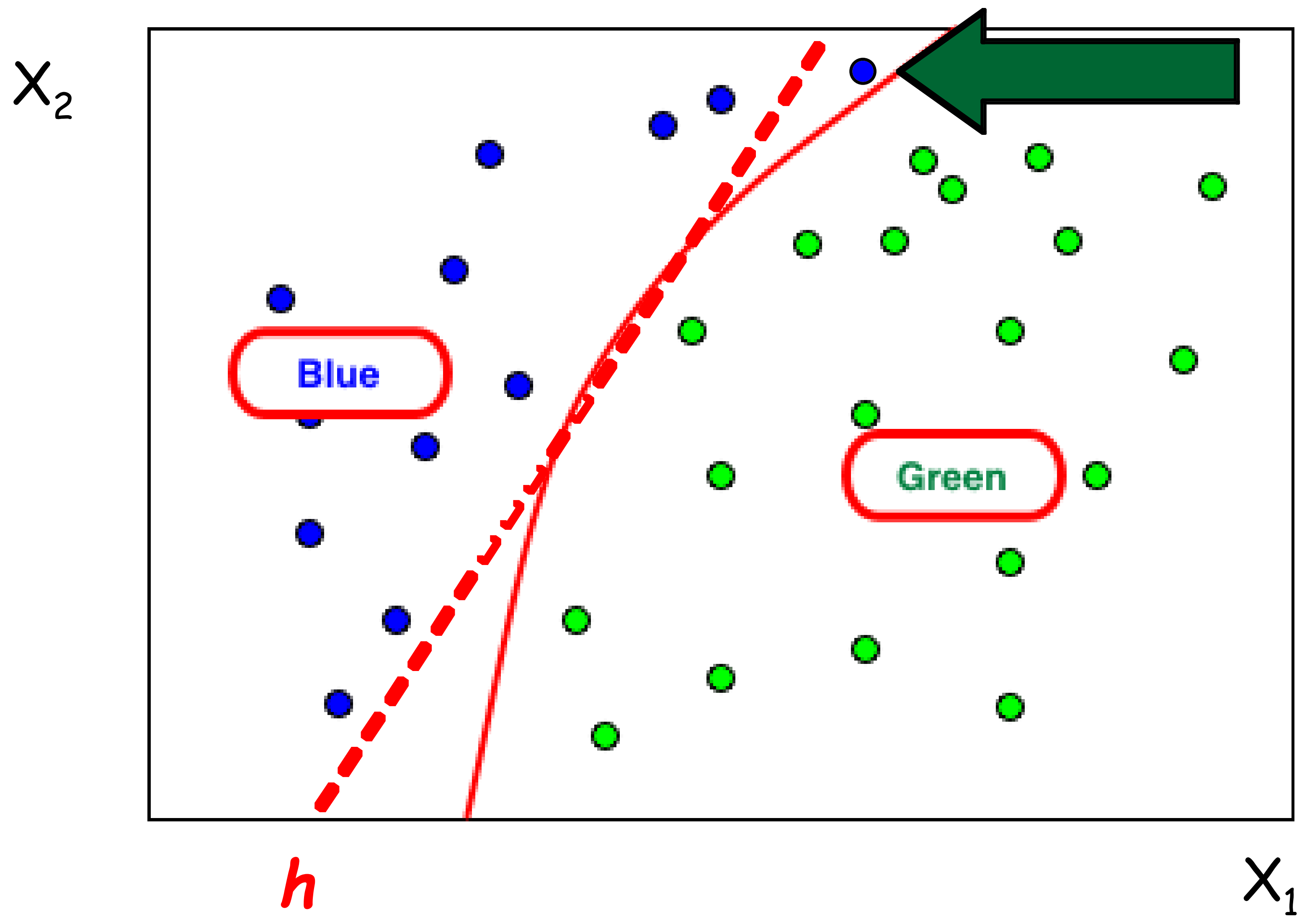
- Data representation:
  - Paired attribute vectors and class labels $<y(i), \boldsymbol{x}(i)>$ or $n \times p$ tabular data with class label (y) and p-1 attributes ($\boldsymbol{x}$)

- Task: Estimate a predictive function $f(\boldsymbol{x};\theta)=y$
  - Assume that there is a function $y=f(\boldsymbol{x})$ that **maps** data instances ($\boldsymbol{x}$) to class labels (y)
  - Construct a model that approximates the mapping
    - Classification: if y is categorical **Focus of this course**
    - Regression: if y is real-valued

CLASSIFICATION OUTPUT

- Different classification tasks can require different kinds of output
- **Class labels** – Each instance is assigned a single label
  - *Model only need to decide on crisp class boundaries*
- **Ranking** – Instances are ranked according to their likelihood of belonging to a particular class
  - *Model implicitly explores many potential class boundaries*
- **Probabilities** – Instances are assigned class probabilities $p(y|\mathbf{x})$
  - *Allows for more refined reasoning about sets of instances*
- Each requires progressively more accurate models (e.g., a poor probability estimator can still produce an accurate ranking)

DISCRIMINATIVE CLASSIFICATION



- Output: Class Labels
  - Direct mapping from inputs ***x*** to class label *y*
  - No attempt to model probability distributions
- Model the decision boundary directly
- May seek a discriminant function $f(\mathbf{x};\theta)$ that maximizes measure of separation between classes
- Examples:
  - Perceptrons, decision trees, nearest neighbor classifiers, support vector machines

