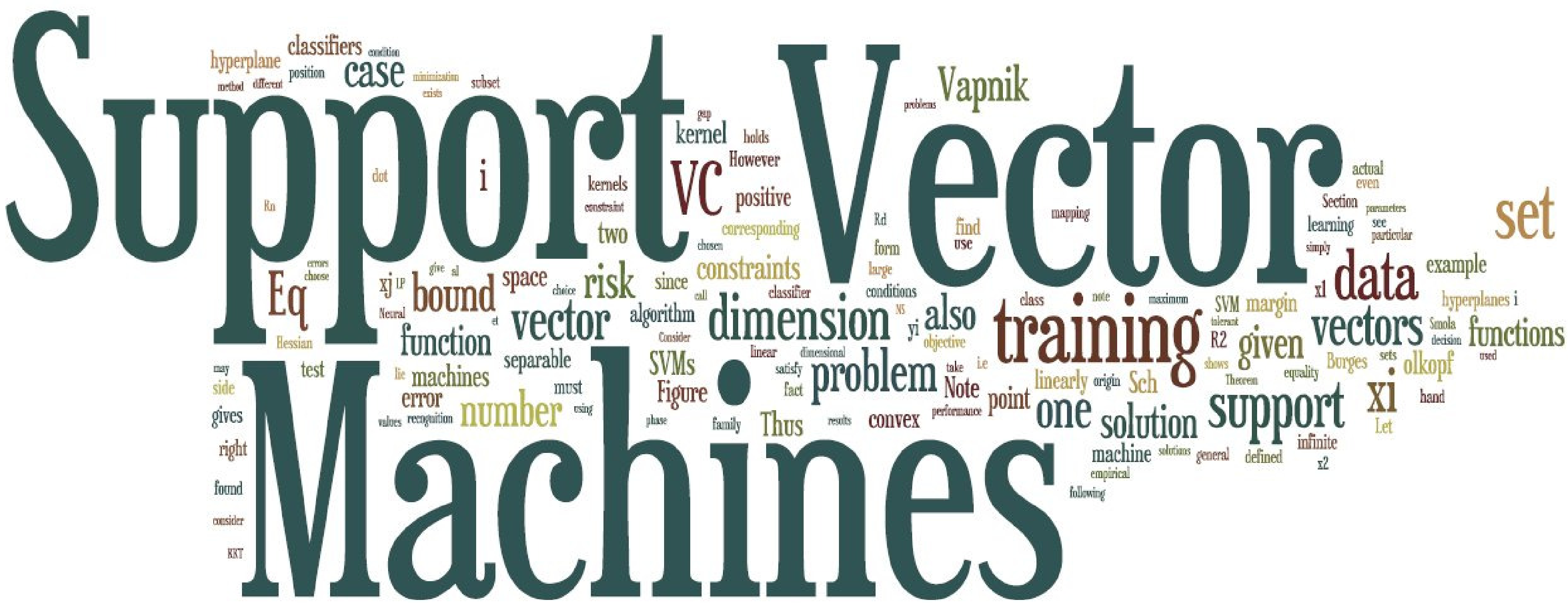


Gautam Kunapuli

University of Wisconsin-Madison

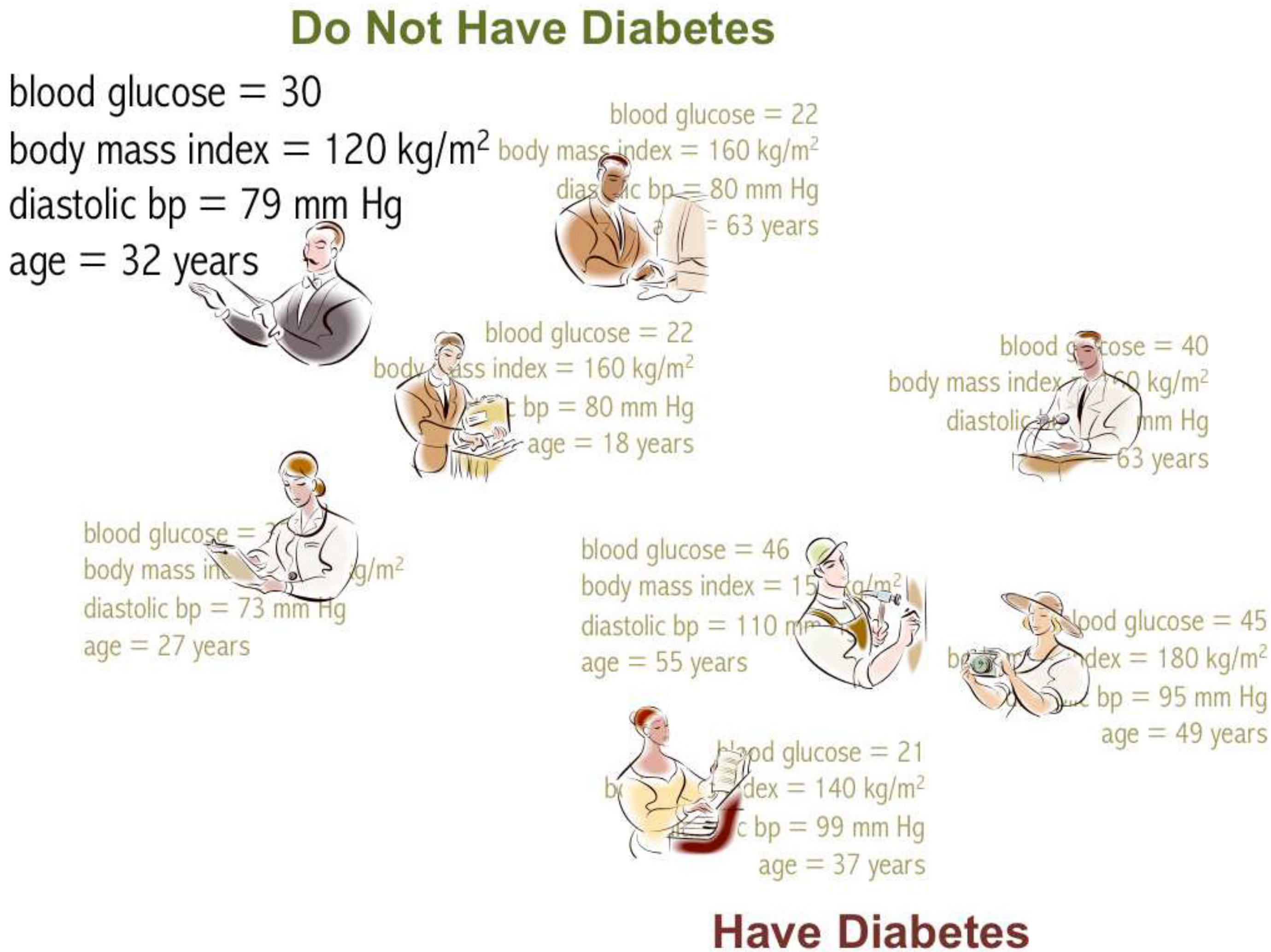
Outline

- **Example of classification problems**
- Formulating Support Vector Machines
- SVM Properties
- Soft-Margin SVMs
- Kernels and Nonlinear Data
- Regularization

