Support Vector Machines: An Introduction

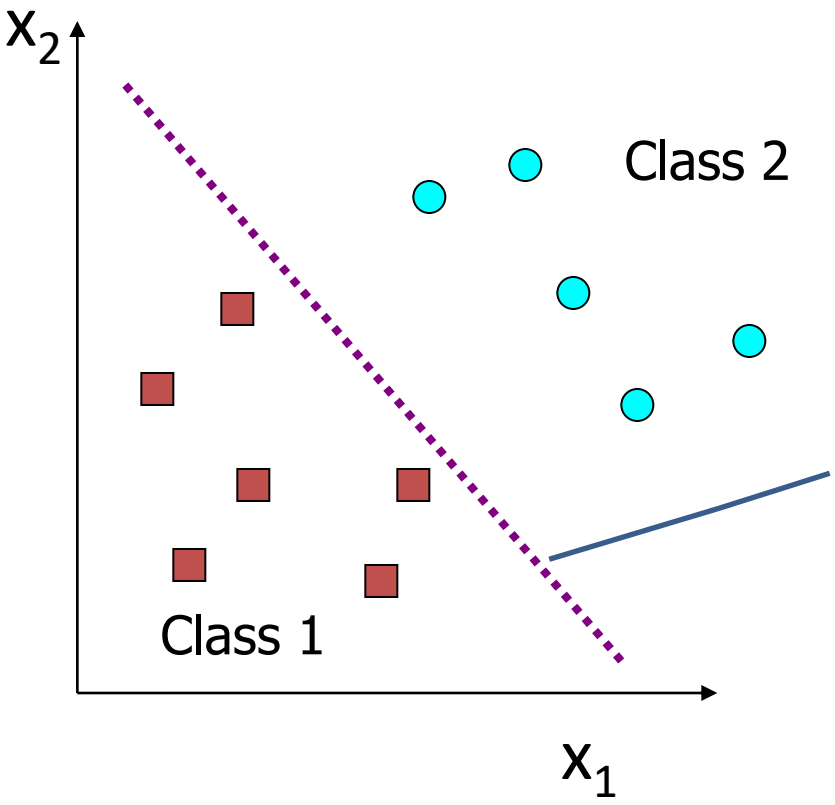
Mostly informal

Linear Classifier

Classifier:

If $f(x_1,x_2) < 0$ assign Class 1;

If $f(x_1,x_2) > 0$ assign Class 2;



$$f(x_1,x_2) = w_1x_1+w_2x_2+b = 0$$

Perceptron

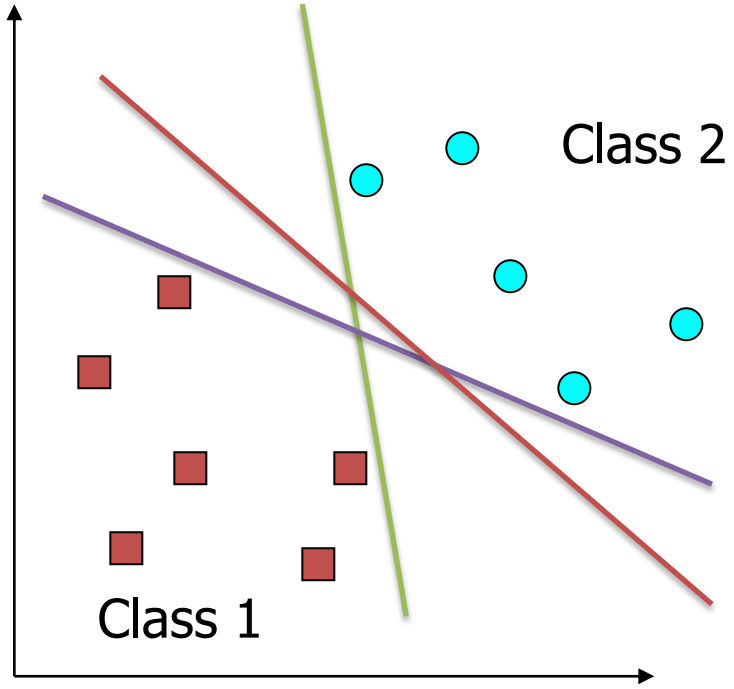
- Perceptron is the name given to the linear classifier.
- If there exists a Perceptron that correctly classifies all training examples, then we say that the training set is linearly separable.
- Different Perceptron learning techniques are available.

Learning a linear classifier

- A linear classifier can be learnt even when the data is NOT a linearly separable one.
- We can define a criterion J and try to minimize this which corresponds to the learnt linear classifier.
- A linear regression method can be directly used.