PROBABILISTIC CLASSIFICATION

- Output: Probabilities
  - Maps from inputs $\mathbf{x}$ to class label y indirectly through posterior class distribution $p(y|\mathbf{x})$
- Model the underlying probability distributions
  - Posterior class probabilities: $p(y|\mathbf{x})$
  - Class-conditional and class prior: $p(\mathbf{x}|y)$ and $p(y)$
- Examples:
  - Naive Bayes classifier, logistic regression

KNOWLEDGE REPRESENTATION

KNOWLEDGE REPRESENTATION

- Underlying structure of the model or patterns that we seek from the data

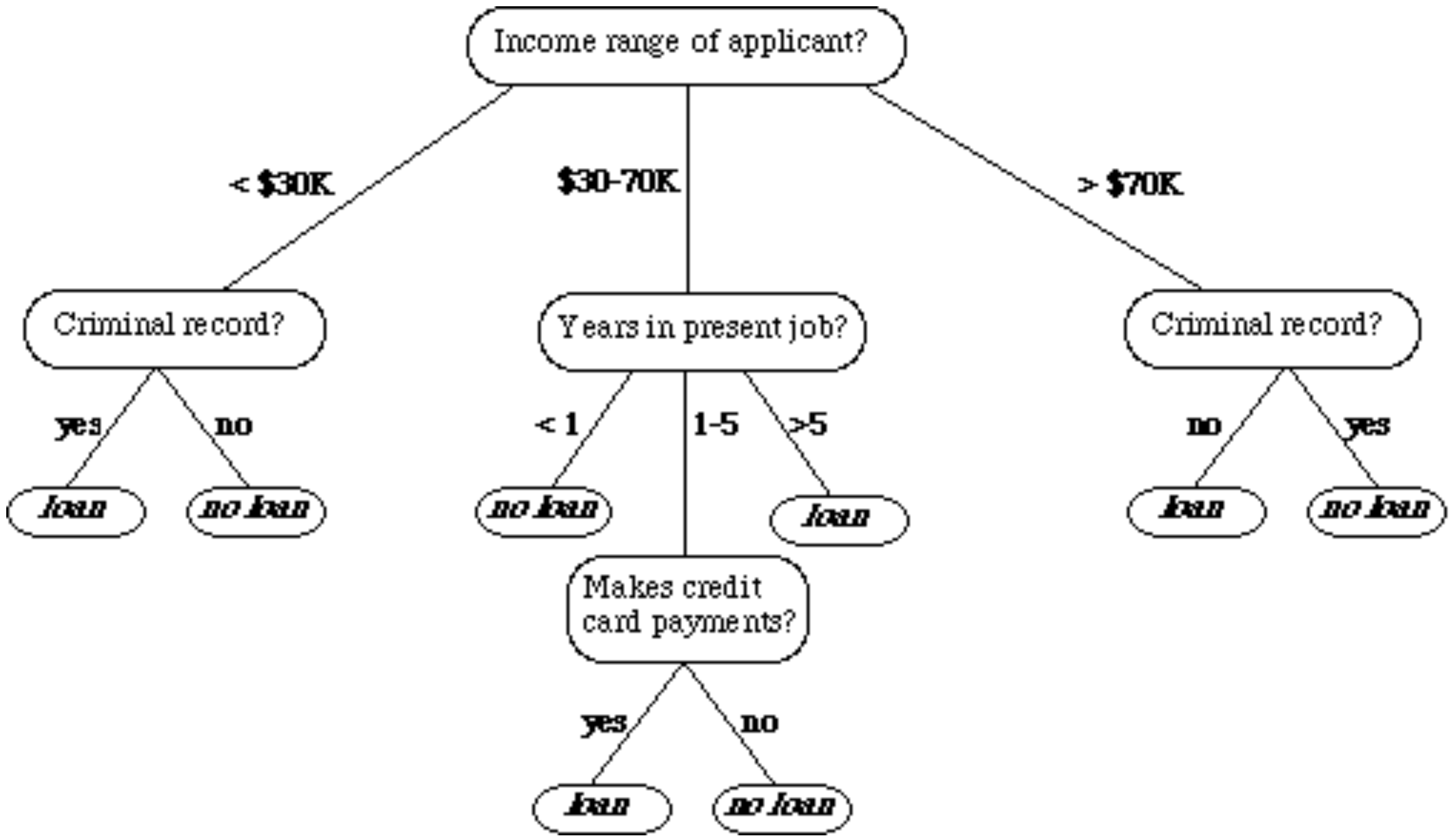
- Model: high-level global description of dataset
  - Choice of model family determines **space** of parameters and structure
  - Estimate model parameters and possibly model structure from data

PERCEPTRON

$$f(x) = \begin{cases} 1 & \sum w_j x_j > 0 \\ 0 & \sum w_j x_j \leq 0 \end{cases}$$

Model space:
weights *w*, for each of *j* attributes

DECISION TREE



Model space:

all possible decision trees

MODEL SPACE

- How large is the space?
- Simplifying assumptions
  - Binary tree
  - Fixed depth
  - 10 binary attributes
- Can we search exhaustively?

Tree depth	Number of trees
1	10
2	8×10^{2}
3	3×10^{6}
4	2×10^{13}
5	5×10^{25}

NEAREST NEIGHBOR

Rule: find *k* closest (training) points to the test instance and assign the most frequently occurring class

Model space:
Choice of *k*, definition of distance, etc.