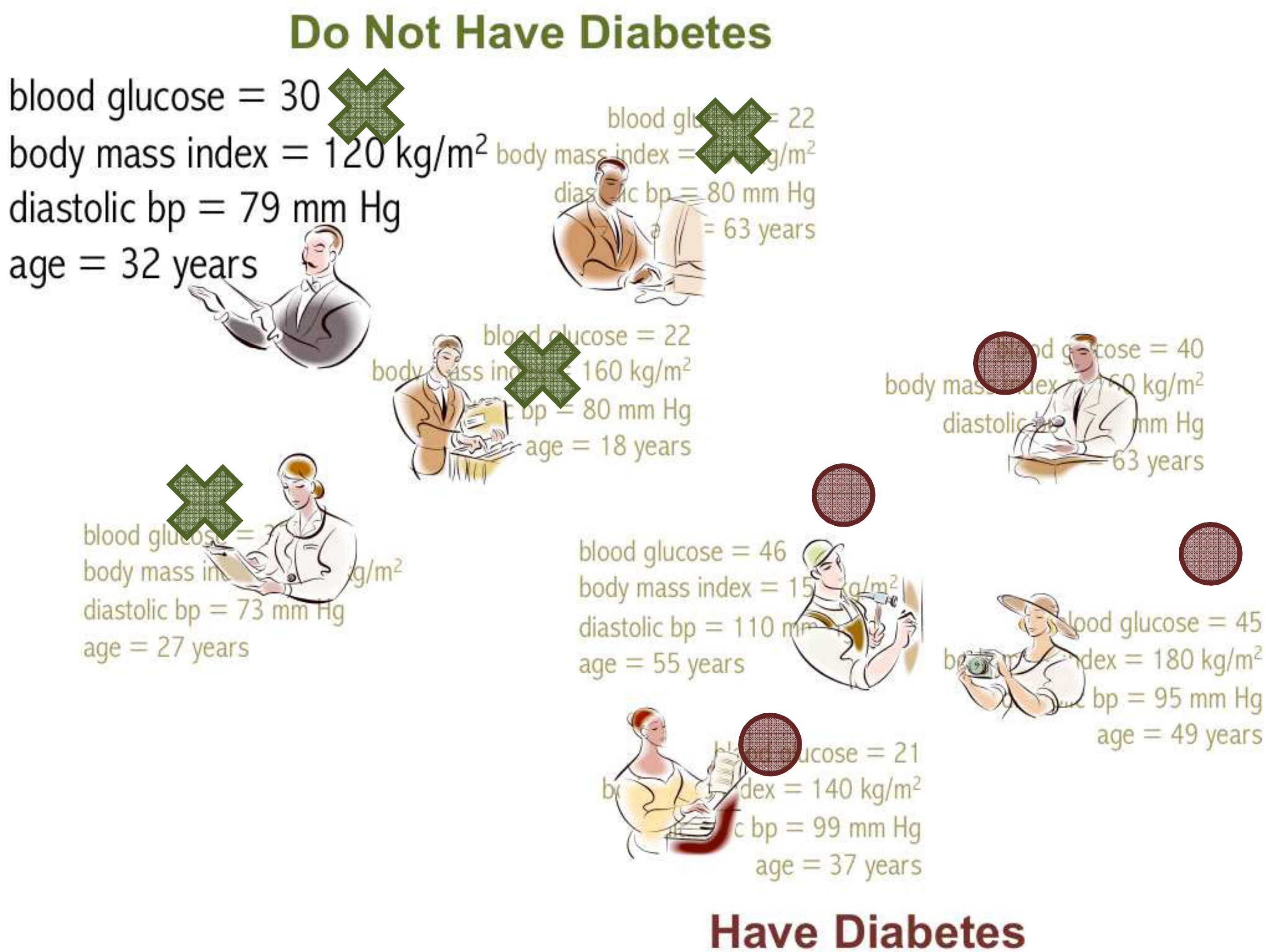
Example: Diabetes Diagnosis

Patient #	Plasma gluc.	Diastolic BP (mm Hg)	Fold test (mm)	2-hr Insulin	BMI (kg/m^2)	pedigree function	Age (yrs)	diabetes diagnosis
1	85	66	29	0	26.6	0.351	31	no
2	183	64	0	0	23.3	0.672	32	yes
3	89	66	23	94	28.1	0.167	21	no
4	137	40	35	168	43.1	2.288	33	yes
5	116	74	0	0	25.6	0.201	30	no
6	78	50	32	88	31	0.248	26	yes
7	197	70	45	543	30.5	0.158	53	yes
…	…	…	…	…	…	…	…	…
768	166	72	19	175	25.8	0.587	51	yes

