CS 484
Data Mining

Classification 7

Some slides are from Professor Padhraic Smyth at UC Irvine

Bayesian Belief networks

- Conditional independence assumption of Naïve Bayes classifier is too strong.
- Allows to specify which pairs of attributes are conditionally independent.
- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions
- Syntax:
  - a set of nodes, one per variable
  - a directed, acyclic graph (link ≈ "directly influences")
  - a conditional distribution for each node given its parents:

$$\mathbf{P}\,(X_i \mid \text{Parents}\,(X_i))$$

- In the simplest case, conditional distribution represented as a conditional probability table (CPT) giving the distribution over X_i for each combination of parent values

Background: Law of Total Probability

- Law of Total Probability (aka "summing out" or marginalization)

$$P(A) = \Sigma_i \; P(A, B_i)$$
$$= \Sigma_i \; P(A \mid B_i) \, P(B_i)$$

- Why is this useful?

  Given a joint distribution (e.g., P(A,B,C,D)) we can obtain any "marginal" probability (e.g., P(B)) by summing out the other variables, e.g.,

$$P(B) = \Sigma_i \, \Sigma_j \, \Sigma_k \, P(A_i, B, C_j, D_k)$$

- Less obvious: we can also compute <u>any conditional probability of interest</u> given a joint distribution, e.g.,

$$P(C \mid B) = \Sigma_i \, \Sigma_j \, P(A_i, C, D_j \mid B)$$
$$= 1 / P(B) \;\; \Sigma_i \, \Sigma_j \, P(A_i, C, D_j, B)$$

  where 1 / P(B) is just a normalization constant

- Thus, the joint distribution contains the information we need to compute any probability of interest.

Background: The Chain Rule or Factoring

- We can always write

  $P(A, B, C, \ldots Z) = P(A \mid B, C, \ldots. Z)\, P(B, C, \ldots Z)$

  (by definition of joint probability)

- Repeatedly applying this idea, we can write

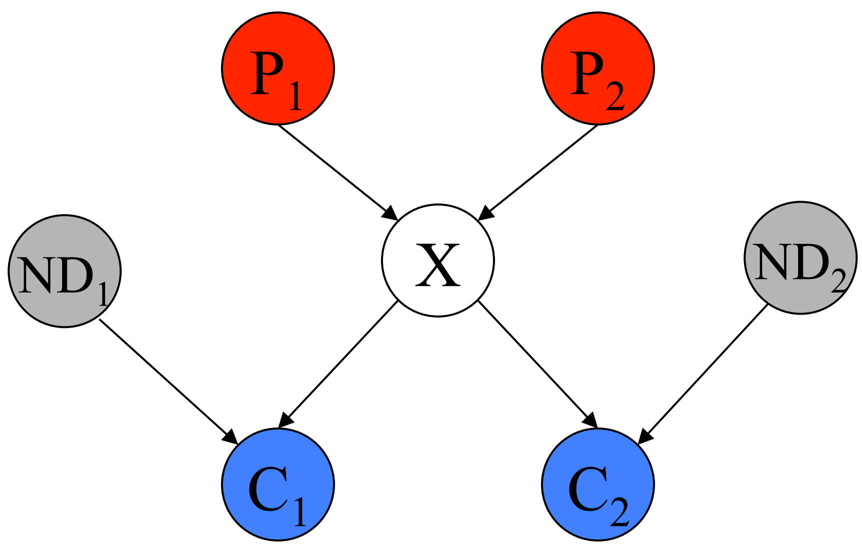
  $P(A, B, C, \ldots Z) = P(A \mid B, C, \ldots. Z)\, P(B \mid C,.. Z)\, P(C \mid .. Z)..P(Z)$

- This factorization holds for any ordering of the variables

- This is the chain rule for probabilities

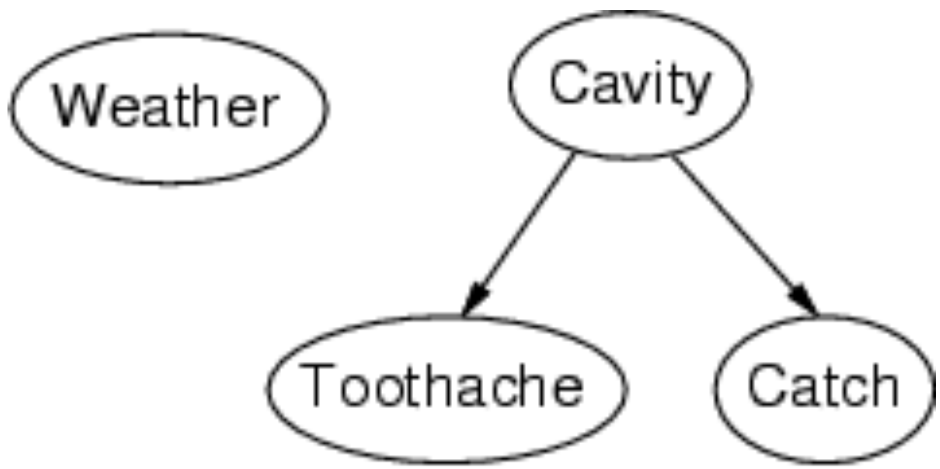
Conditional Independence

The Markov condition: given its parents (P_1, P_2), a node (X) is conditionally independent of its non-descendants (ND_1, ND_2)



Example

- Topology of network encodes conditional independence assertions:



- *Weather* is independent of the other variables
- *Toothache* and *Catch* are conditionally independent given *Cavity*

Conditional Independence

- 2 random variables A and B are conditionally independent given C iff

  $$P(A, B \mid C) = P(A \mid C)\, P(B \mid C)$$

- More intuitive (equivalent) conditional formulation
  - A and B are conditionally independent given C iff

    $$P(A \mid B, C) = P(A \mid C)$$

  - Intuitive interpretation:

    $P(A \mid B, C) = P(A \mid C)$ tells us that learning about B, given that we already know C, provides no change in our probability for A, i.e., B contains no information about a beyond what C provides
- Can generalize to more than 2 random variables
  - E.g., K different symptom variables $X_1, X_2, \ldots X_K$, and C = disease
  - $P(X_1, X_2, \ldots. X_K \mid C) = \Pi\ P(Xi \mid C)$
  - Also known as the naïve Bayes assumption

Bayesian Networks

- A Bayesian network specifies a joint distribution in a structured form
- Represent dependence/independence via a directed graph
  - Nodes = random variables
  - Edges = direct dependence
- Structure of the graph ⇔ Conditional independence relations

$$p(X_1, X_2, \ldots X_N) = \Pi\, p(X_i \mid parents(X_i))$$

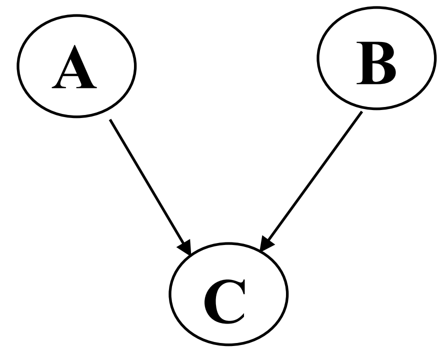
The full joint distribution

The graph-structured approximation

- Requires that graph is acyclic (no directed cycles)
- 2 components to a Bayesian network
  - The graph structure (conditional independence assumptions)
  - The numerical probabilities (for each variable given its parents)

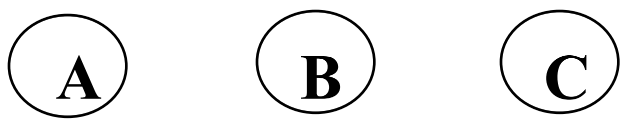
Example of a simple Bayesian network

$$P(A,B,C) = P(C|A,B)P(A)P(B) \longleftrightarrow$$



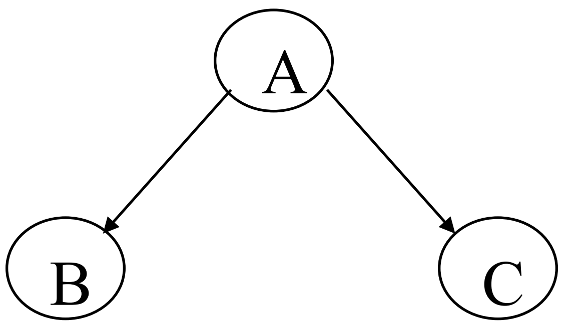
- Probability model has simple factored form
- Directed edges => direct dependence
- Absence of an edge => conditional independence
- Also known as belief networks, graphical models, causal networks
- Other formulations, e.g., undirected graphical models

Examples of 3-way Bayesian Networks



Marginal Independence:
P(A,B,C) = P(A) P(B) P(C)