Perceptron – Let us begin with linearly separable data

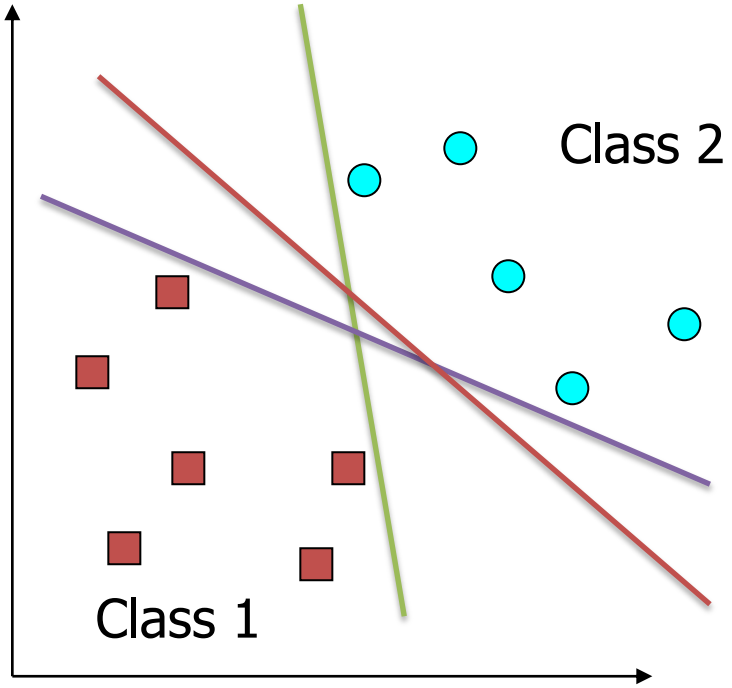
- If the data is linearly separable then many Perceptrons are possible that correctly classifies the training set.



All being doing equally good on training set, which one is good on the unseen test set?

Perceptron – Let us begin with linearly separable data

- If the data is linearly separable then many Perceptrons are possible that correctly classifies the training set.

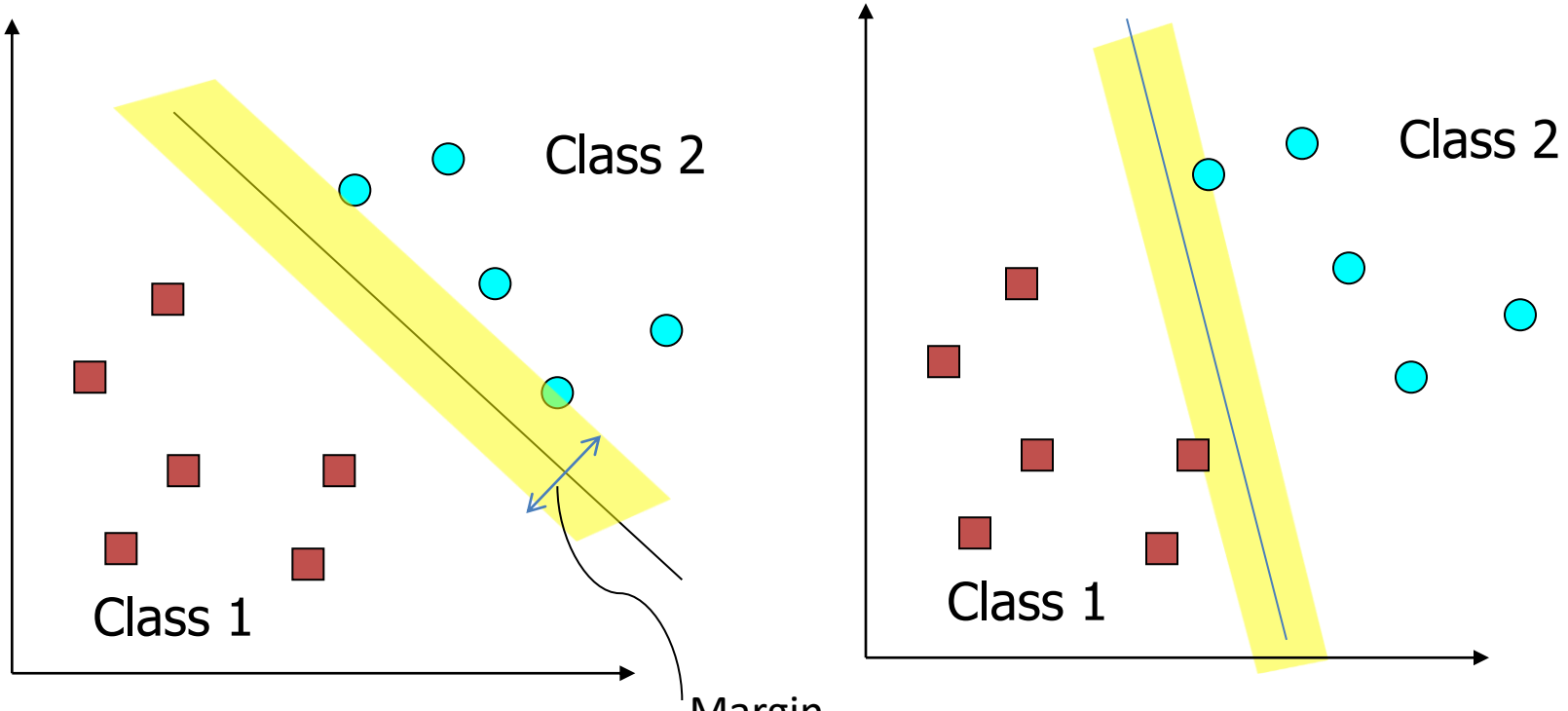


All being doing equally good on training set, which one is good on the unseen test set?