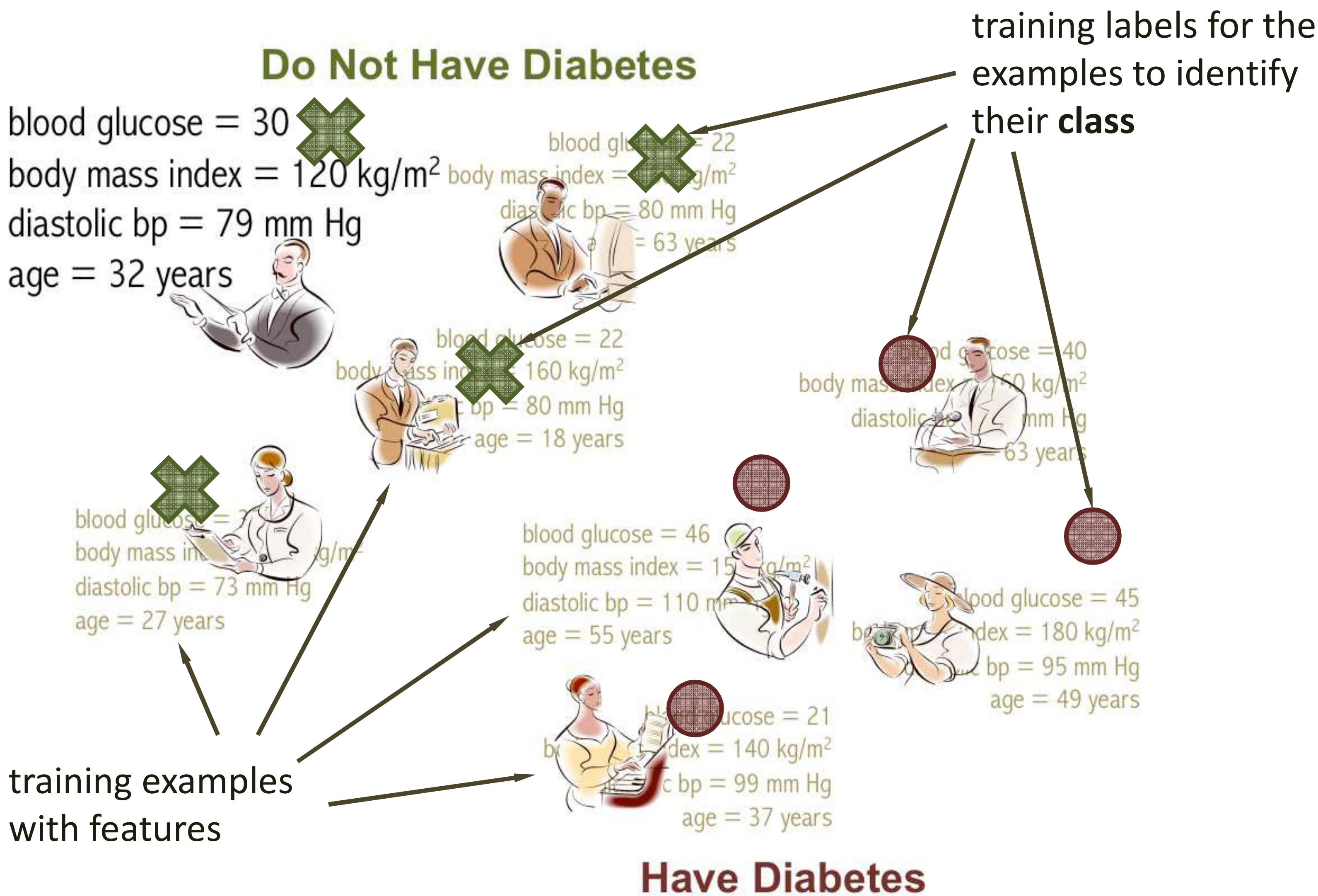
Example: Diabetes Diagnosis



Example: Diabetes Diagnosis

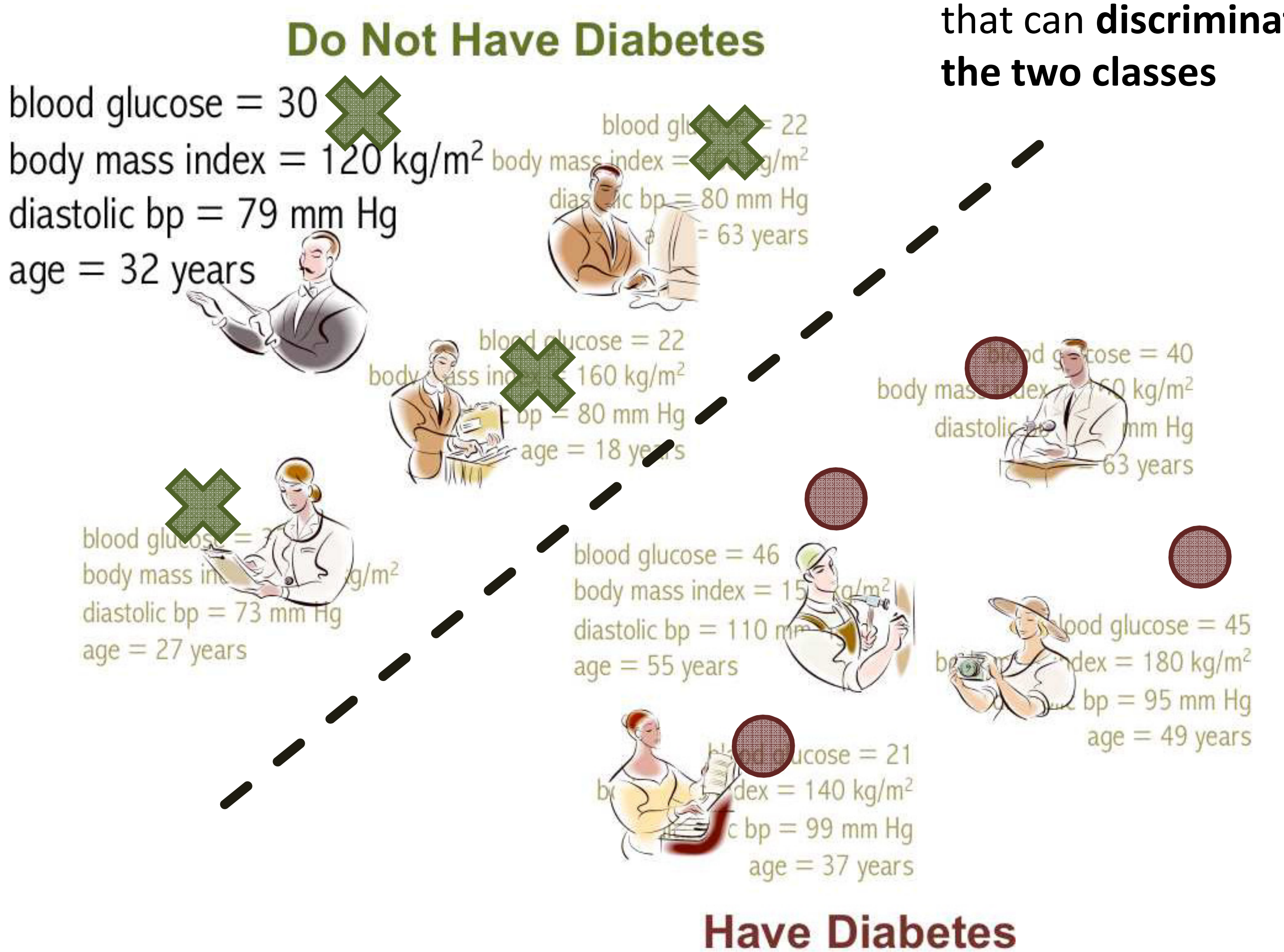


Example: Diabetes Diagnosis



Example: Diabetes Diagnosis

Learn a **classification function** that can **discriminate between the two classes**

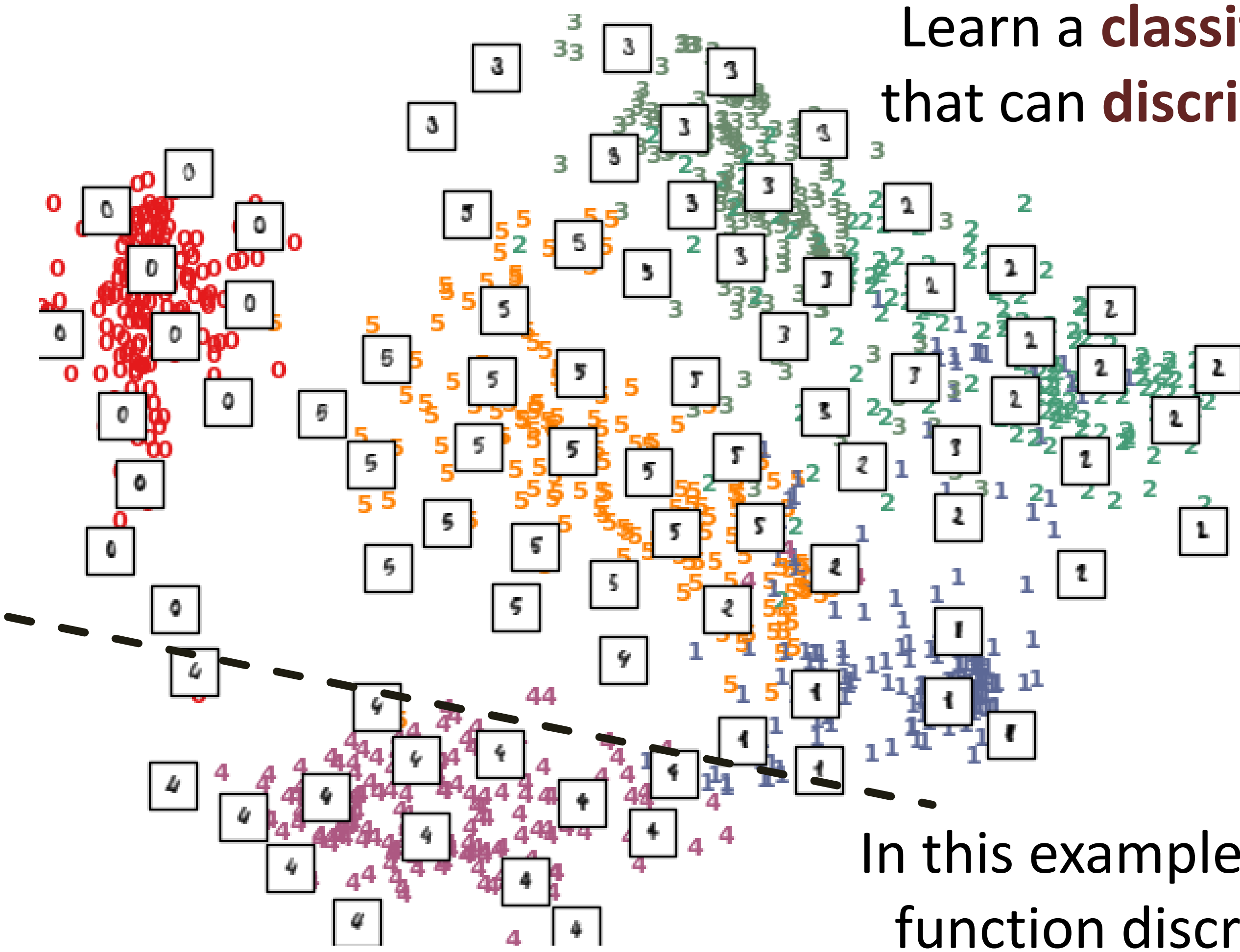


Example: Credit Card Application

		Age (yrs)	Income ($)	Rent/Own? (binary)	Monthly Rent/Mort.	Manager's Decision
1	Greg J.	38	65,000	rent	1,050	yes
2	James T.	24	21,000	rent	350	no
3	Hannah M.	45	98,000	own	2,400	no
4	Rashard K.	19	19,500	own	400	yes
5	Xavier N.	29	75,000	rent	1,570	no
6	Jillian A.	29	39,000	own	1,000	yes
7	Ramon H.	35	103,000	rent	3,000	yes
…	…	…	…	…	…	…
2000	Mary C.	55	45,000	rent	1,200	no

Learn a classification function to determine who gets a credit card.