Examples of 3-way Bayesian Networks

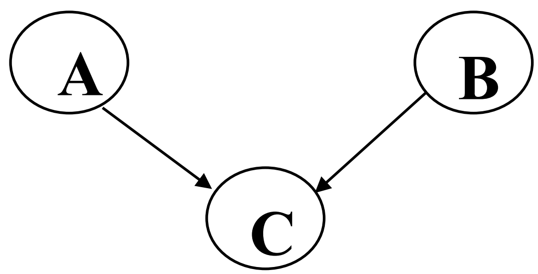


Conditionally independent effects:
$P(A,B,C) = P(B|A)P(C|A)P(A)$

B and C are conditionally independent
Given A

e.g., A is a disease, and we model
B and C as conditionally independent
symptoms given A

Examples of 3-way Bayesian Networks

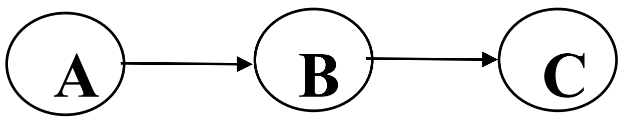


Independent Causes:
$P(A,B,C) = P(C|A,B)P(A)P(B)$

"Explaining away" effect:
Given C, observing A makes B less likely
e.g., earthquake/burglary/alarm example

A and B are (marginally) independent
but become dependent once C is known

Examples of 3-way Bayesian Networks



Markov dependence:
$P(A,B,C) = P(C|B)\, P(B|A)P(A)$

Example

- I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: *Burglary (B)*, *Earthquake (E)*, *Alarm (A)*, *JohnCalls (J)*, *MaryCalls (M)*

- What is P(B | M, J)? (for example)

Example

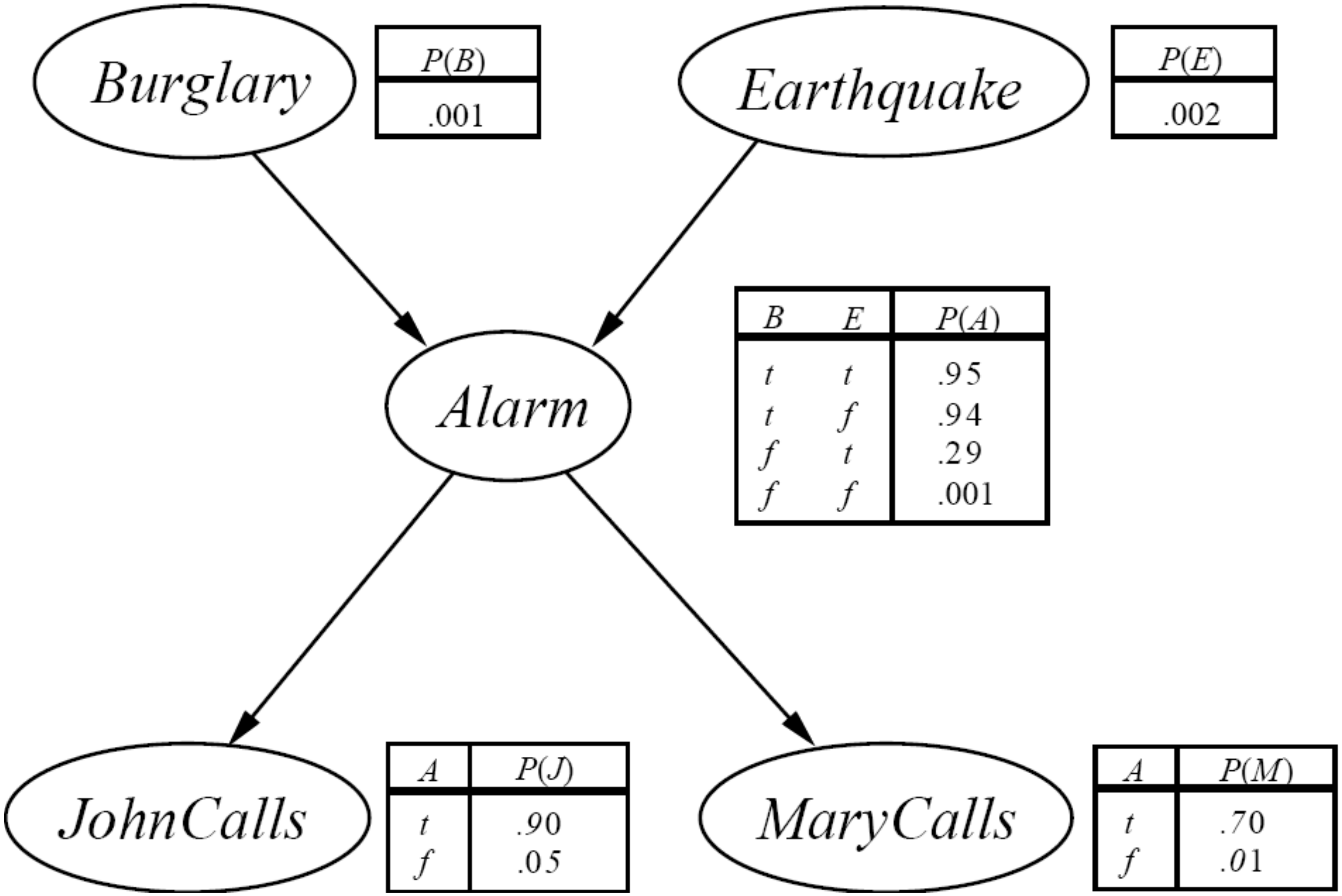
- We can use the full joint distribution to answer this question.
  - Requires $2^5 = 32$ probabilities
  - Can we use prior domain knowledge to come up with a Baysian network that requires fewer probabilities?
- Network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call

Constructing a Baysian Network – Step 1

- Order the variables in terms of causality (may be a partial order), e.g. {E, B} -> {A} -> {J, M}
- $$\begin{aligned} P(J, M, A, E, B) &= P(J, M \mid A, E, B)\ P(A \mid E, B)\ P(E, B) \\ &\approx P(J, M \mid A)\quad P(A \mid E, B)\ P(E)\ P(B) \\ &\approx P(J \mid A) P(M \mid A)\quad P(A \mid E, B)\ P(E)\ P(B) \end{aligned}$$

Conditionally independent assumption

The Resulting Bayesian Network



$P(B)$
.001

$P(E)$
.002

B	E	$P(A)$
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	$P(J)$
t	.90
f	.05

A	$P(M)$
t	.70
f	.01

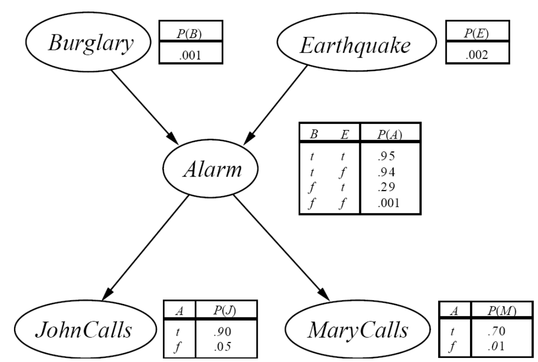
Constructing this Bayesian Network: Step 2

- P(J, M, A, E, B) = P(J | A) P(M | A) P(A | E, B) P(E) P(B)

- There are 3 conditional probability tables (CPTs) to be determined: P(J | A), P(M | A), P(A | E, B)
  - Requiring 2 + 2 + 4 = 8 probabilities

- And 2 marginal probabilities P(E), P(B) -> 2 more probabilities

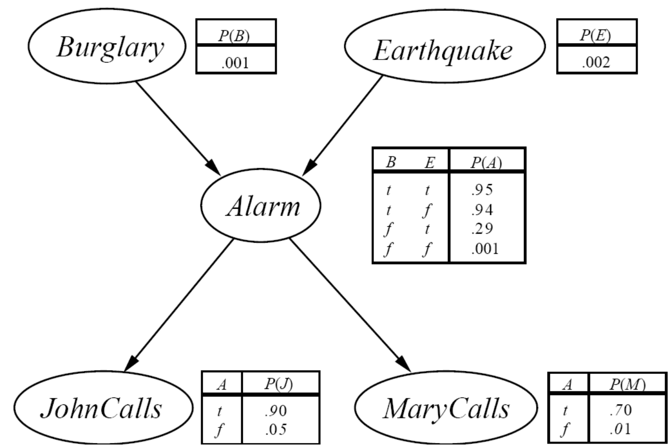
- Where do these probabilities come from?
  - Expert knowledge
  - From data (relative frequency estimates)



Inference (Reasoning) in Bayesian Networks

- Consider answering a query in a Bayesian Network
  - Q = set of query variables
  - e = evidence (set of instantiated variable-value pairs)
  - Inference = computation of conditional distribution P(Q | e)