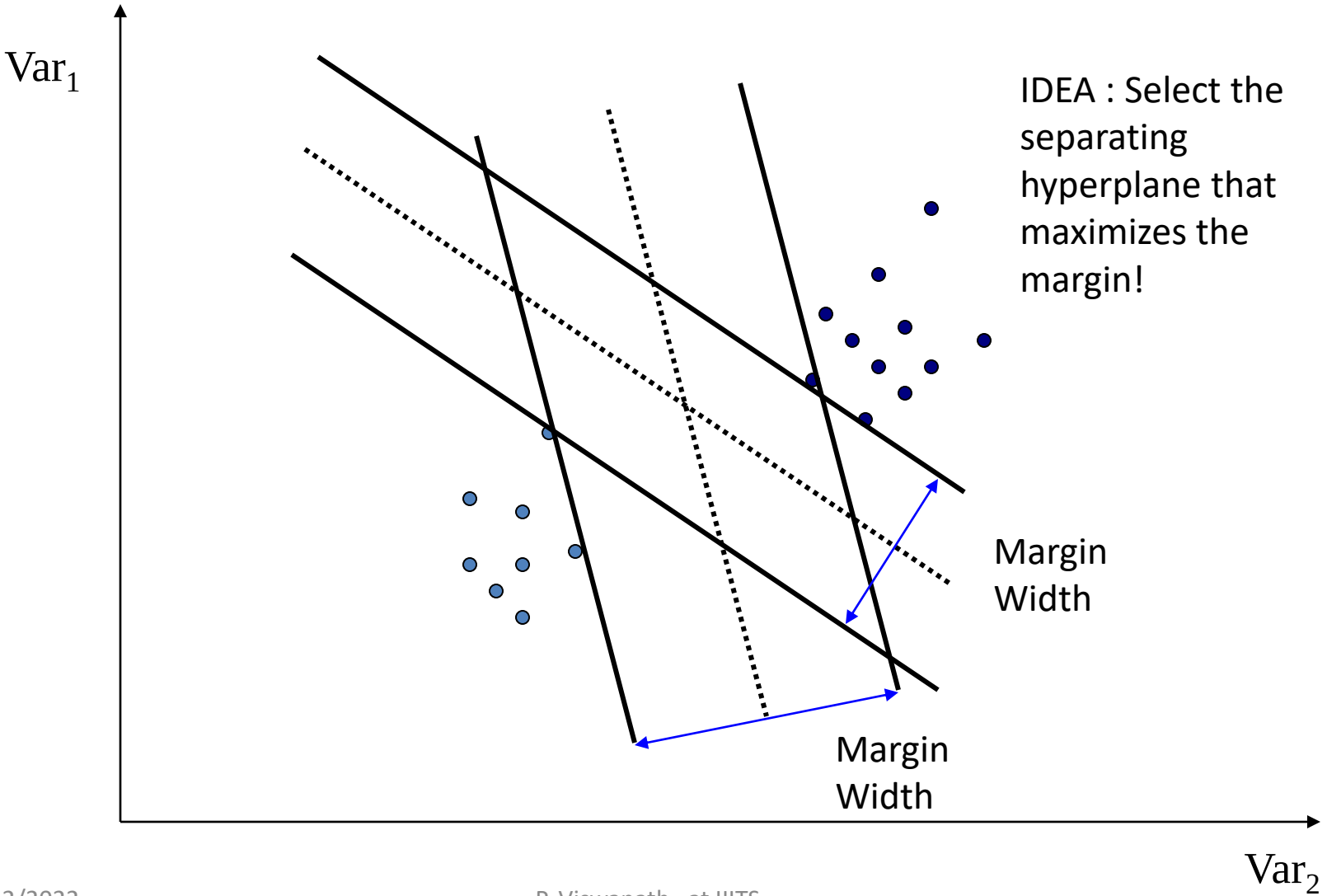
The solution we are getting depends on the starting point i.e., the parameter vector we chose to begin the gradient descent like method.