Example: Handwritten Digit Recognition

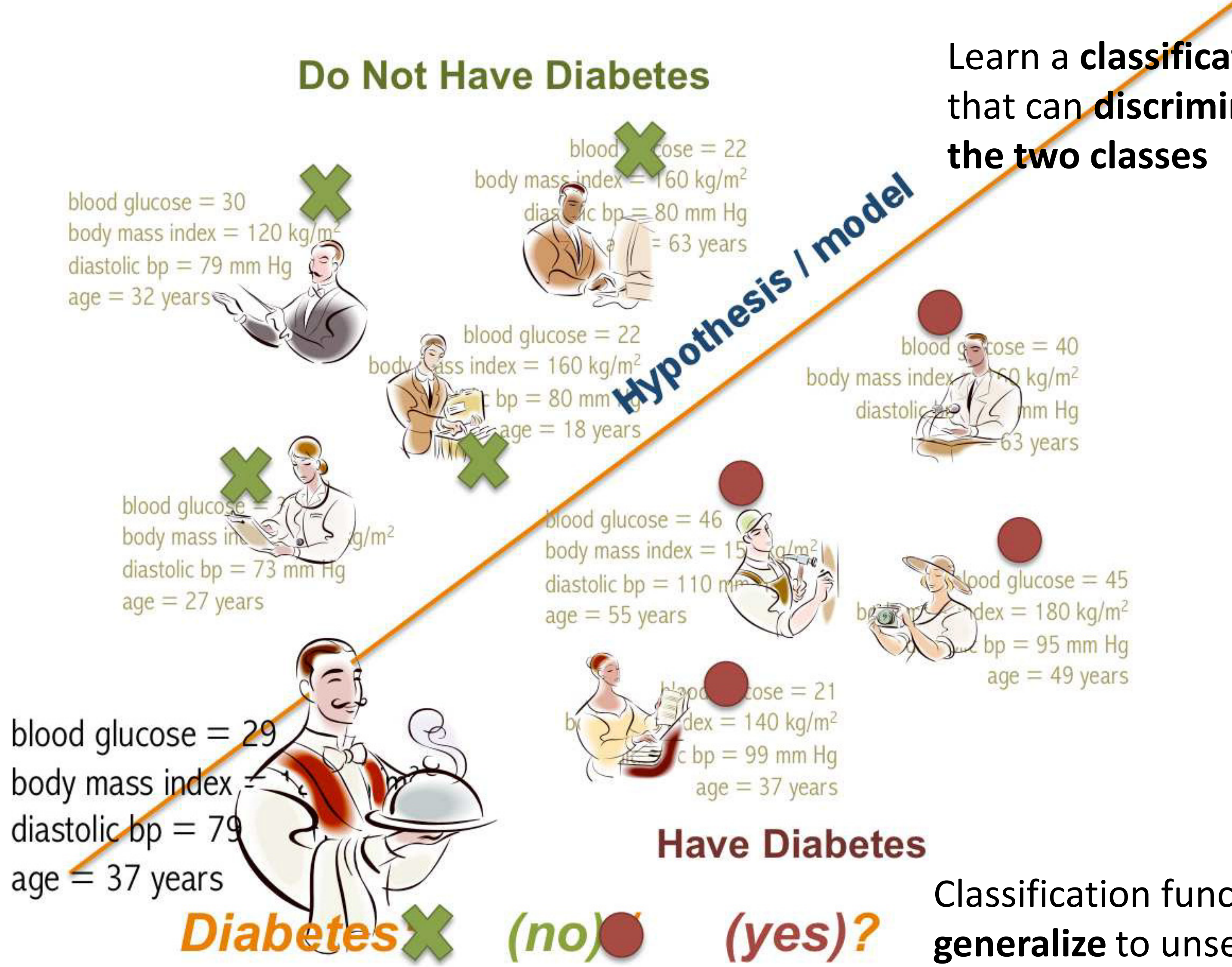


Learn a **classification function** that can **discriminate between multiple classes**

In this example, the classification function discriminates between **4**s and not **4**s. So, still an example of a 2-class problem

Goal of Classification: Generalization

Learn a **classification function** that can **discriminate between the two classes**

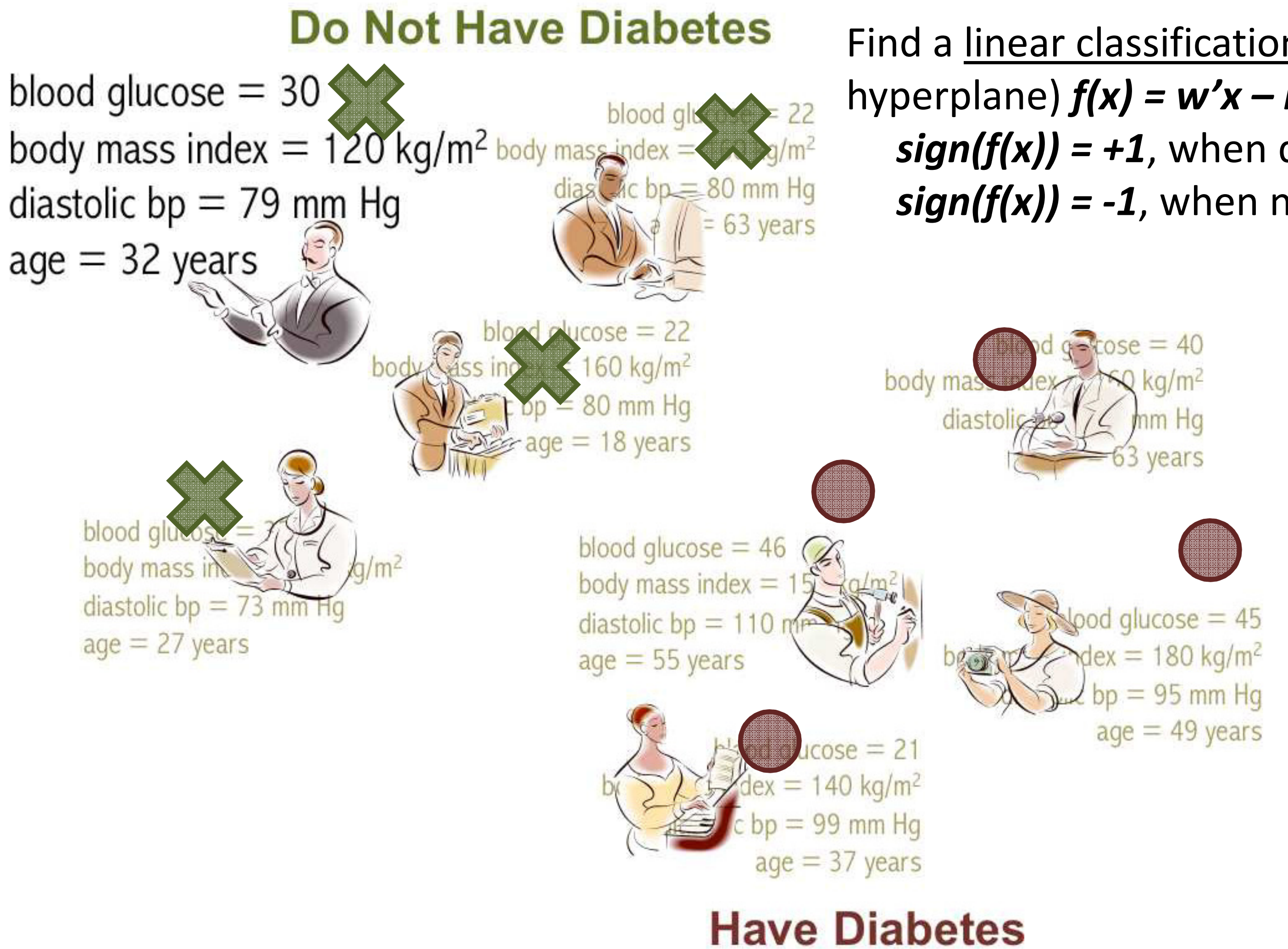


Classification function can also **generalize** to unseen examples, and classify them correctly

Outline

- Example of classification problems
- **Formulating Support Vector Machines**
- SVM Properties
- Soft-Margin SVMs
- Kernels and Nonlinear Data
- Regularization