- Examples
  - P(burglary | alarm)
  - P(earthquake | JCalls, Mcalls)
  - P(JCalls, MCalls | burglary, earthquake)



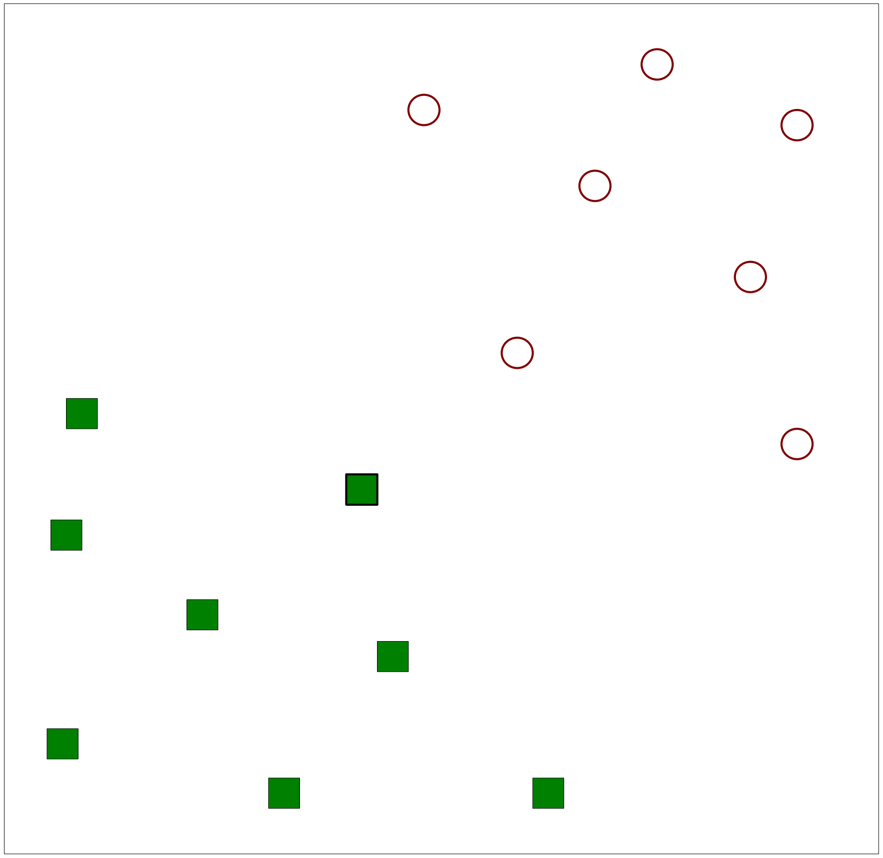
- Can we use the structure of the Bayesian Network to answer such queries efficiently? Answer = yes
  - Generally speaking, complexity is inversely proportional to sparsity of graph

Why Bayesian Classifiers ?

- Captures prior knowledge of a particular domain. Encodes causality.
- Works well with incomplete data.
- Robust to model overfitting.
- Can add new variables easily.
- Probabilistic outputs.
- But lots of time and effort spent in constructing the network.

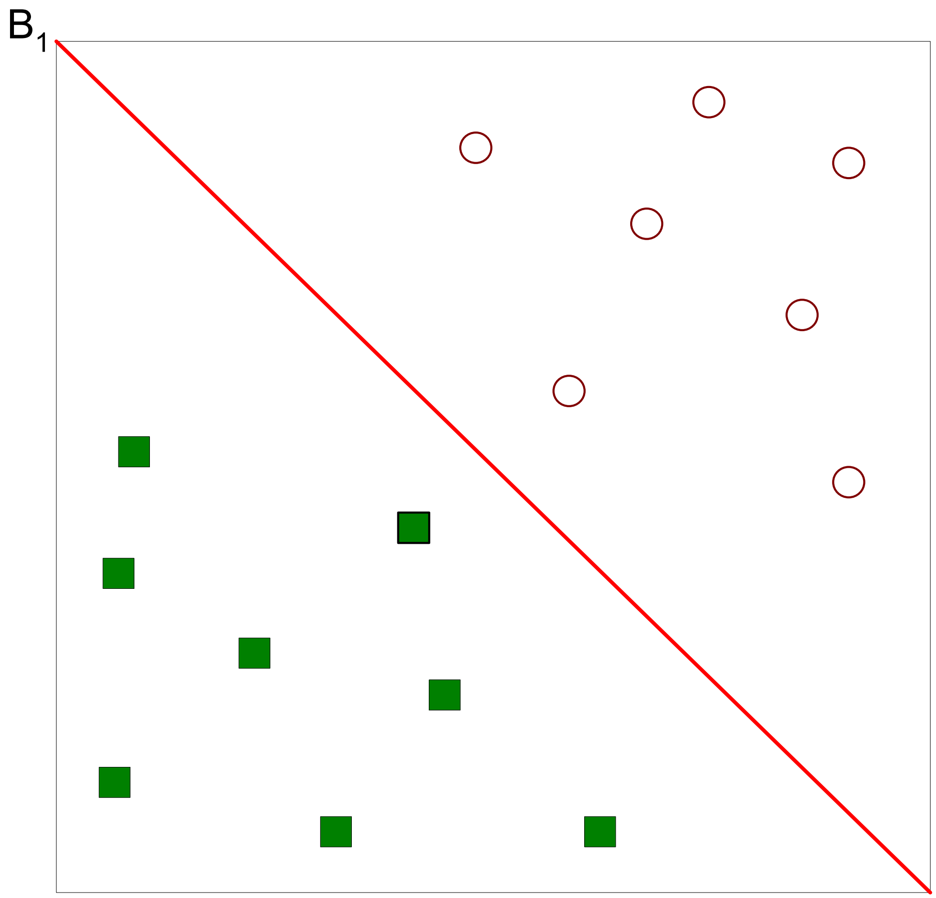
Support Vector Machines

Support Vector Machines



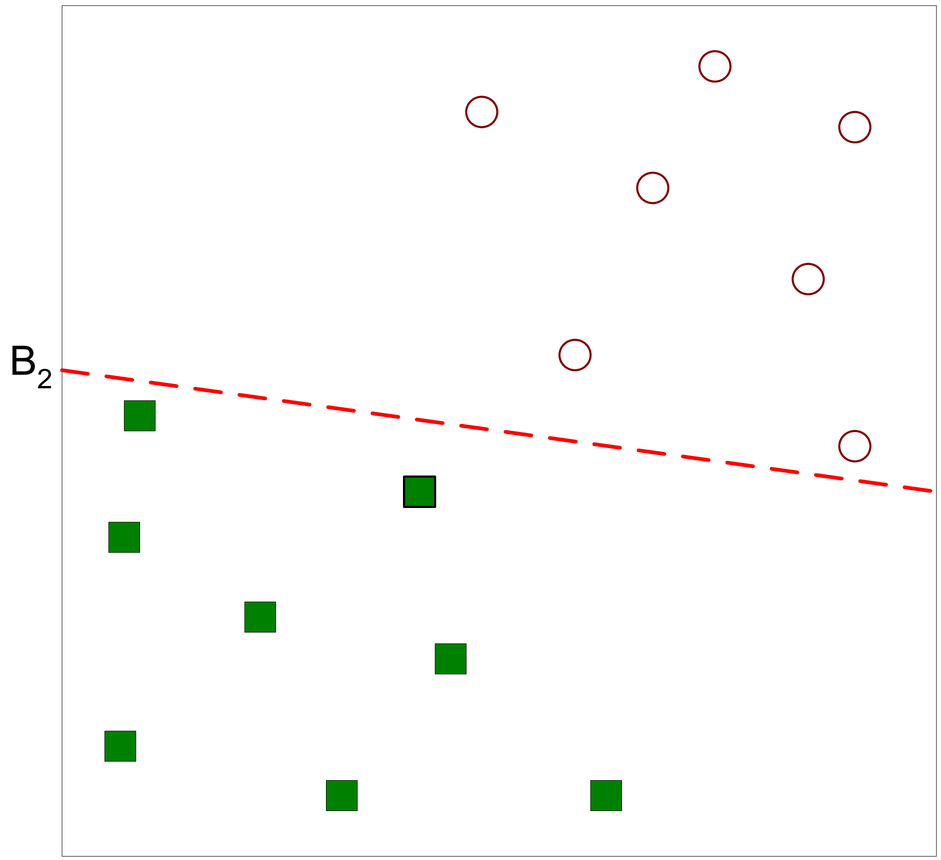
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