Hard Linear SVM

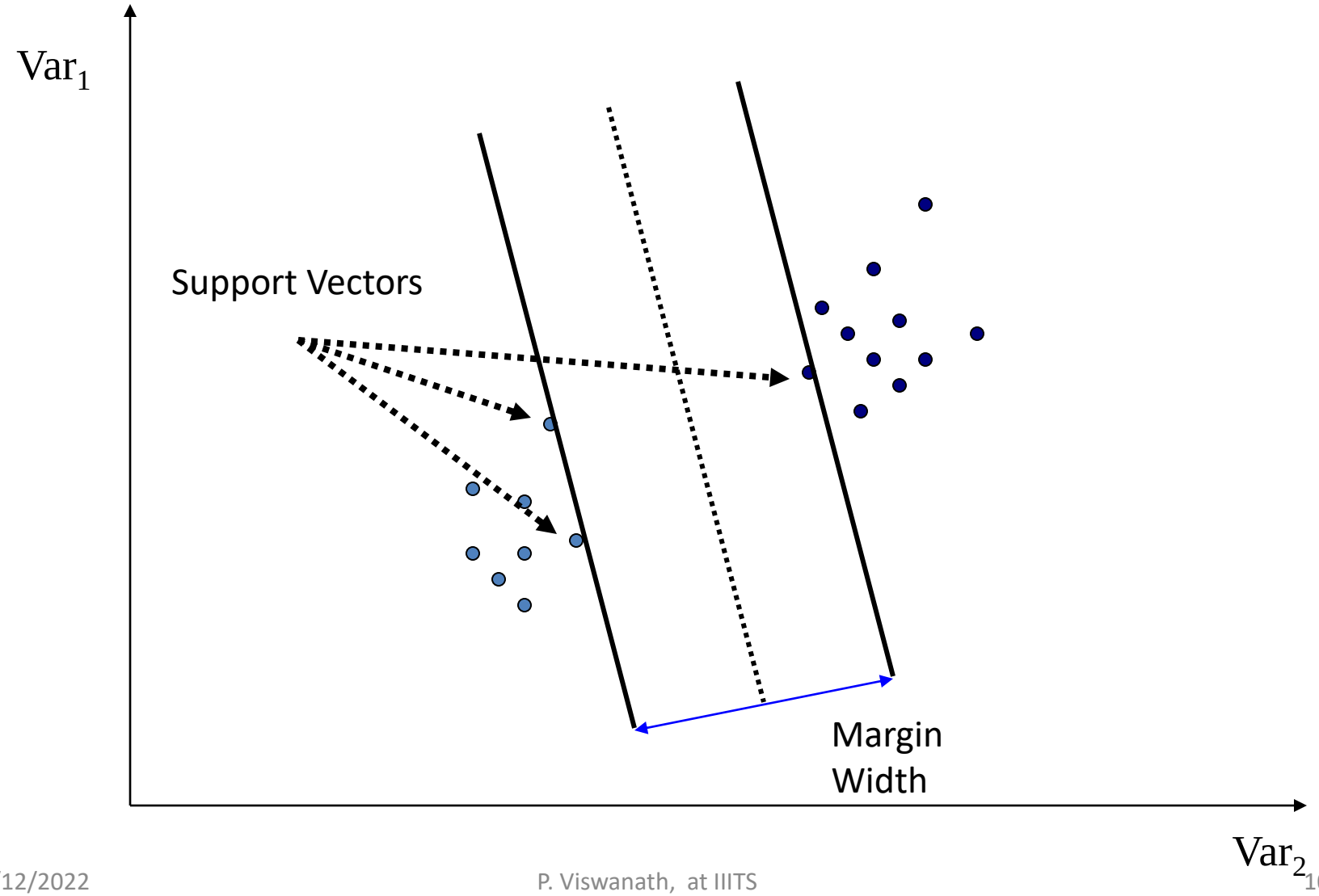
- The best perceptron for a linearly separable data is called “hard linear SVM”.
- For each linear function we can define its margin.
- That linear function which has the maximum margin is the best one.