Setting Up The SVM Problem

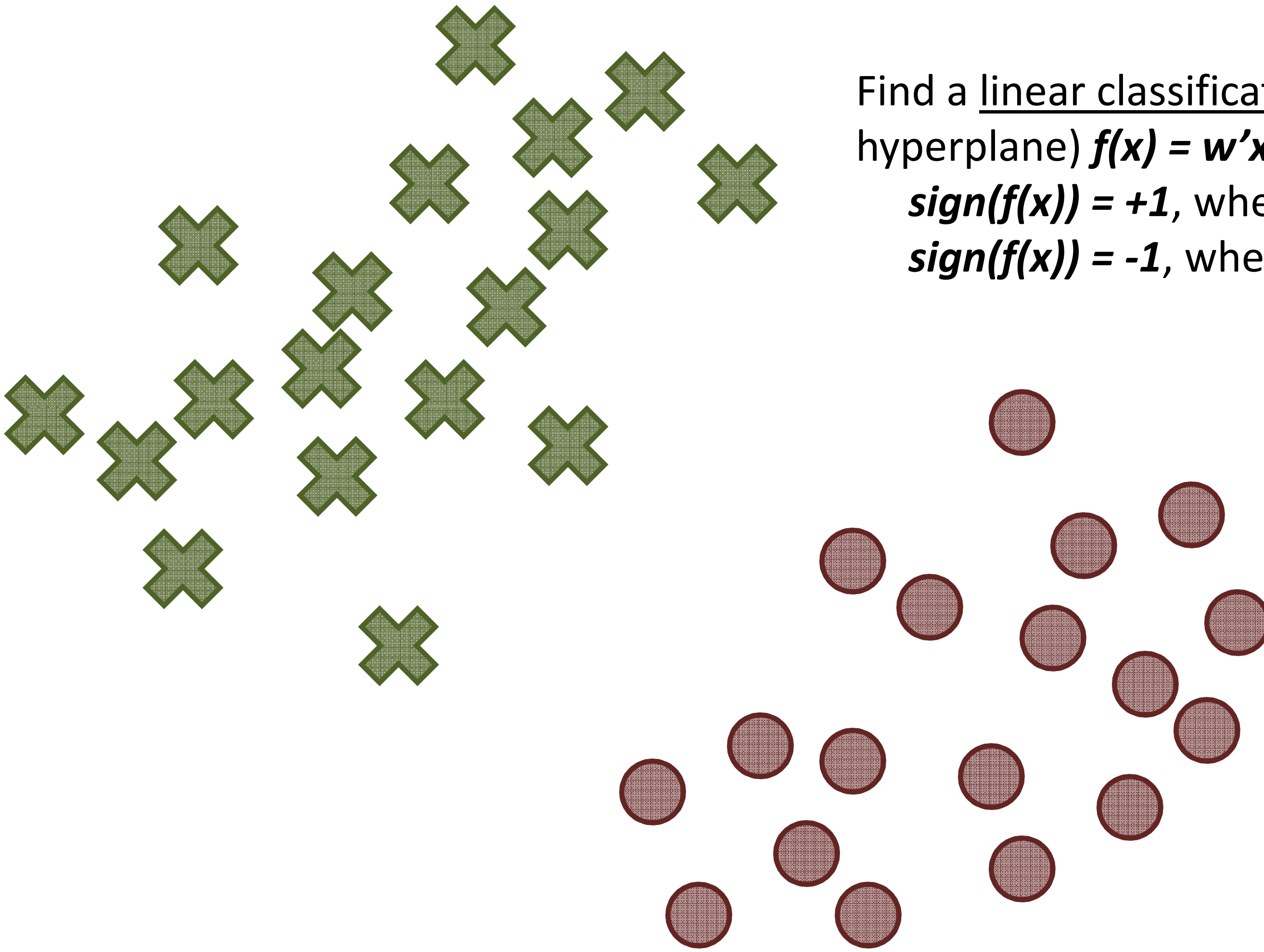


Find a <u>linear classification function</u> (a hyperplane) ***f(x) = w'x – b*** such that

sign(f(x)) = +1, when diabetes

sign(f(x)) = -1, when not diabetes

The SVM Problem



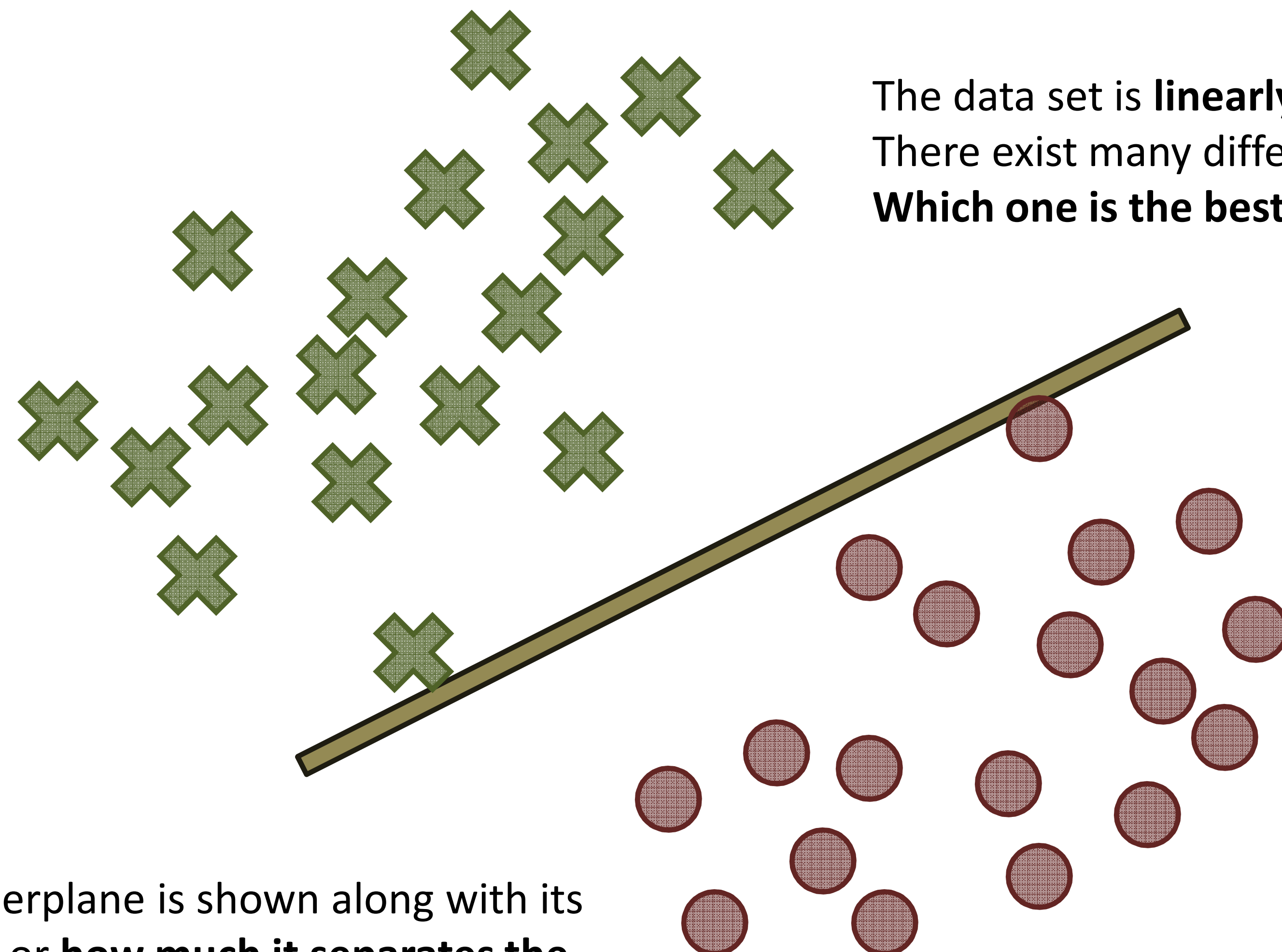
Find a <u>linear classification function</u> (a hyperplane) ***f(x) = w'x – b*** such that