NAIVE BAYES CLASSIFIER

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

$$= \frac{\prod_i p(x_i|y)\, p(y)}{\sum_j p(\mathbf{x}|y_j)p(y_j)}$$

Model space:

parameters in conditional distributions $p(x_i|y)$

parameters in prior distribution $p(y)$

PREDICTIVE MODELING: LEARNING

LEARNING PREDICTIVE MODELS

- Select a **knowledge representation** (a "model")
  - Defines a **space** of possible models $M=\{M_1, M_2, \ldots, M_k\}$
- Define **scoring functions** to "score" different models
- Use **search** to identify "best" model(s)
  - Search the space of models (i.e., with alternative structures and/or parameters)
  - Evaluate possible models with **scoring function** to determine the model which best fits the data
  - Score function can be used to search over ***parameters*** and/or ***model structure***

PREDICTIVE SCORING FUNCTIONS

- Assess the quality of predictions for a set of instances
  - Measures **difference** between the prediction *M* makes for an instance *i* and the true class label value of *i*

$$S(M) = \underbrace{\sum_{i=1}^{N}}_{\text{Sum over examples}} \underbrace{d}_{\text{Distance between predicted and true}}\left[\underbrace{f(x(i); M)}_{\text{Predicted class label for item } i}, \underbrace{y(i)}_{\text{True class label for item } i}\right]$$

PREDICTIVE SCORING FUNCTIONS

- Common score functions:
  - Zero-one loss

$$S_{0/1}(M) = \frac{1}{N} \sum_{i=1}^{N} I\big[f(x(i); M), y(i)\big]$$

$$\text{where } I(a,b) = \begin{cases} 1 & a \neq b \\ 0 & \text{otherwise} \end{cases}$$

  - Squared loss

$$S_{sq}(M) = \frac{1}{N} \sum_{i=1}^{N} \big[f(x(i); M) - y(i)\big]^2$$

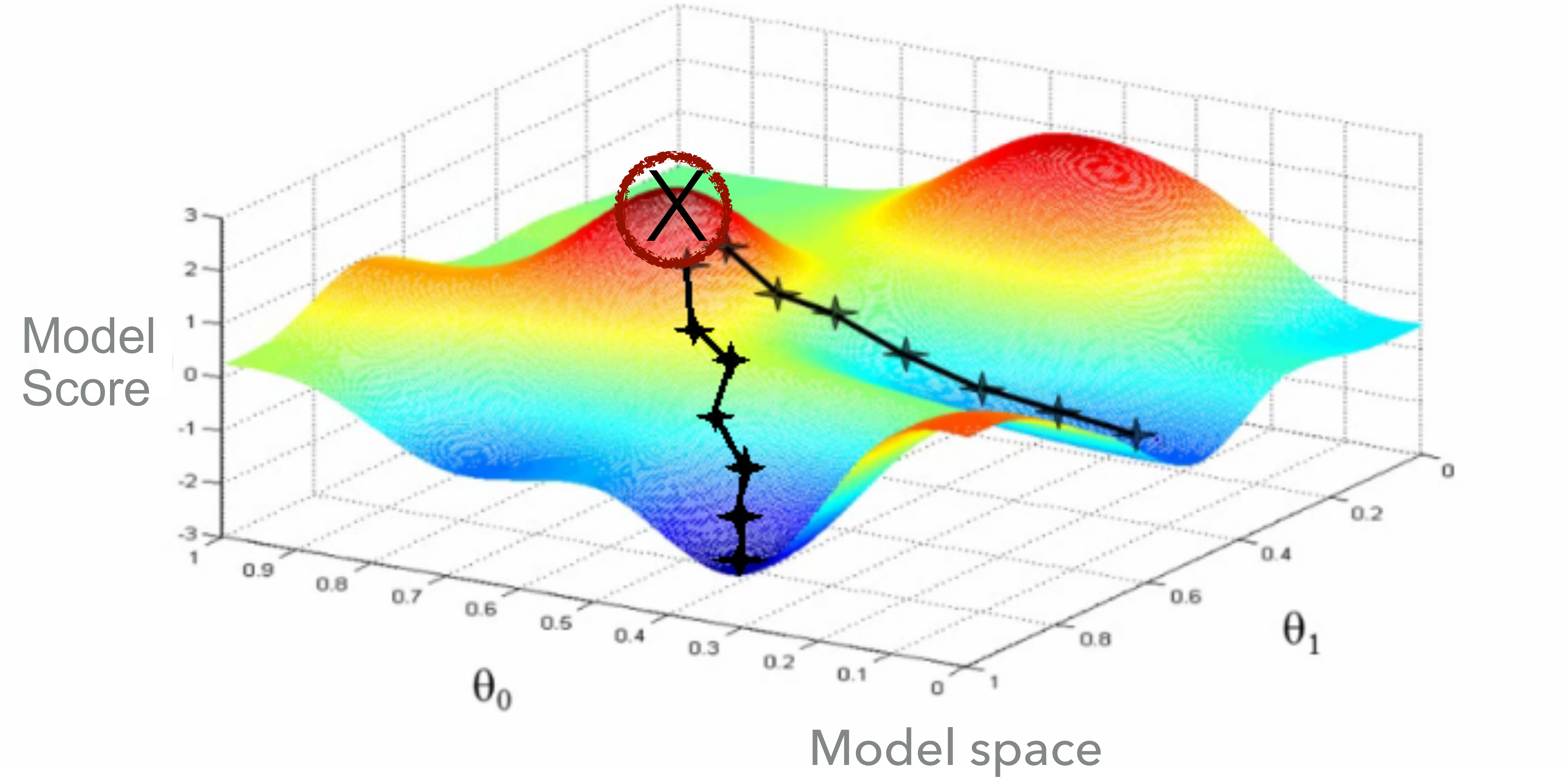
- Do we minimize or maximize these functions?