Class 2
Class 1
Margin
Class 2
Class 1

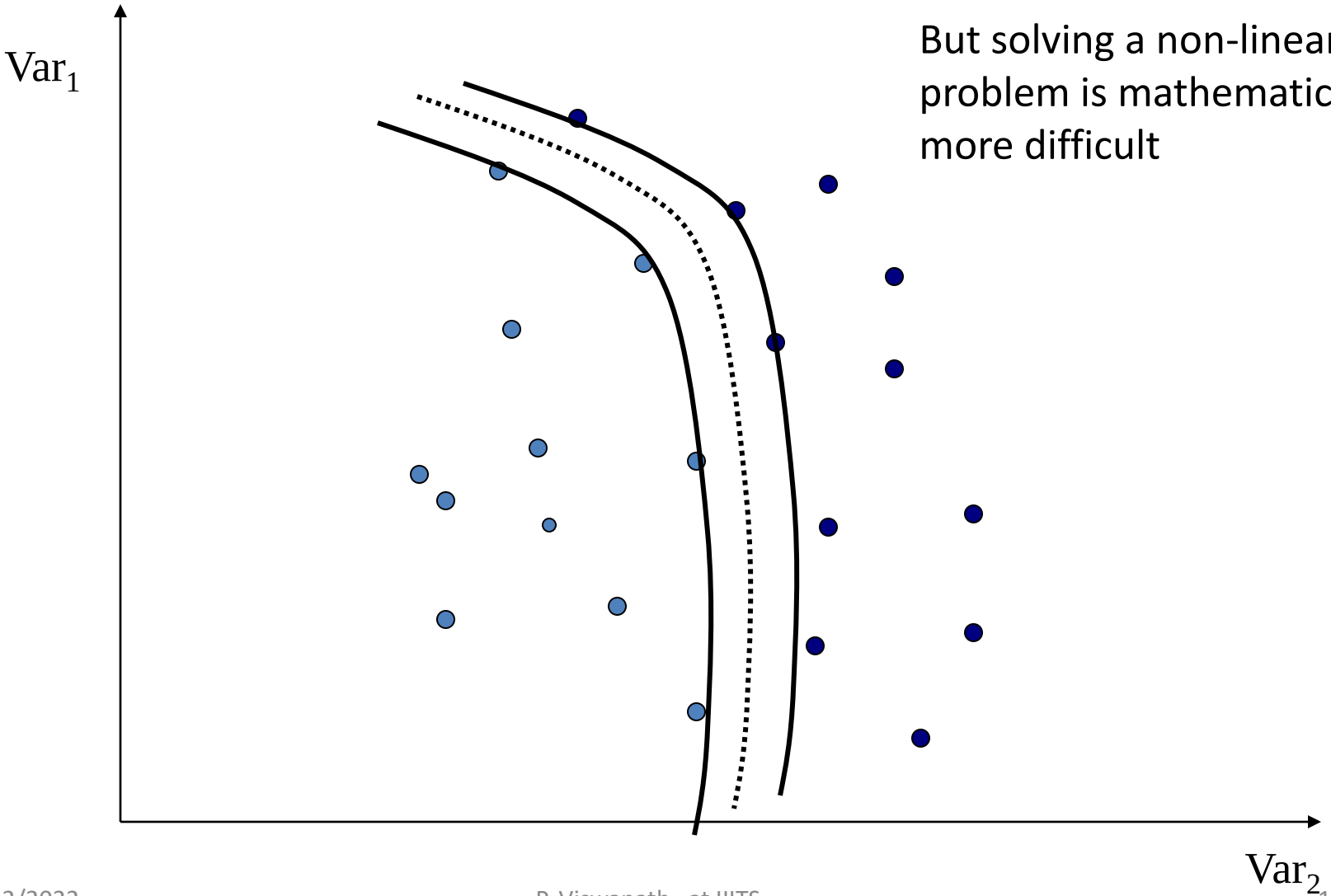
Maximizing the Margin



Support Vectors

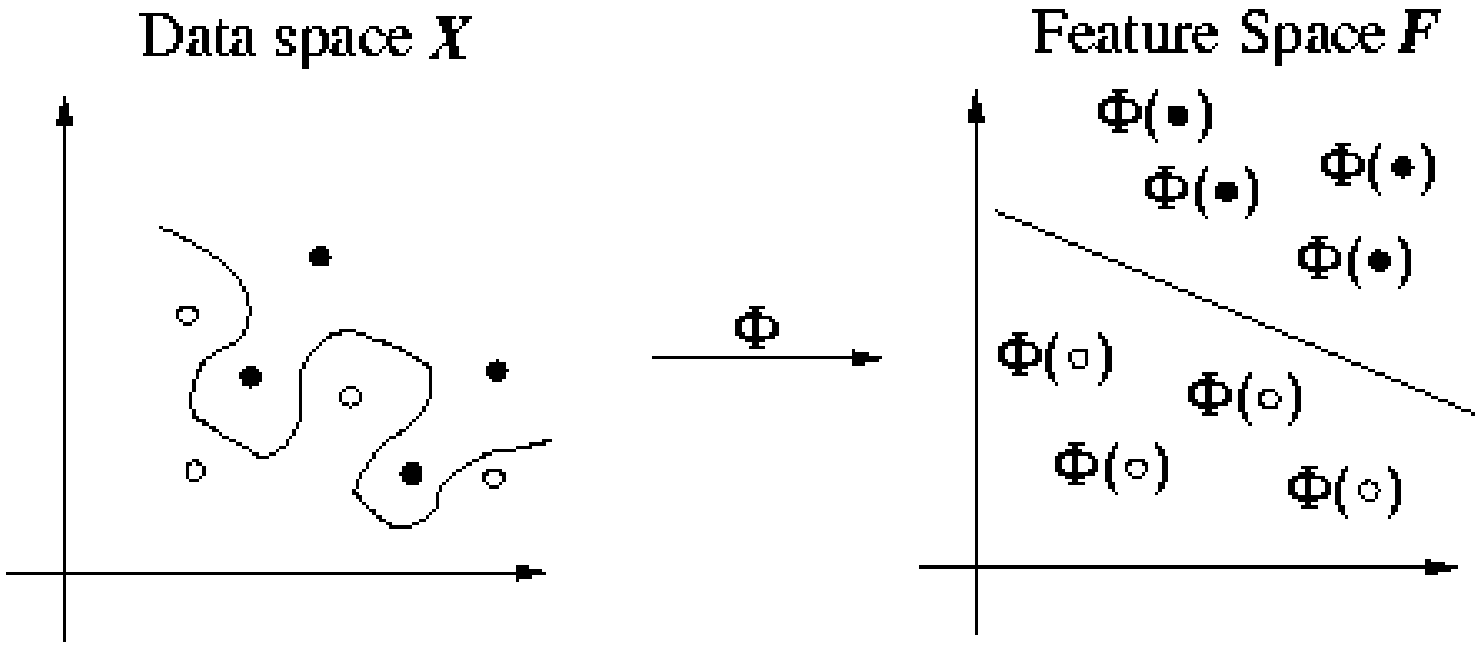


What if the data is not linearly separable?