sign(f(x)) = +1, when diabetes

sign(f(x)) = -1, when not diabetes

The Notion of Margin

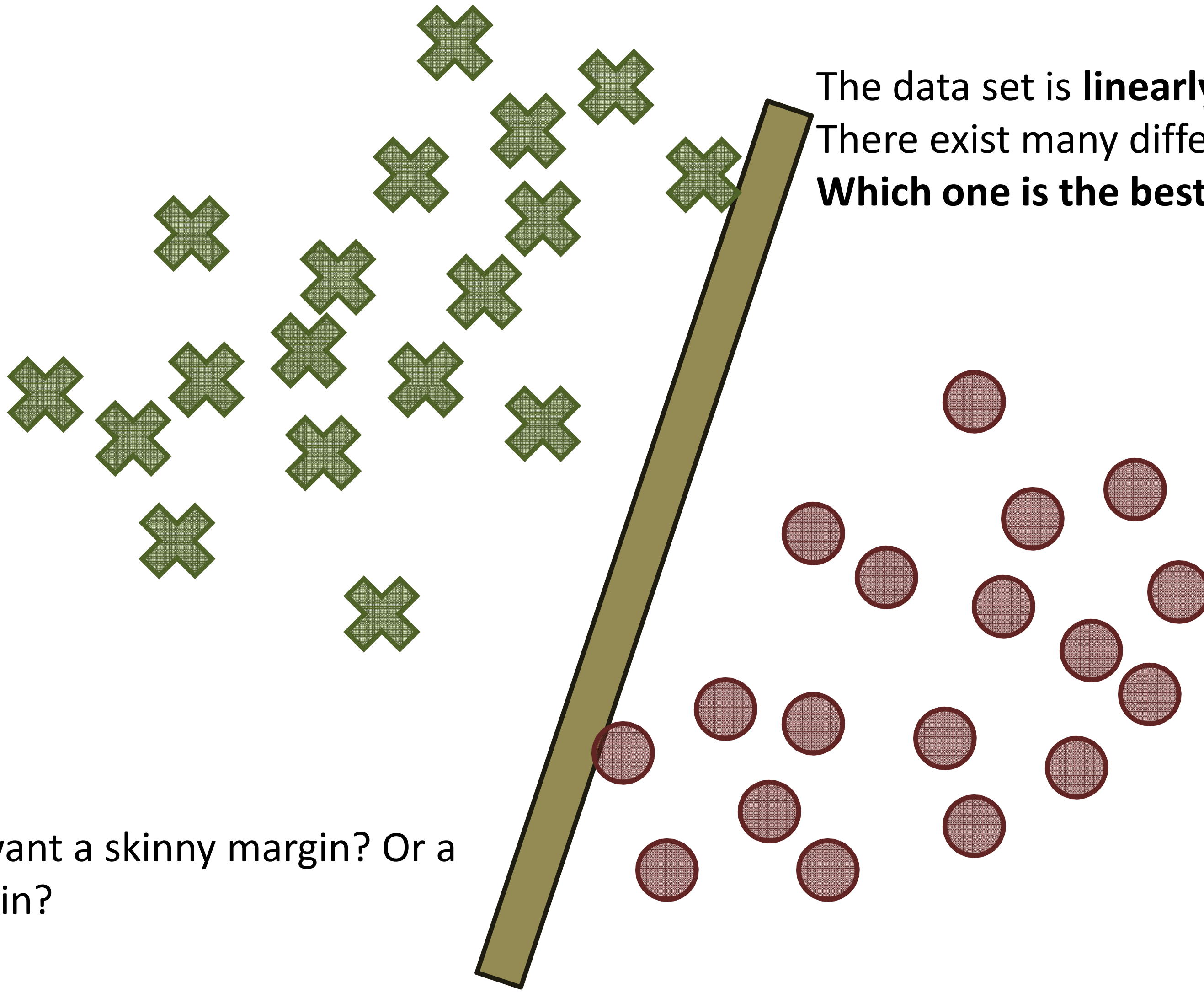
The data set is **linearly separable**.
There exist many different classifiers!
Which one is the best?



The hyperplane is shown along with its **margin**, or **how much it separates the two classes.**

The Notion of Margin

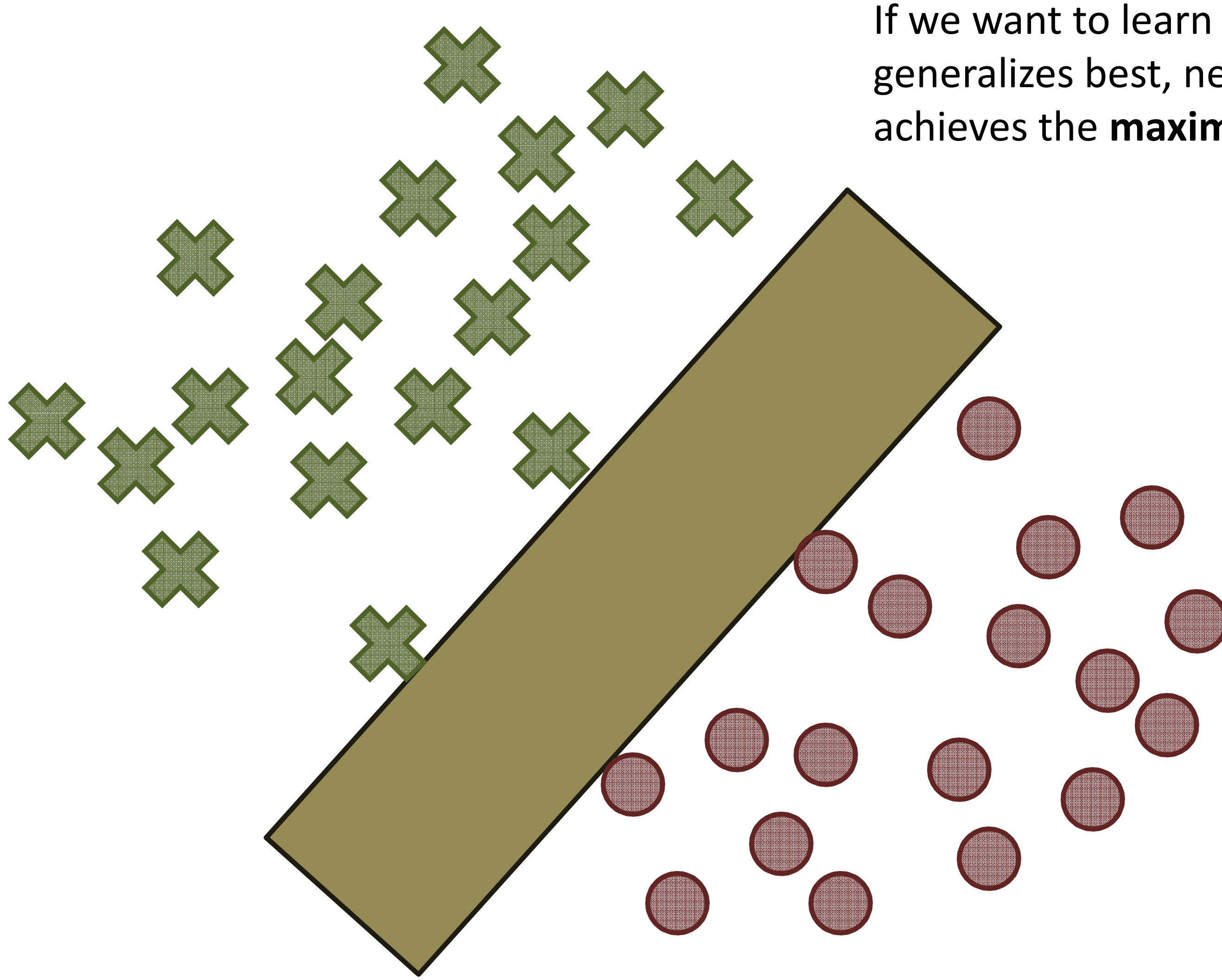
The data set is **linearly separable**.
There exist many different classifiers!
Which one is the best?



Do we want a skinny margin? Or a fat margin?

The Notion of Margin

If we want to learn a classifier that generalizes best, need one that achieves the **maximum margin**.