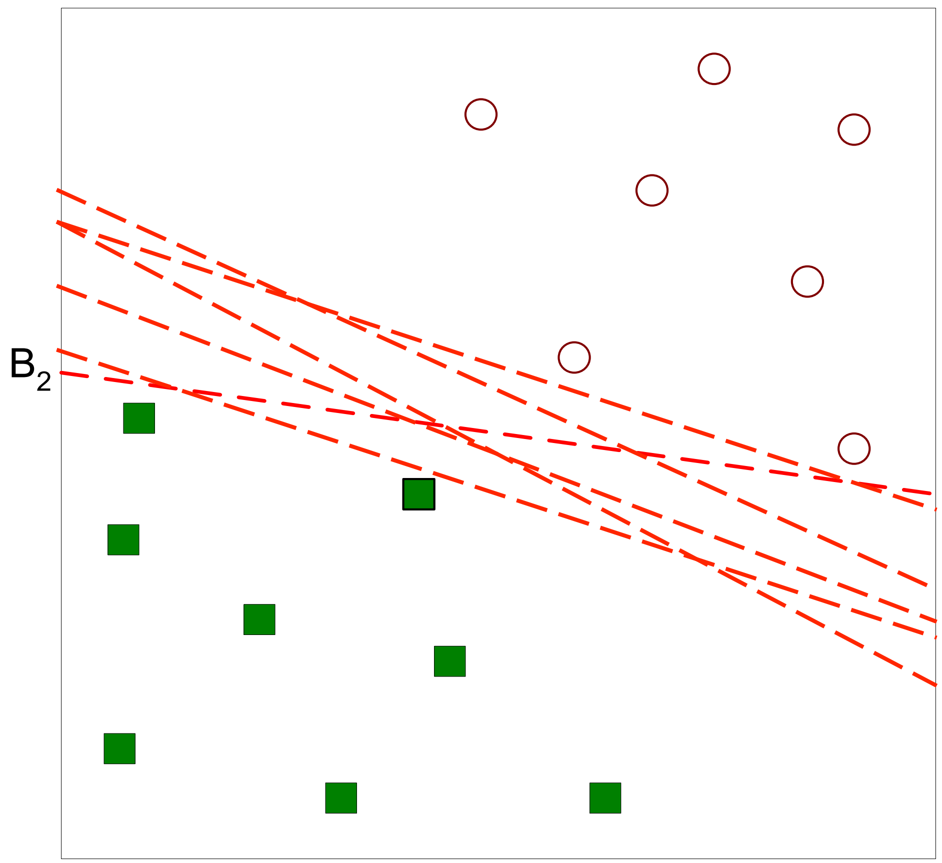
- One Possible Solution

Support Vector Machines



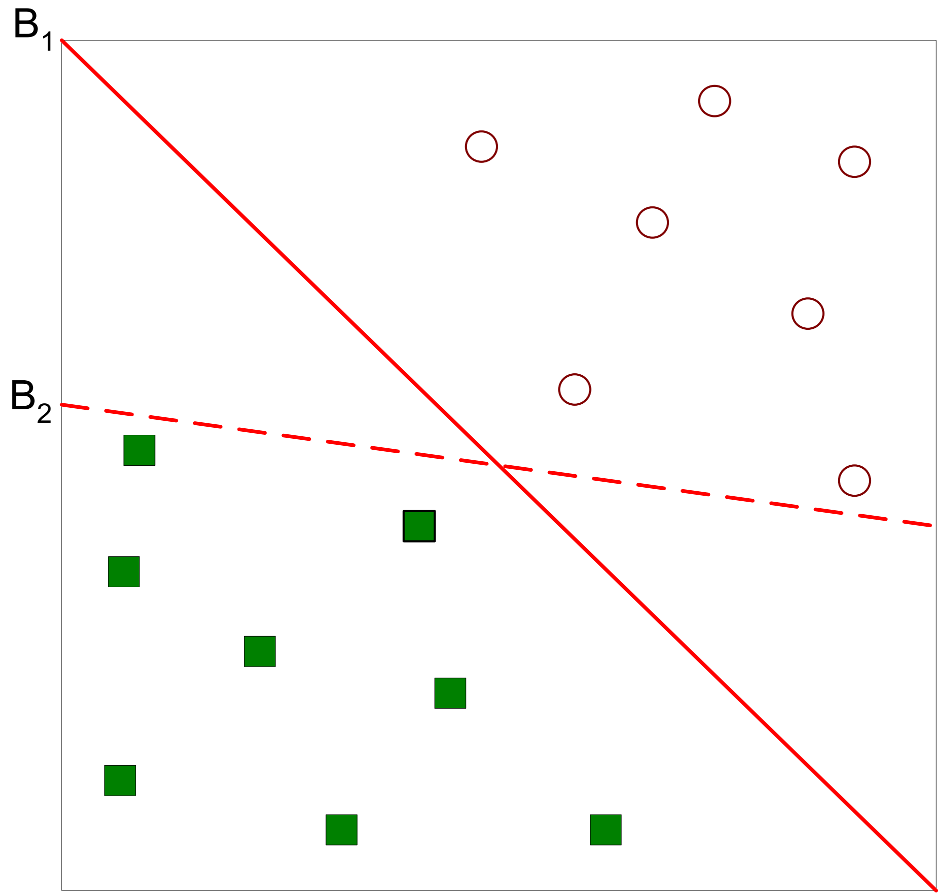
- Another possible solution

Support Vector Machines



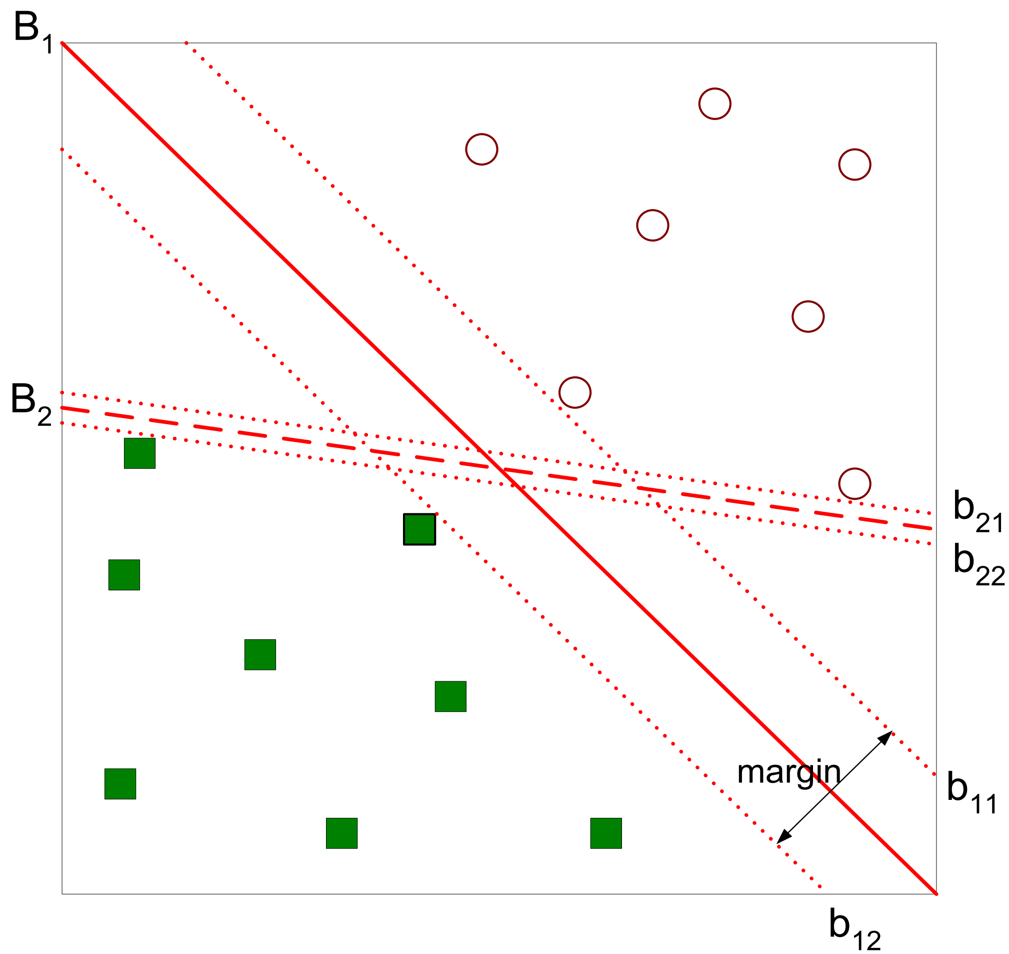
- Other possible solutions

Support Vector Machines



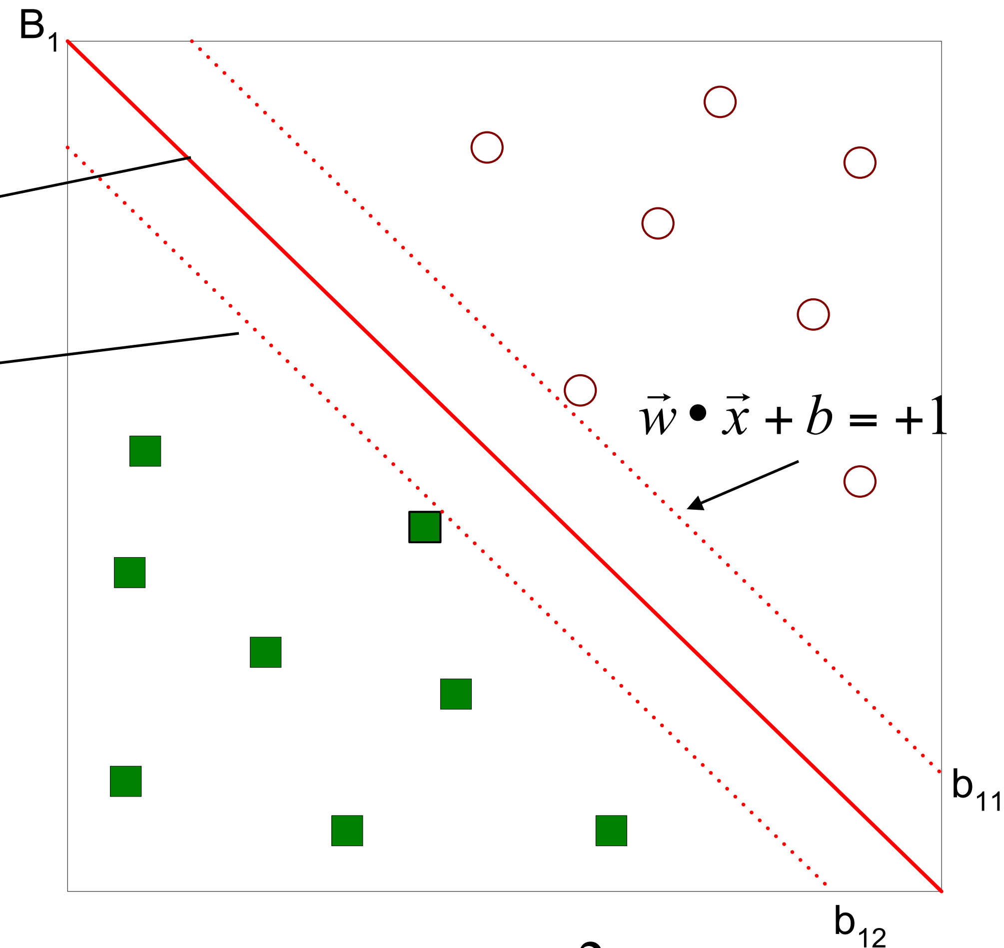
- Which one is better? B1 or B2?
- How do you define better?

Support Vector Machines



- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines



$$\vec{w} \bullet \vec{x} + b = 0$$

$$\vec{w} \bullet \vec{x} + b = -1$$

$$\vec{w} \bullet \vec{x} + b = +1$$

Classify a point into:

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}} + \mathrm{b} \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|^2}$$

Support Vector Machines

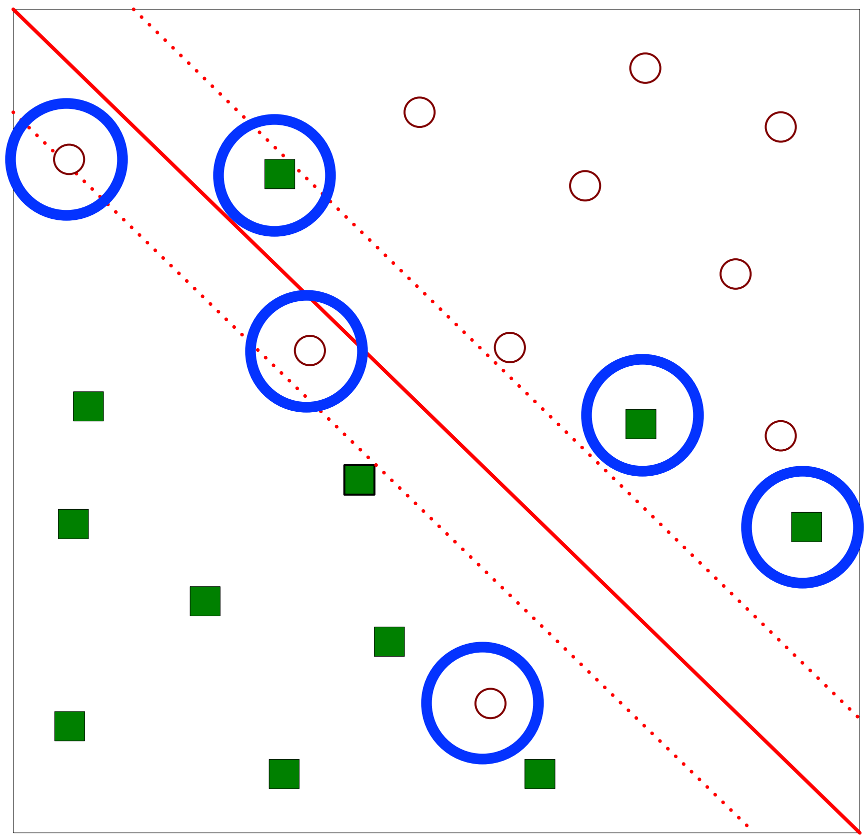
- We want to maximize: $$\text{Margin} = \frac{2}{\|\vec{w}\|^2}$$
  - Which is equivalent to minimizing: $$L(w) = \frac{\|\vec{w}\|^2}{2}$$
  - But subjected to the following constraints:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 \end{cases}$$

- This is a constrained optimization problem
  - Numerical approaches to solve it (e.g., quadratic programming)

Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:

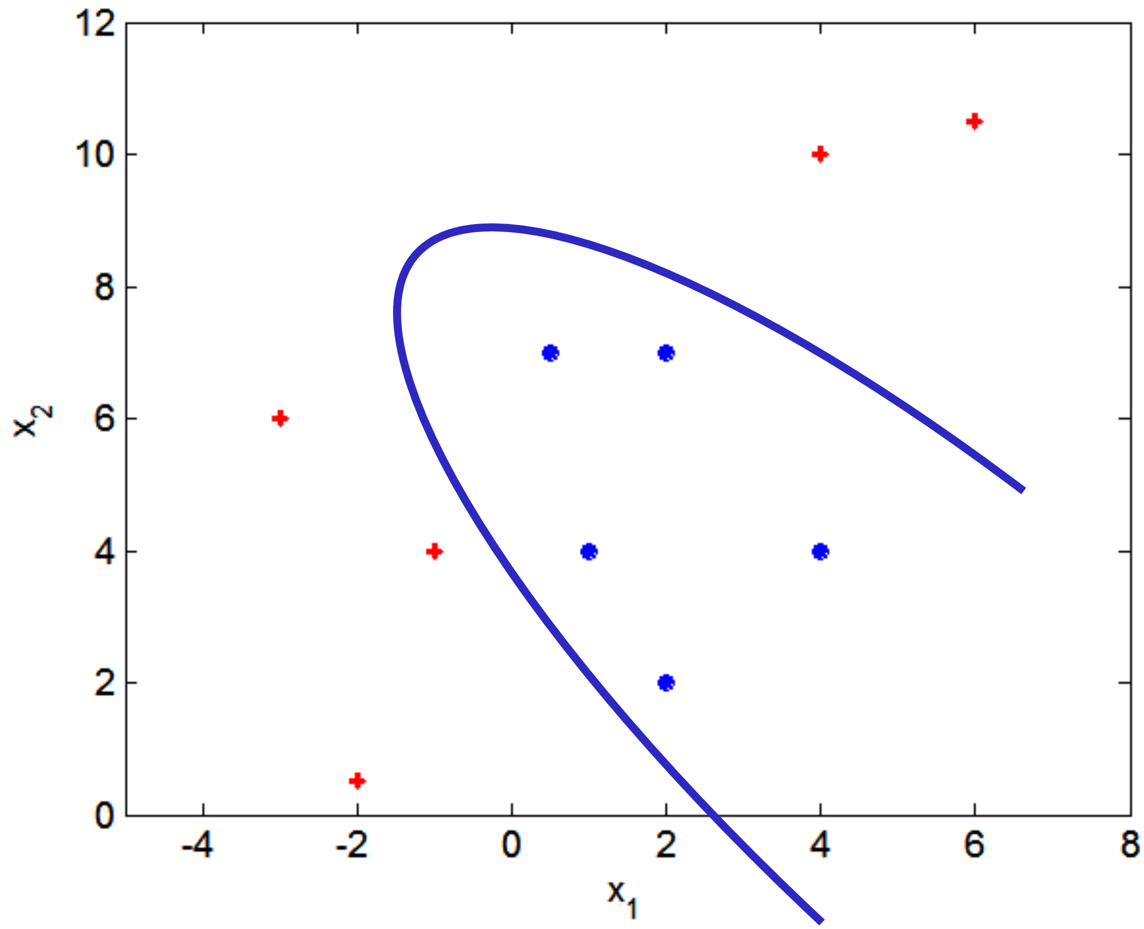
$$L(w) = \frac{\|\vec{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

    - Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \geq 1 - \xi_\mathrm{i} \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \leq -1 + \xi_\mathrm{i} \end{cases}$$

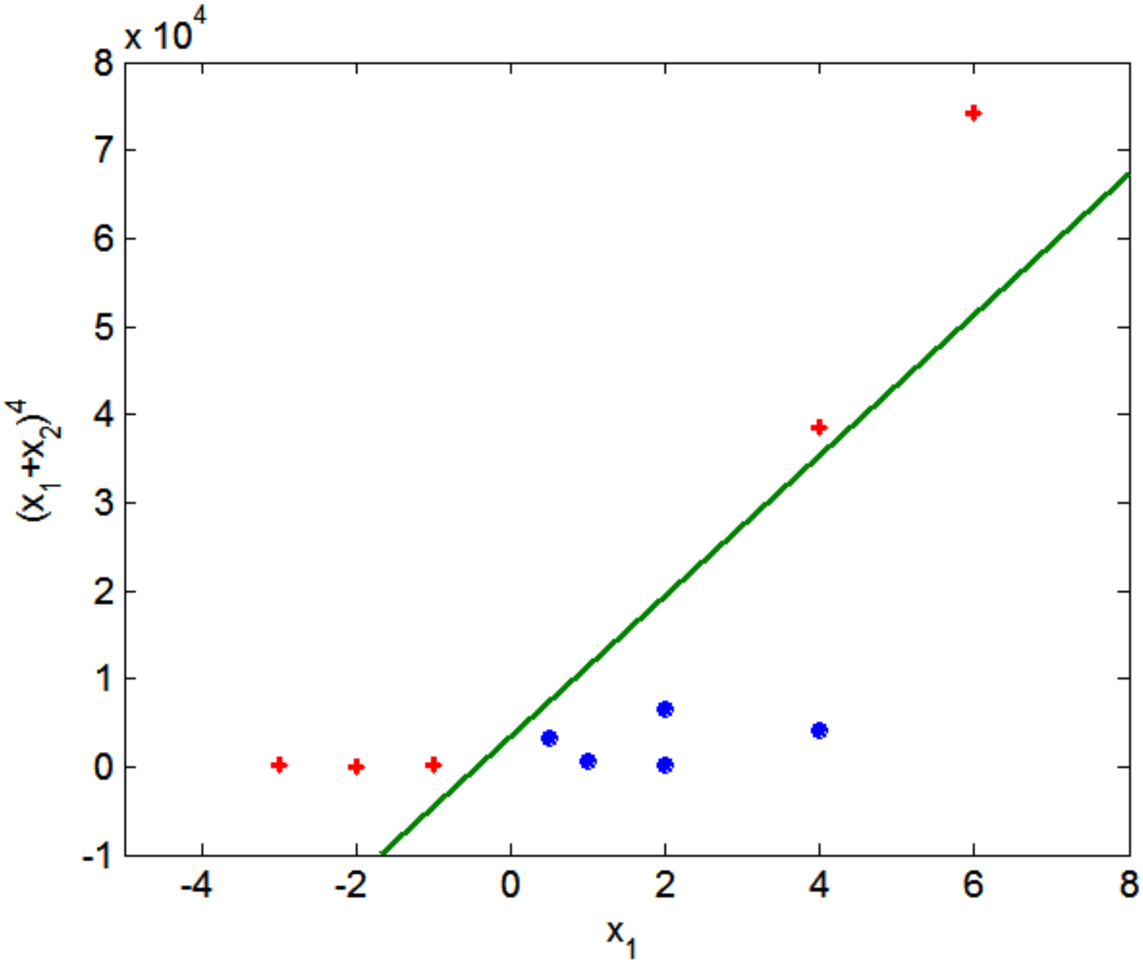
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

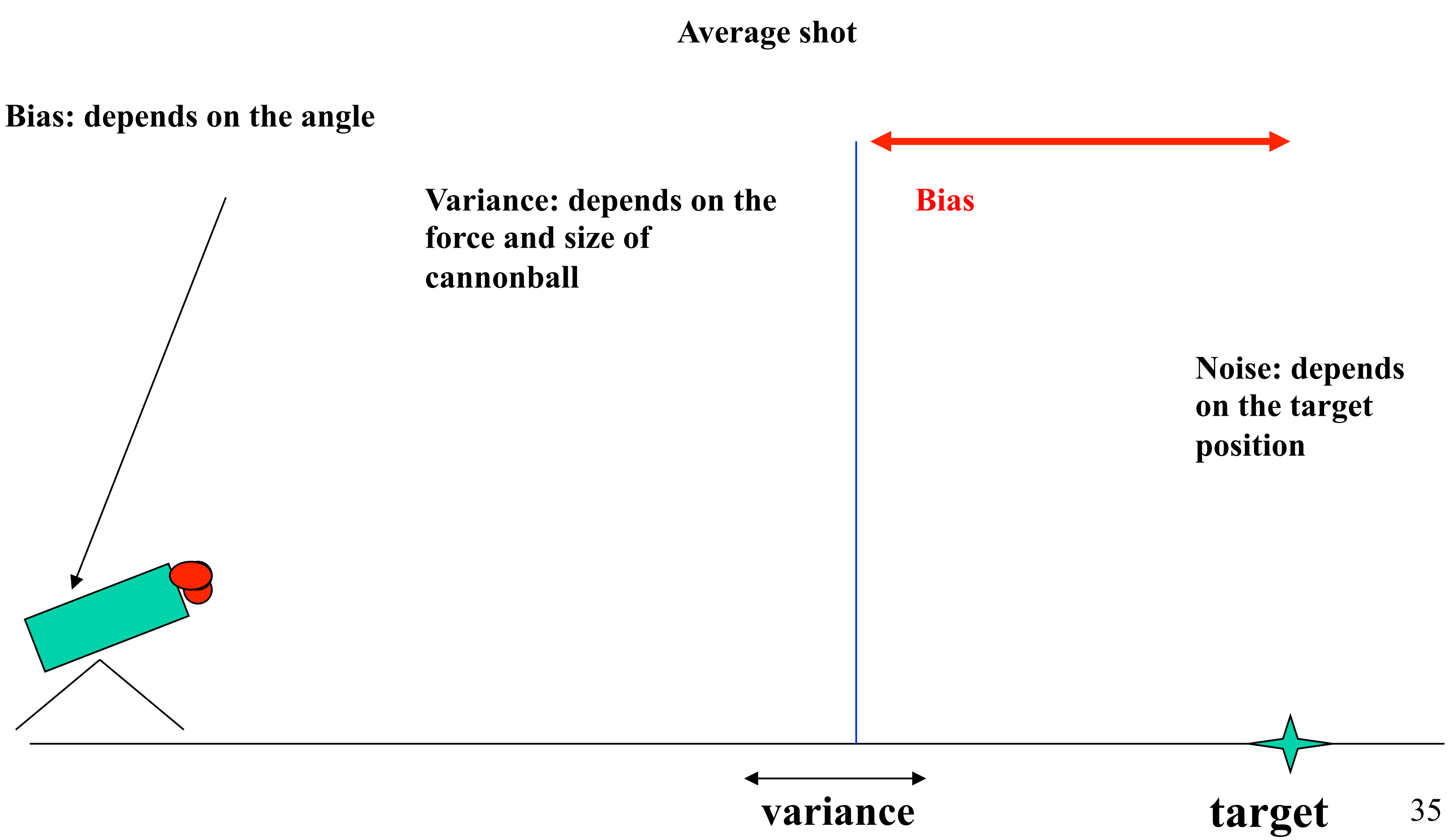
- Transform data into higher dimensional space