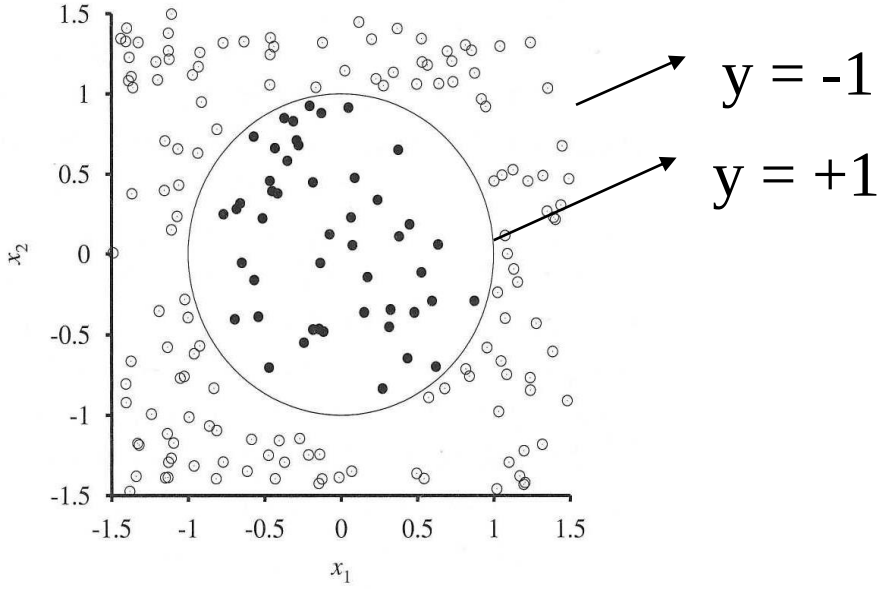
But solving a non-linear problem is mathematically more difficult

Kernel Mapping

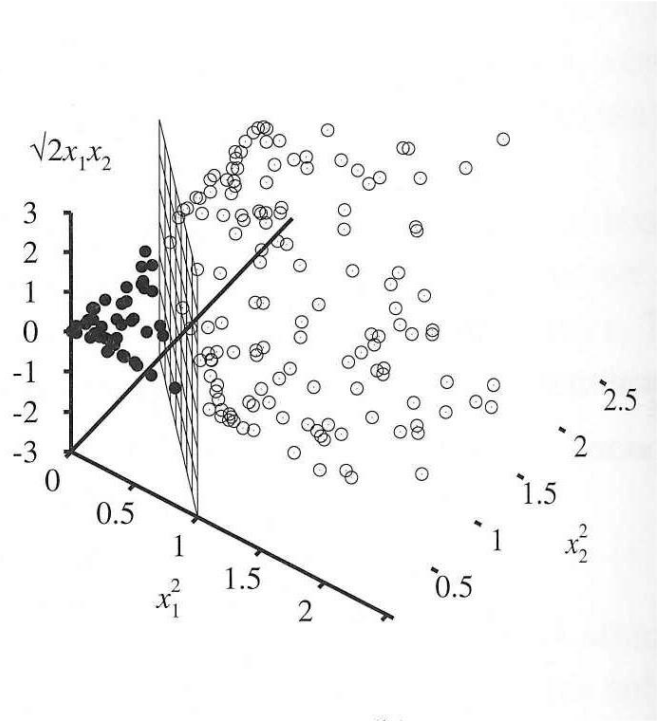


An example

Input Space



Feature Space



The Trick !!