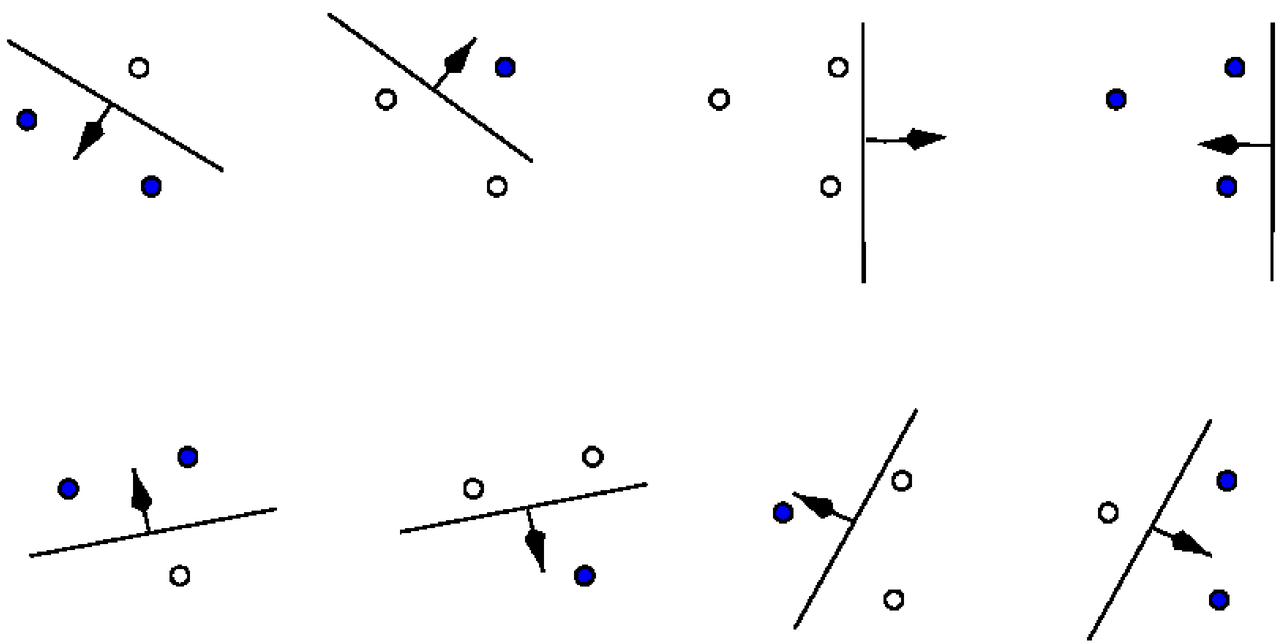


A Theoretical Digression

- When we have infinitely many solutions that reduce the error (**variance**), which one should we pick?
- We introduce a **bias**! Our bias is toward simpler solutions (minimal complexity).
- How do we measure complexity of hyperplanes with respect to data? **Vapnik-Chervonenkis dimension**

A Theoretical Digression: VC Dimension

- Complexity for a **class of functions *H*** is measured by **VC Dimension**
- A given set of $\boldsymbol{\ell}$ points can be labeled in $\boldsymbol{2^{\ell}}$ ways.
- If for each labeling, some function ***f*** from ***H*** can be found which correctly assigns those labels, we say that that set of points is ***shattered by H***.
- The VC dimension for the set of functions H is defined as the **maximum number of training points that can be shattered**



A Theoretical Digression: VC Dimension

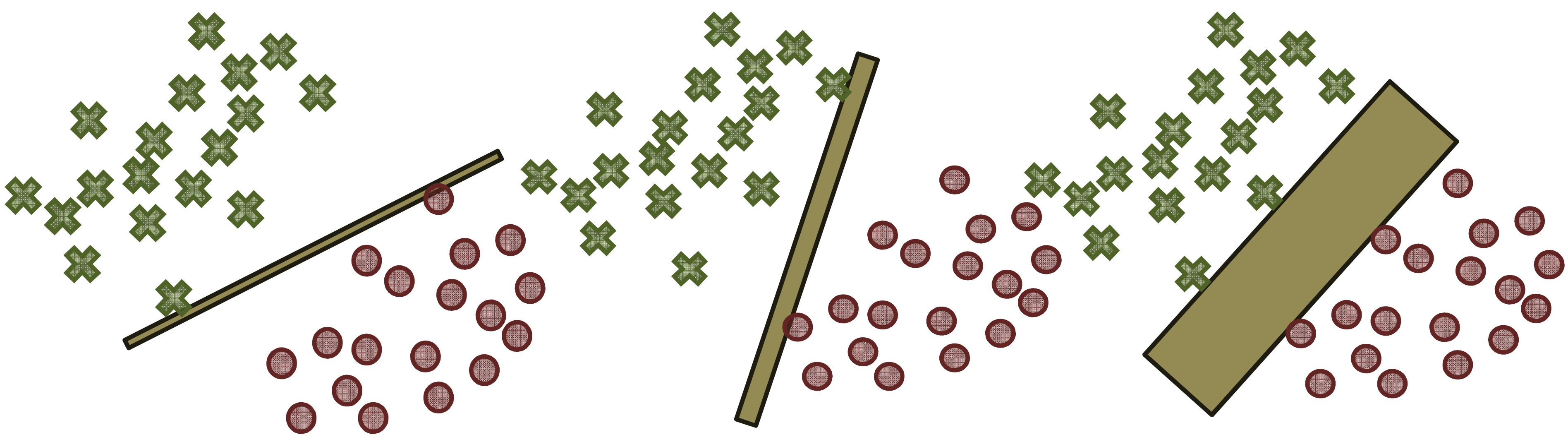
Vladimir Vapnik showed that there is a **connection between VC dimension and margin** (Vapnik, 1995)

$$h \leq \frac{4R^2}{\gamma^2}$$

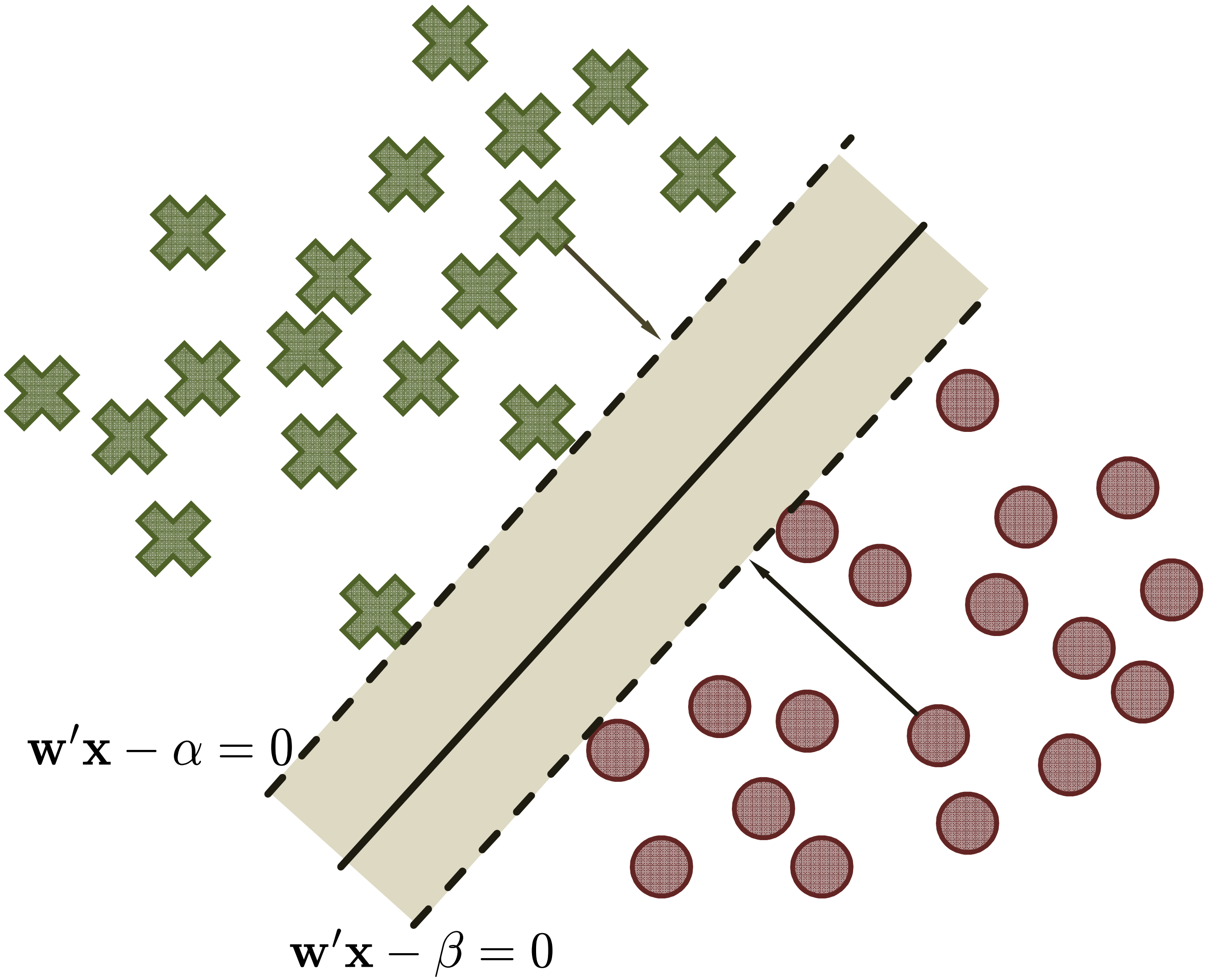
VC dimension

margin

To **minimize the VC dimension** (and hence complexity), we have to **maximize the margin**!