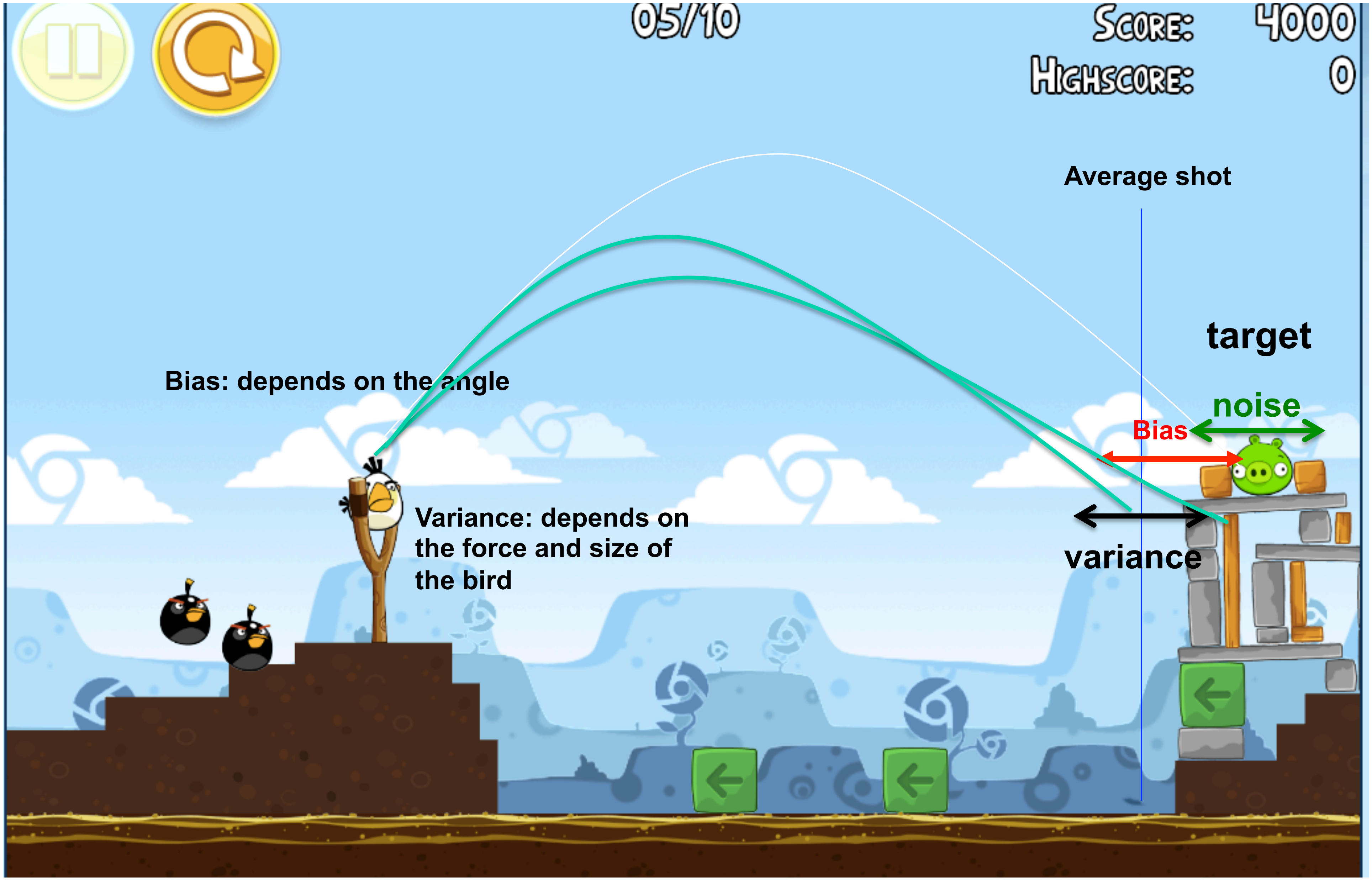


Why SVMs?

- Convex Convex Convex
  - No trapping in local minima
- SVMs work for categorical and continuous data.
- Can control the model complexity by providing the control on cost function, margin parameters to use.
- Kernel Trick (Not discussed) extends it to non-linear spaces.

Loss, bias, variance and noise




05/10
SCORE: 4000
HIGHSCORE: 0
Average shot
target
Bias: depends on the angle
noise
Bias
Variance: depends on
the force and size of
the bird
variance

Example: Bias

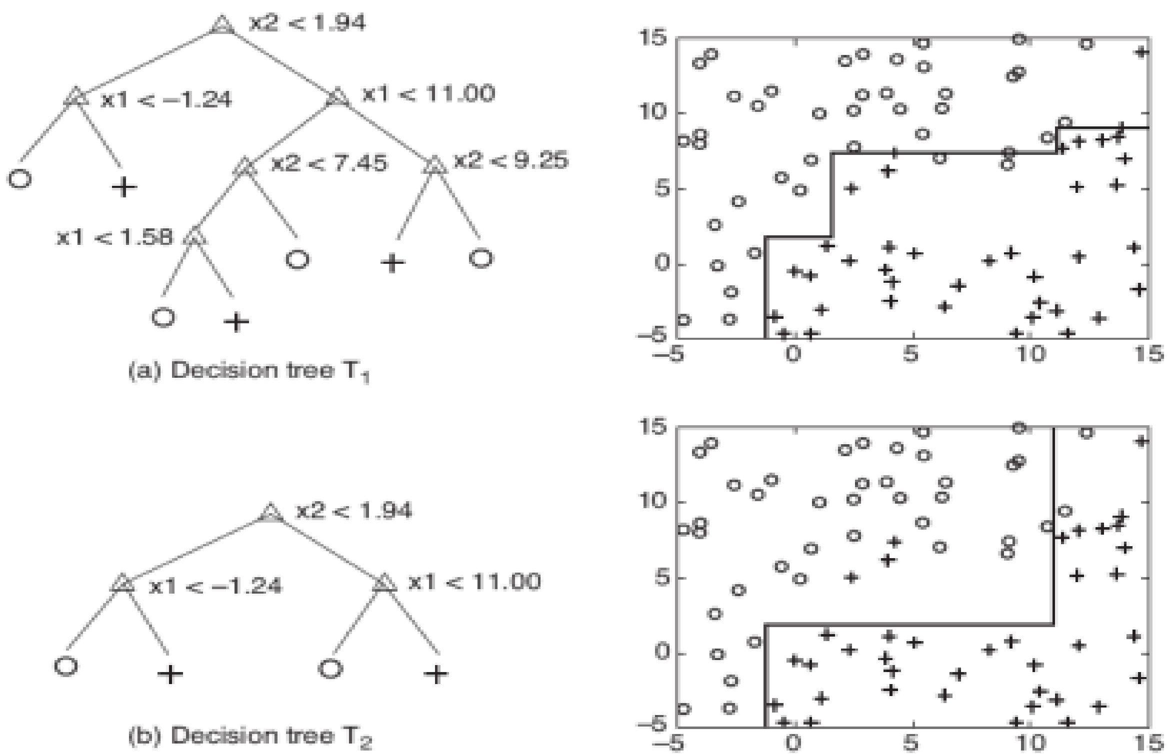
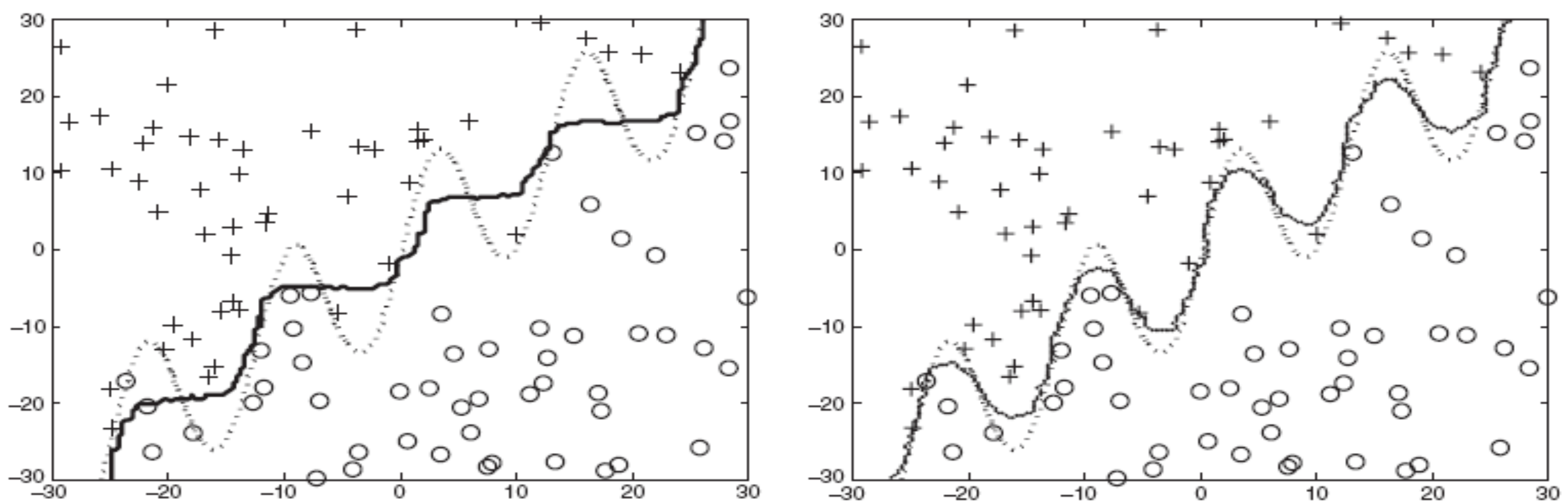


Figure 5.33. Two decision trees with different complexities induced from the same training data.