SEARCHING OVER MODELS

- Consider a **space** of possible models $M=\{M_1, M_2, \ldots, M_k\}$ with parameters θ
- Search could be over model structures or parameters, e.g.:
  - **Parameters**: In a perceptron, find the weights (**w**) that minimize 0-1 loss
  - **Model structure**: In decision trees, find the tree structure that maximizes accuracy on the training data

WHAT SPACE ARE WE SEARCHING?

$$\text{Learned model} \approx (\theta_0 = 0.8, \theta_1 = 0.4)$$



Alex Holehuse, Notes from Andrew Ng's Machine Learning Class, http://www.holehouse.org/mlclass/01_02_Introduction_regression_analysis_and_gr.html

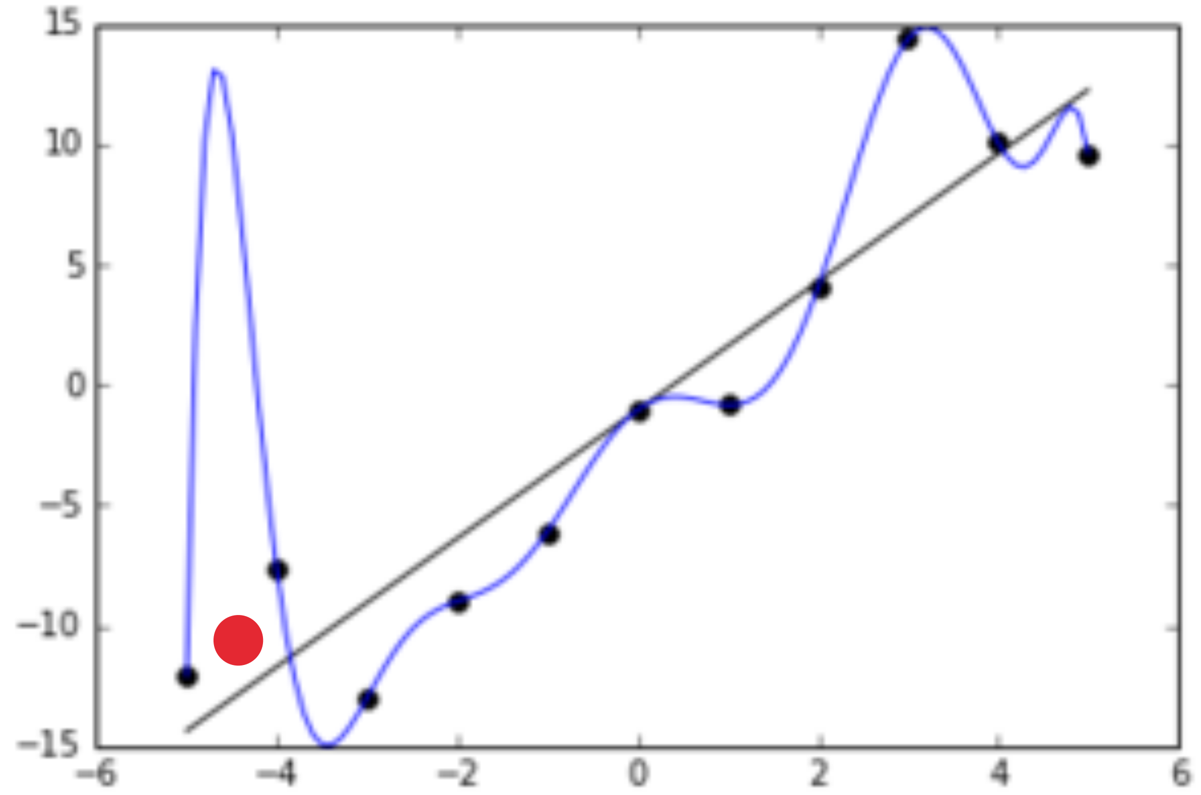
OPTIMIZATION OVER SCORE FUNCTIONS

- **Smooth** functions:
  - If a function is *smooth*, it is differentiable and the derivatives are continuous, then we can use gradient-based optimization
    - If function is *convex*, we can solve the minimization problem in closed form: $\nabla S(\theta)$ using **convex optimization**
    - If function is smooth but not convex/concave, we can use iterative search over the surface of S to find a local minimum (e.g., hill-climbing)
- **Non-smooth** functions:
  - If the function is *discrete*, then traditional optimization methods that rely on smoothness are not applicable. Instead we need to use **combinatorial optimization**