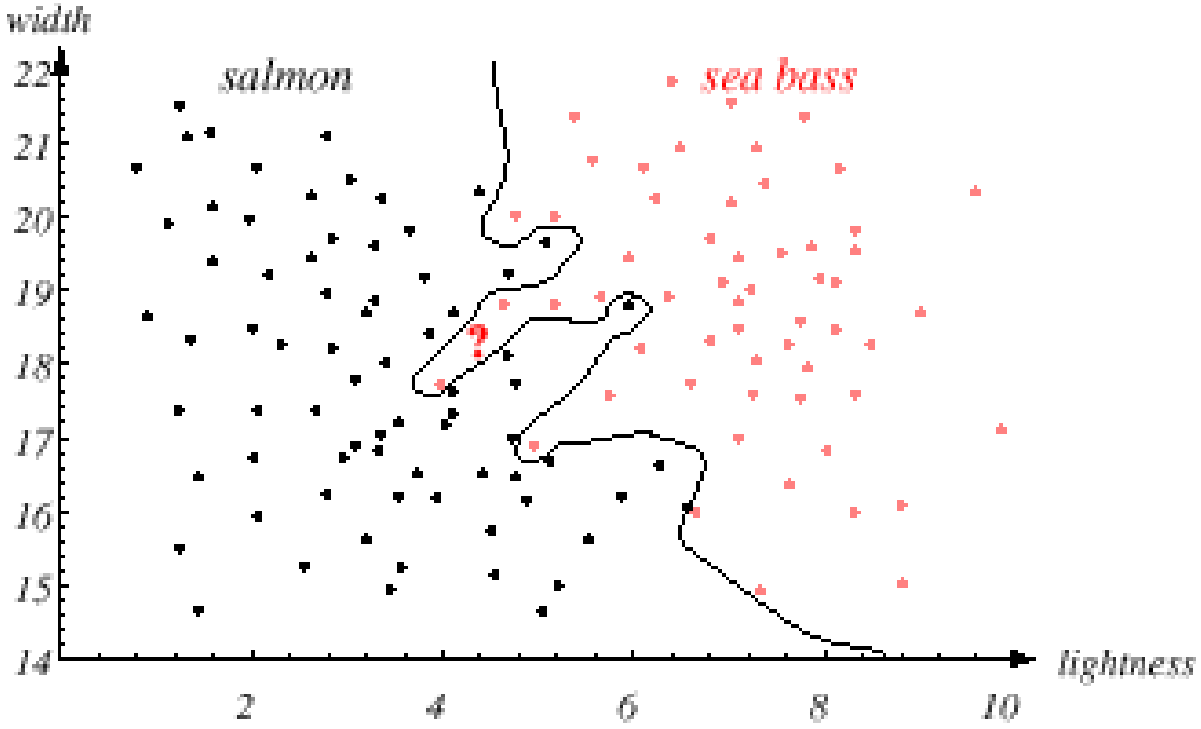
- There is no need to do this mapping explicitly.
- For some mappings, the dot product in the feature space can be expressed as a function in the Input space.
- $\varphi(X_1) \cdot \varphi(X_2) = k(X_1, X_2)$

Eg: Consider a two dimensional problem with $X = (x_1, x_2)^t$. Let $\phi(X) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2)^t$. Then $\phi(X_i) \cdot \phi(X_j) = \kappa(X_i, X_j) = (X_i \cdot X_j)^2$.

- So, if the solution is going to involve only dot products then it can be solved using kernel trick (of course, appropriate kernel function has to be chosen).

- The problem is, with powerful kernels like “Gaussian kernel” it is possible to learn a non-linear classifier which does extremely well on the training set.

Discriminant functions: non-linear



This makes zero mistakes with the training set.