Bias-Variance (Generalize)



(a) Decision boundary for decision tree.

(b) Decision boundary for 1-nearest neighbor.

Figure 5.34. Bias of decision tree and 1-nearest neighbor classifiers.

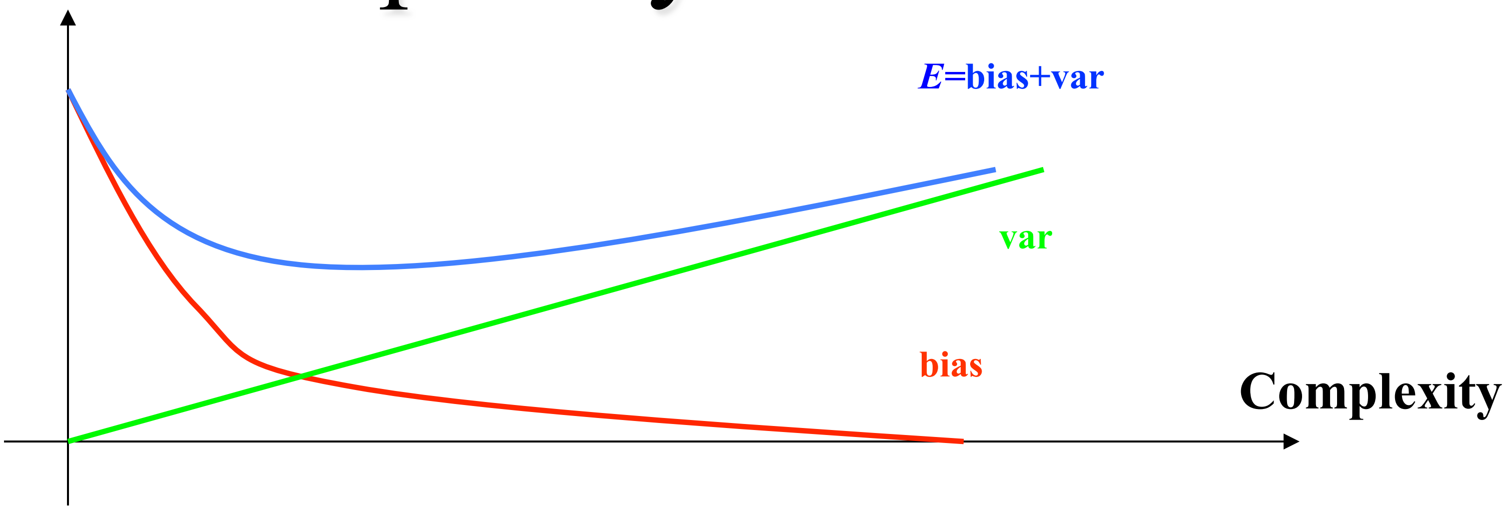
For better generalizable model

- Minimize both bias and variance
- However,
  - Neglect the input data and predict the output to be a constant value gives “zero” variance but high bias.
  - On the other hand, perfectly interpolate the given data to produce f=f* - implies zero bias but high variance.

Model Complexity

- Simple models of low complexity
  - high bias, small variance
  - potentially rubbish, but stable predictions
- Flexible models of high complexity
  - small bias, high variance
  - over-complex models can be always massaged to exactly explain the observed training data
- What is the right level of model complexity?
  - The problem of model selection

Complexity of the model



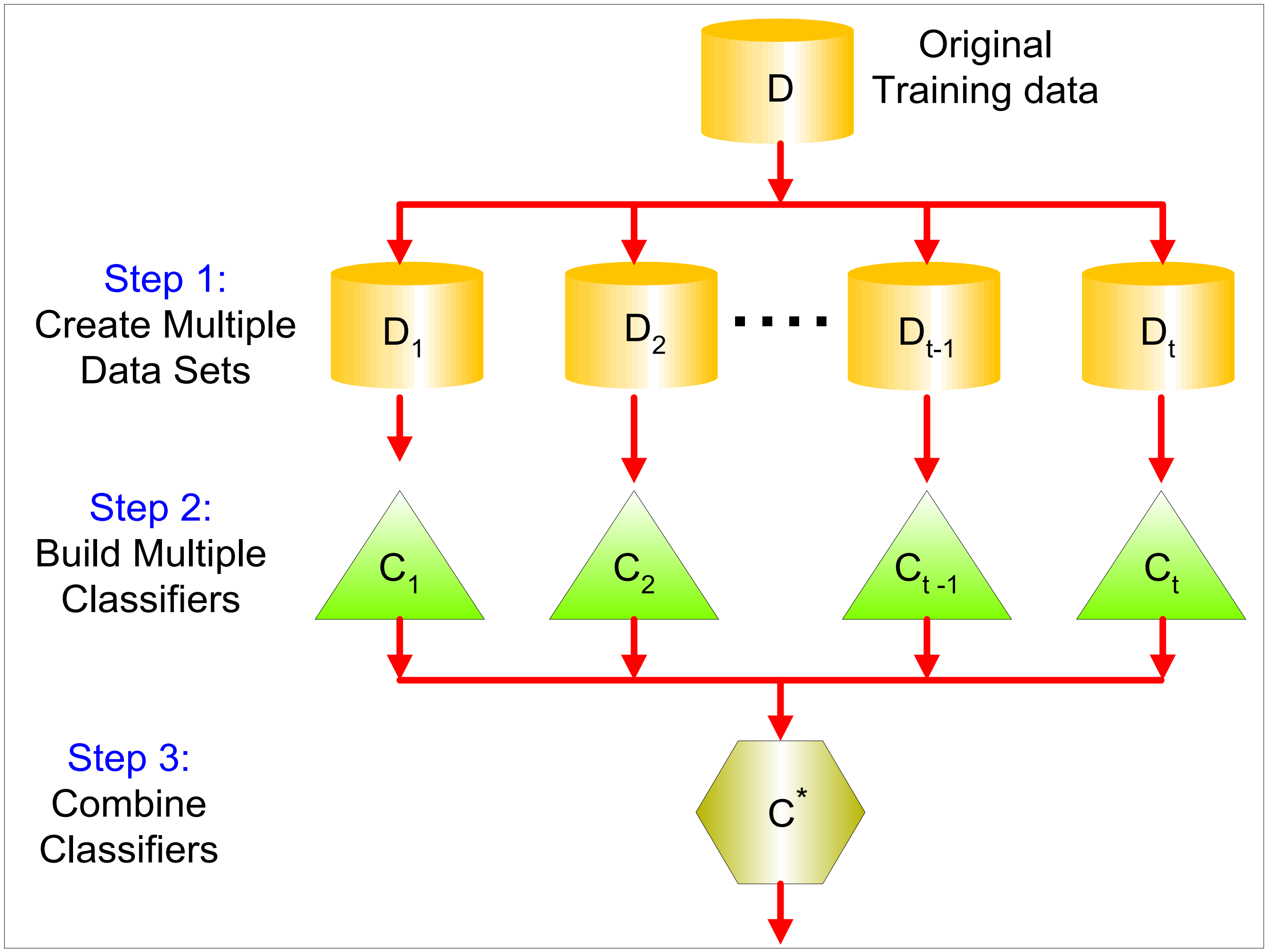
Usually, the bias is a decreasing function of the complexity, while variance is an increasing function of the complexity.

Ensemble Methods

- Construct a set of classifiers from the training data

- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

General Idea



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Bagging

  - Boosting

Bagging

- Bootstrap Aggregation
  - Create classifiers by drawing samples of size equal to the original dataset. (Appx 63% of data will be chosen)
  - Learn classifier using these samples.
  - Vote on them.
- Why does this help ?
  - If there is a high variance i.e. classifier is unstable, bagging will help to reduce errors due to fluctuations in the training data.
  - If the classifier is stable i.e. error of the ensemble is primarily by bias in the base classifier -> may degrade the performance.

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  – Initially, all N records are assigned equal weights
  – Unlike bagging, weights may change at the end of boosting round

Adaboost (Freund et. al. 1997)

- Given a set of n class-labeled tuples (x_1,y_1) … (x_n,y_n) i.e T
- Initially all weights of tuples are set to same (1/n)
- Generate k classifiers in k rounds. At the i-th round
  - Tuples from T are sampled from T to form training set T_i
  - Each tuple's chance of selection depends on its weight.
  - Learn a model M_i from T_i
  - Compute error rate using T_i
  - If tuple is misclassified its weight is increased.
- During prediction use the error of the classifier as a weight (vote) on each of the models

Why boosting/bagging?

- Improves the variance of unstable classifiers.
  - Unstable Classifiers
    - Neural nets, decision trees
  - Stable Classifiers
    - K-NN
- May lead to results that are not explanatory.