PREDICTION AND EVALUATION

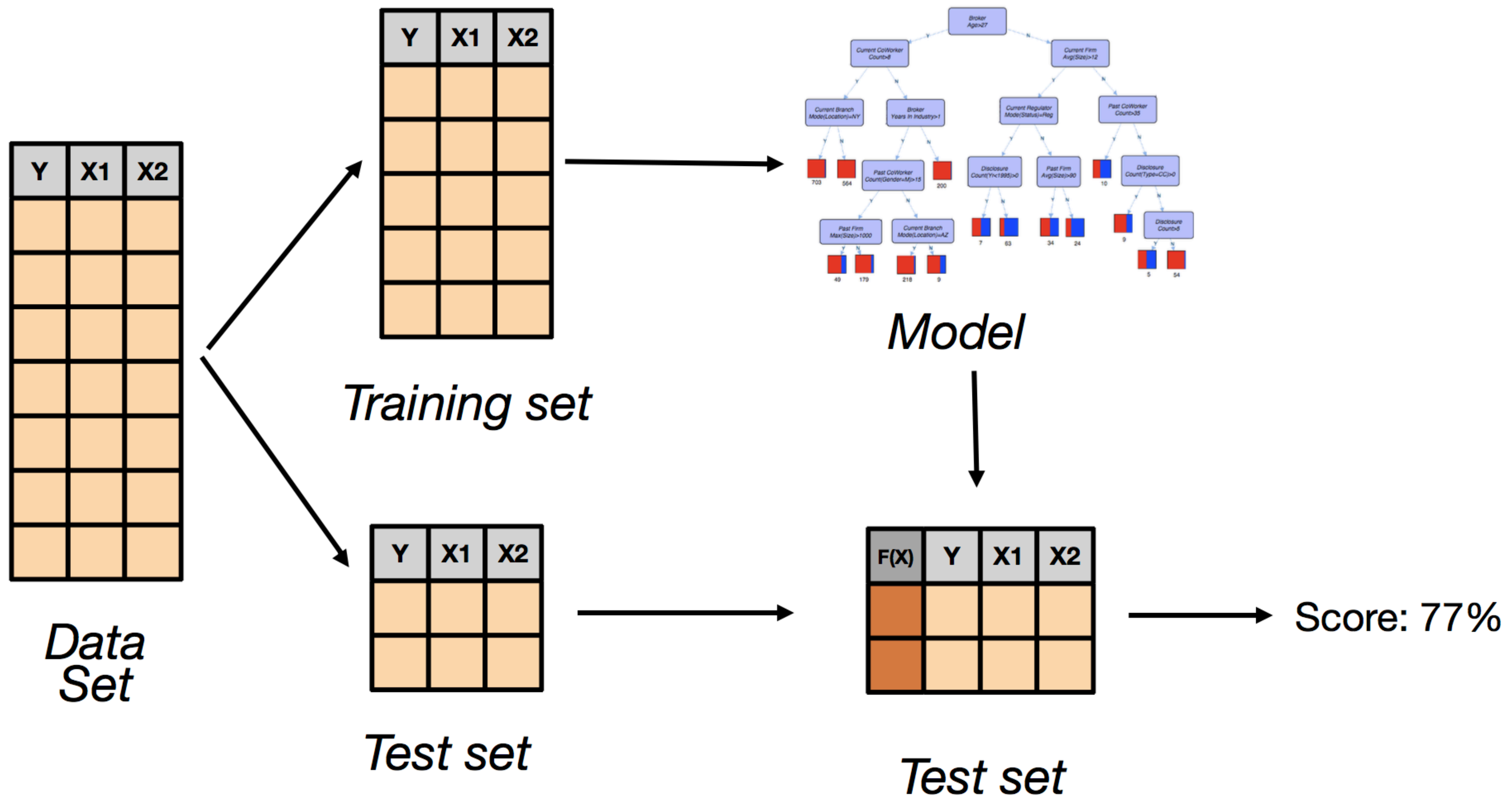
PREDICTING UNSEEN DATA

- Use the learned model to predict class label *y* for unseen data ***x***
- Evaluating the performance of the learned model (i.e., its capability to generalize) by computing the scoring functions on the unseen data
- How to evaluate the performance of a model given a limited amount of data?
  - Can we train the model on the entire dataset and use the scoring function value on it as an estimate for the model's performance on the unseen data?



SPLIT INTO TRAINING DATA AND TESTING DATA

- Split the dataset into disjoint two sets: training set and testing set
- Learn the model using the training set and evaluate the model's performance on testing set



OVERFITTING

- When the performance of your model on the training data is much better than its performance on the testing data, you are likely overfitting…
- Consider a distribution D of data representing a population and a sample D_S drawn from D, which is used as training data
- Given a model space M, a score function S, and a learning algorithm that returns a model $m \in M$, the algorithm **overfits** the training data D_S if:
  $\exists m' \in M$ such that $S(m, D_S) > S(m', D_S)$ but $S(m, D) < S(m', D)$
  - In other words, there is another model (m') that is better on the entire distribution and if we had learned from the full data we would have selected it instead