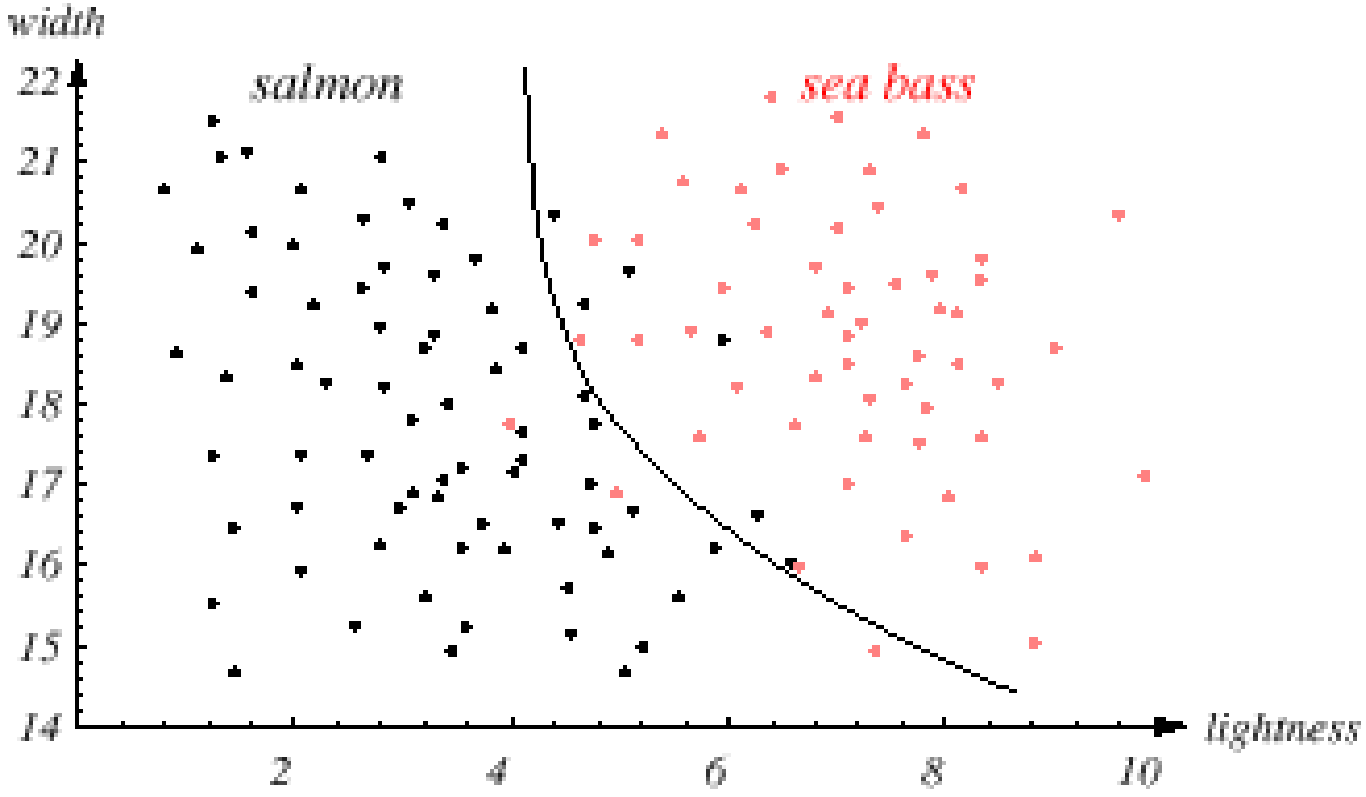
Other important issues …

- This classifier is doing very well as for the training data is considered.
- But this does not guarantee that the classifier works well with a data element which is not there in the training set (that is, with unseen data).
- This is overfitting the classifier with the training data.
- May be we are respecting noise also (There might be mistakes while taking the measurements).
- The ability “to perform better with unseen test patterns too” is called the generalization ability of the classifier.

Generalization ability

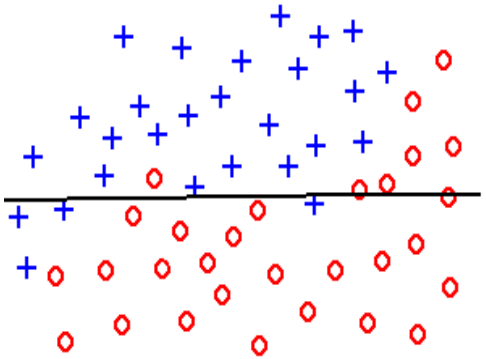
- This is discussed very much.
- It is argued that the simple one will have better generalization ability (eg: Occam's razor: Between two solutions, if everything else is same then choose the simple one).
- How to quantify this?
- *(Training error + a measure of complexity)* should be taken into account while designing the classifier.
- Support vector machines are proved to have better generalization ability.

Discriminant functions …



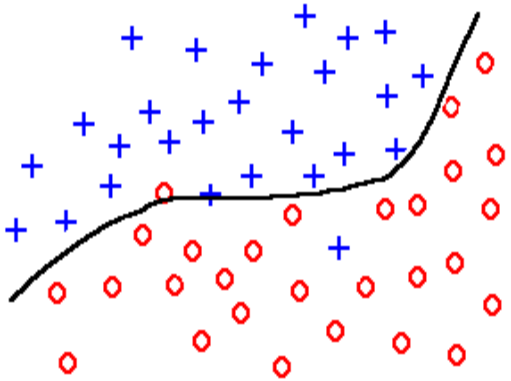
- This has some training error, but this is a relatively simple one.

Overfitting and underfitting



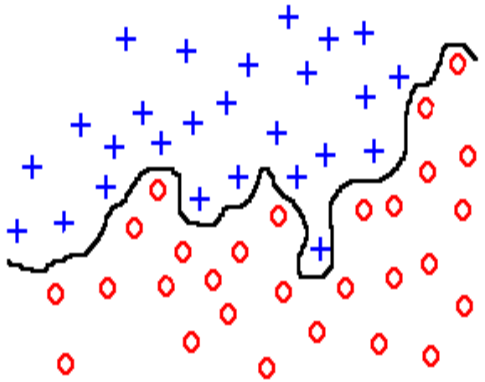
underfitting

High bias,
Low variance



good fit

Low bias,
Low variance



overfitting

Low bias,
High variance

Soft SVM

- Allow for some mistakes with the training set !
- But, this is to achieve a better margin.

Soft SVM