EXAMPLE LEARNING PROBLEM

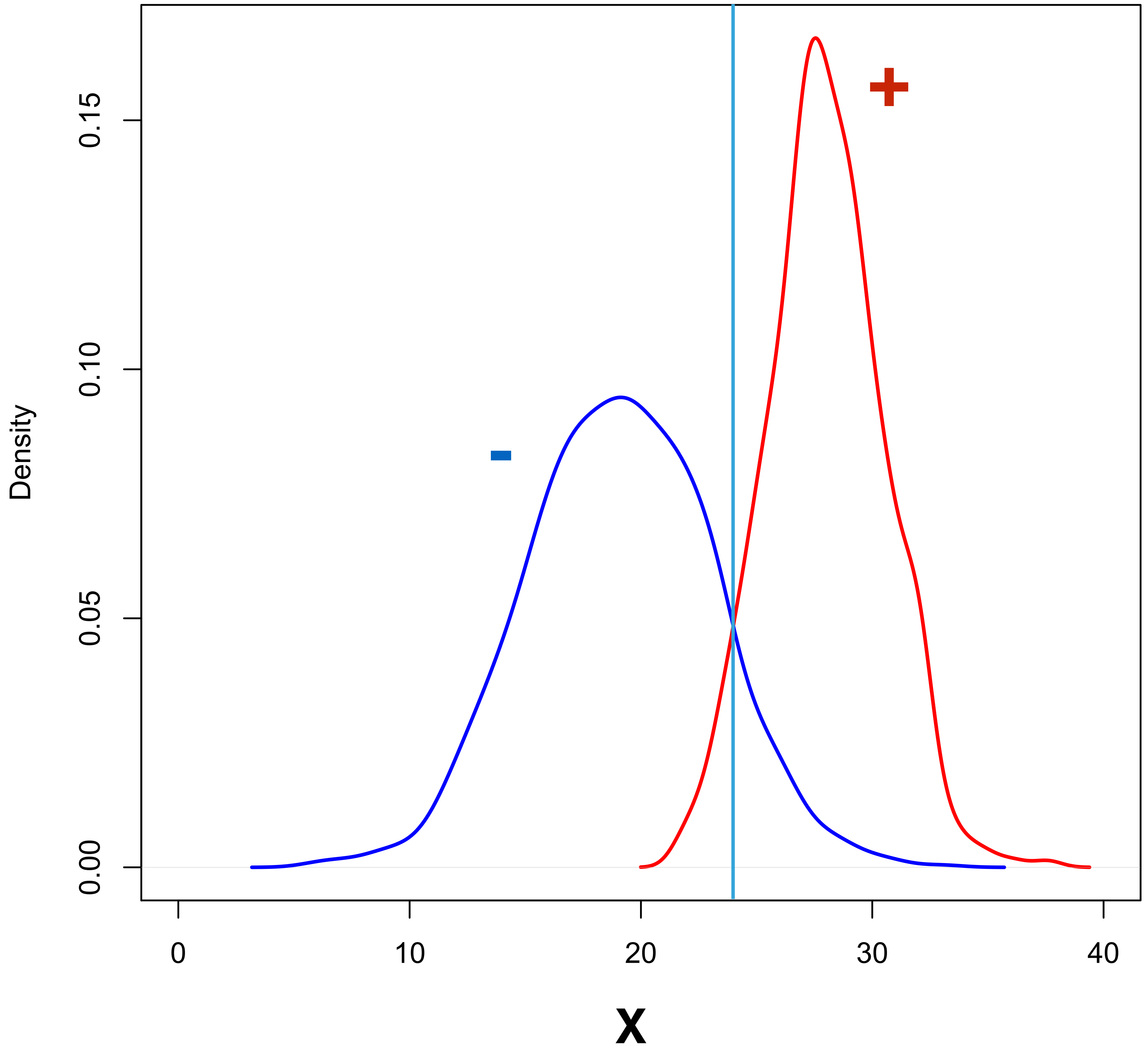
Task: Devise a rule to classify items based on the attribute ***X***

Knowledge representation:
If-then rules

Example rule:
If x > 24 then +
Else -

What is the model space?

All possible thresholds



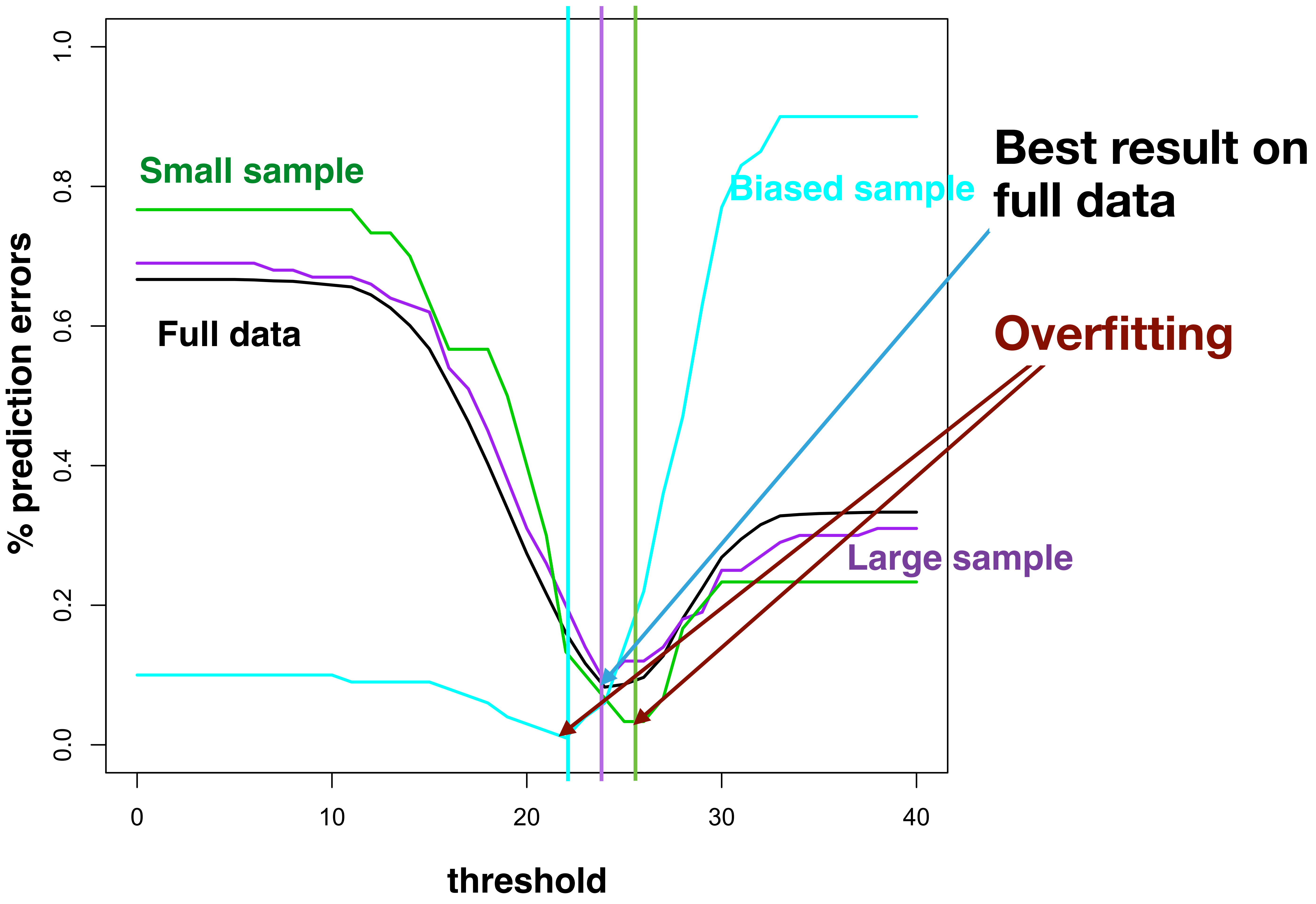
What score function?

Prediction error rate

SCORE FUNCTION OVER MODEL SPACE

Search procedure?

Try all thresholds, select one with lowest score



DEAL WITH OVERFITTING

- It's easier for more complex models to overfit
- Approaches to avoid overfitting
  - Regularization
  - Penalty term in scoring function
  - Model selection through cross-validation

NAIVE BAYES CLASSIFIERS

CLASSIFICATION AS PROBABILITY ESTIMATION

- Instead of learning a function f that assigns labels
- Learn a conditional probability distribution over the output of function f
- $P(f(x) \mid x) = P(f(x) = y \mid x_1, x_2, \ldots, x_p)$
- Can use probabilities for the other two tasks
  - Classification
  - Ranking

KNOWLEDGE REPRESENTATION AND MODEL SPACE

BAYES RULE FOR PROBABILISTIC CLASSIFIER

$$P(C|\mathbf{X}) = \frac{P(\mathbf{X}|C)P(C)}{P(\mathbf{X})}$$

BAYES RULE

$$= \frac{P(\mathbf{X}|C)P(C)}{[P(\mathbf{X}|C=+)P(C=+)] + [P(\mathbf{X}|C=-)P(C=-)]}$$

$$\propto P(\mathbf{X}|C)P(C)$$

DENOMINATOR: NORMALIZING FACTOR TO MAKE PROBABILITIES SUM TO 1 (CAN BE COMPUTED FROM NUMERATORS)

NAIVE BAYES CLASSIFIER

$$P(C|\mathbf{X}) \propto P(\mathbf{X}|C)P(C)$$

BAYES RULE

$$\propto \prod_{i=1}^{m} P(X_i|C)P(C)$$

NAIVE ASSUMPTION

Assumption: Attributes are *conditionally independent* given the class

NBC LEARNING

$$P(BC|A,I,S,CR) = \frac{P(A,I,S,CR|BC)P(BC)}{P(A,I,S,CR)}$$
$$= \frac{P(A|BC)P(I|BC)P(S|BC)P(CR|BC)P(BC)}{P(A,I,S,CR)}$$
$$\propto P(A|BC)P(I|BC)P(S|BC)P(CR|BC)P(BC)$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

NBC parameters = CPDs+prior

```
CPDs:  P(A|BC)
       P(I|BC)
       P(S|BC)
       P(CR|BC)
Prior:P(BC)
```

SCORING FUNCTION

LIKELIHOOD

- Let $D = \{x(1), ..., x(n)\}$, where $x(i)$ = <$\boldsymbol{x}_i$, c_i>
- Assume the data D are independently sampled from the same distribution:

$$p(X|\theta)$$

- The likelihood function represents the probability of the data as a function of the model parameters:

$$\begin{aligned} L(\theta|D) &= L(\theta|x(1), ..., x(n)) \\ &= p(x(1), ..., x(n)|\theta) \\ &= \prod_{i=1}^{n} p(x(i)|\theta) \end{aligned}$$

If instances are independent, likelihood is product of probs

LIKELIHOOD (CONT')

- Likelihood is not a probability distribution
    - Gives relative probability of data given a parameter
    - Numerical value of *L* is not relevant, only the ratio of two scores is relevant, e.g.,:

$$\frac{L(\theta_1|D)}{L(\theta_2|D)}$$

- **Likelihood function**: allows us to determine unknown parameters based on known outcomes
- **Probability distribution**: allows us to predict unknown outcomes based on known parameters

NBC: LIKELIHOOD

- NBC likelihood uses the NBC probabilities for each data instance (i.e., probability of the class given the attributes)

$$L(\theta|D) = \prod_{i=1}^{n} p(x(i)\,|\,\theta)$$

General likelihood

$$= \prod_{i=1}^{n} p(\mathbf{x}_i|c_i,\theta)P(c_i|\theta)$$

Product rule

$$= \prod_{i=1}^{n}\prod_{j=1}^{m} p(x_{ij}|c_i,\theta)P(c_i|\theta)$$

Naive assumption

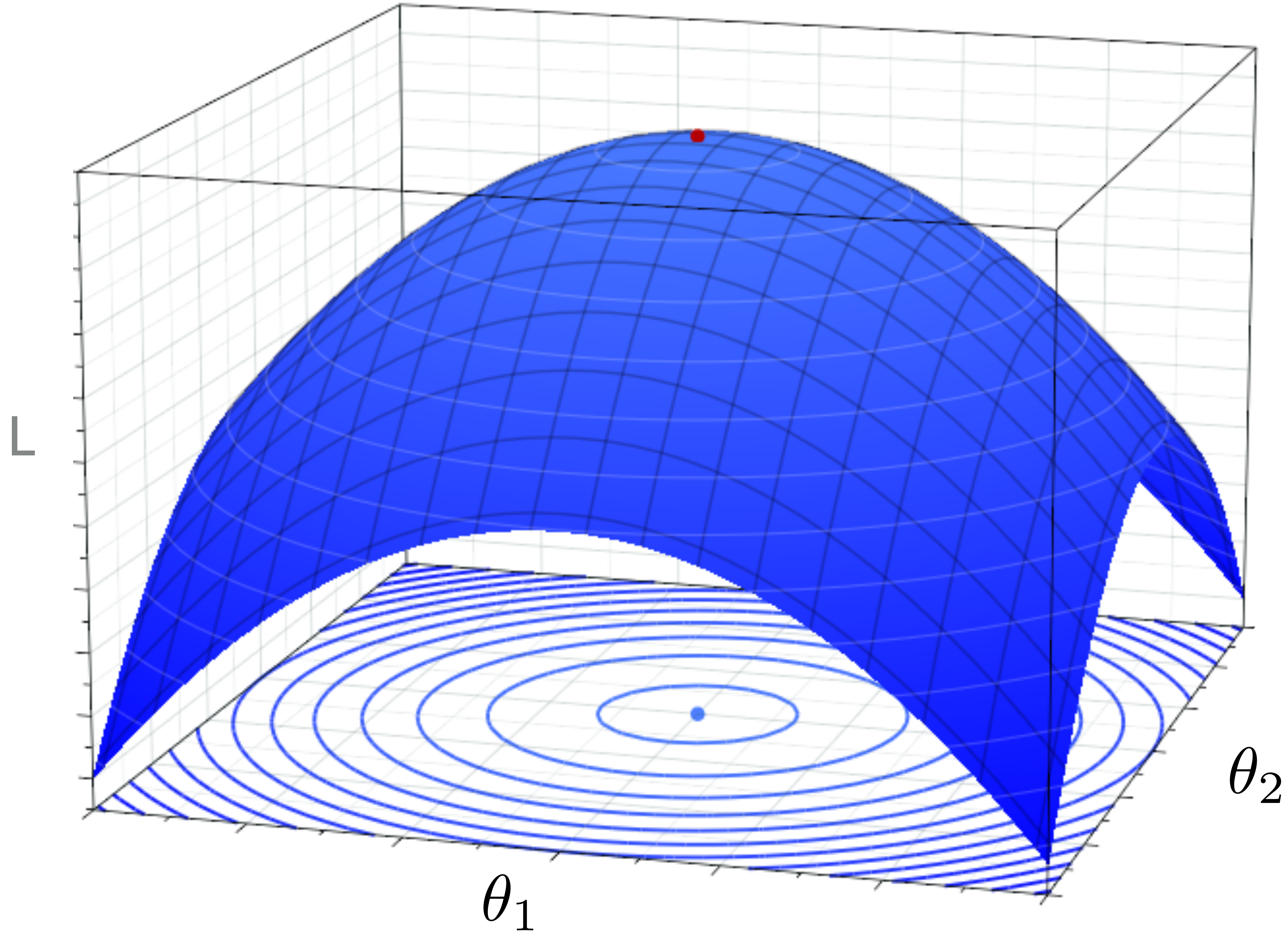
SEARCH

MAXIMUM LIKELIHOOD ESTIMATION

- "Learn" the best parameters by finding the values of θ that maximizes likelihood: $$\hat{\theta}_{MLE} = \arg\max_{\theta} L(\theta)$$
- Often easier to work with loglikelihood:

$$\begin{aligned} l(\theta|D) &= log\ L(\theta|D) \\ &= log \prod_{i=1}^{n} p(x(i)|\theta) \\ &= \sum_{i=1}^{n} logp(x(i)|\theta) \end{aligned}$$

LIKELIHOOD SURFACE



If the likelihood surface is convex we can often determine the parameters that maximize the function analytically