Formulating the SVM



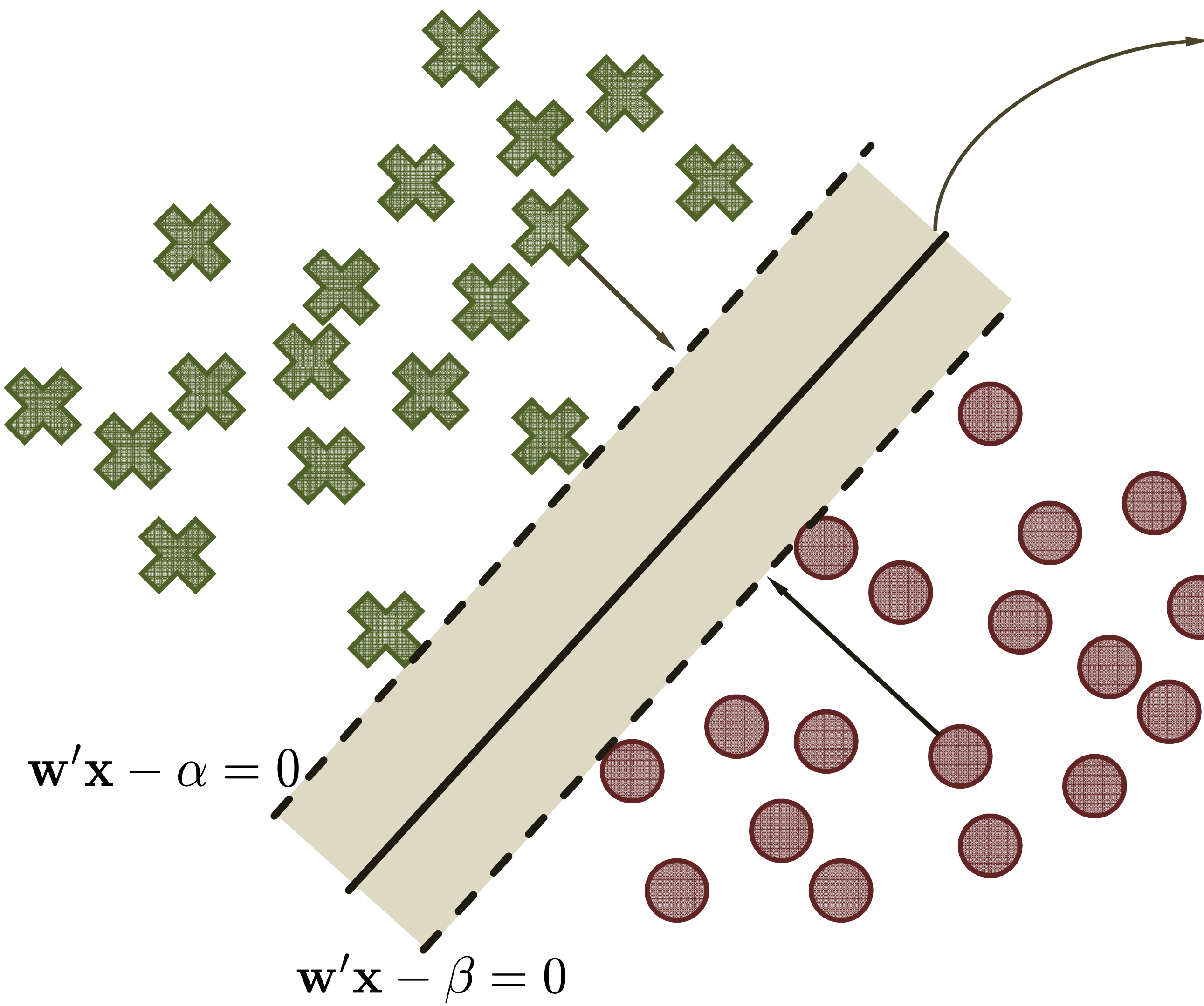
Distance of a point in Class to the margin is

$$\frac{|\mathbf{w}'\mathbf{x} - \alpha|}{\|\mathbf{w}\|}$$

Distance of a point in Class to the margin is

$$\frac{|\mathbf{w}'\mathbf{x} - \beta|}{\|\mathbf{w}\|}$$

Formulating the SVM



Distance between the hyperplanes is the **margin**

$$\gamma = \frac{|\alpha - \beta|}{\|\mathbf{w}\|}$$

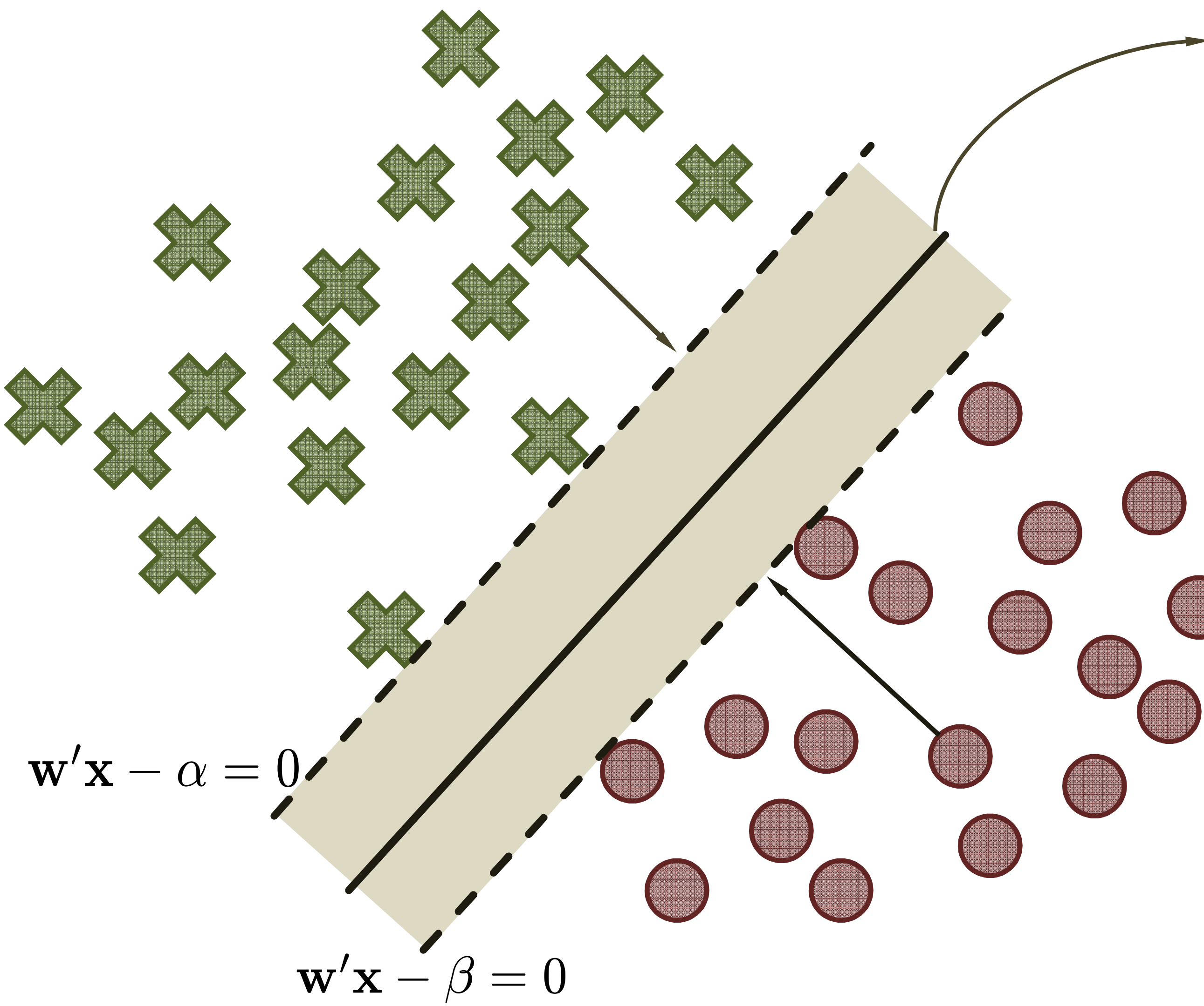
Distance of a point in Class to the margin is

$$\frac{|\mathbf{w}'\mathbf{x} - \alpha|}{\|\mathbf{w}\|}$$

Distance of a point in Class to the margin is

$$\frac{|\mathbf{w}'\mathbf{x} - \beta|}{\|\mathbf{w}\|}$$

Formulating the SVM



Distance between the hyperplanes is the **margin**

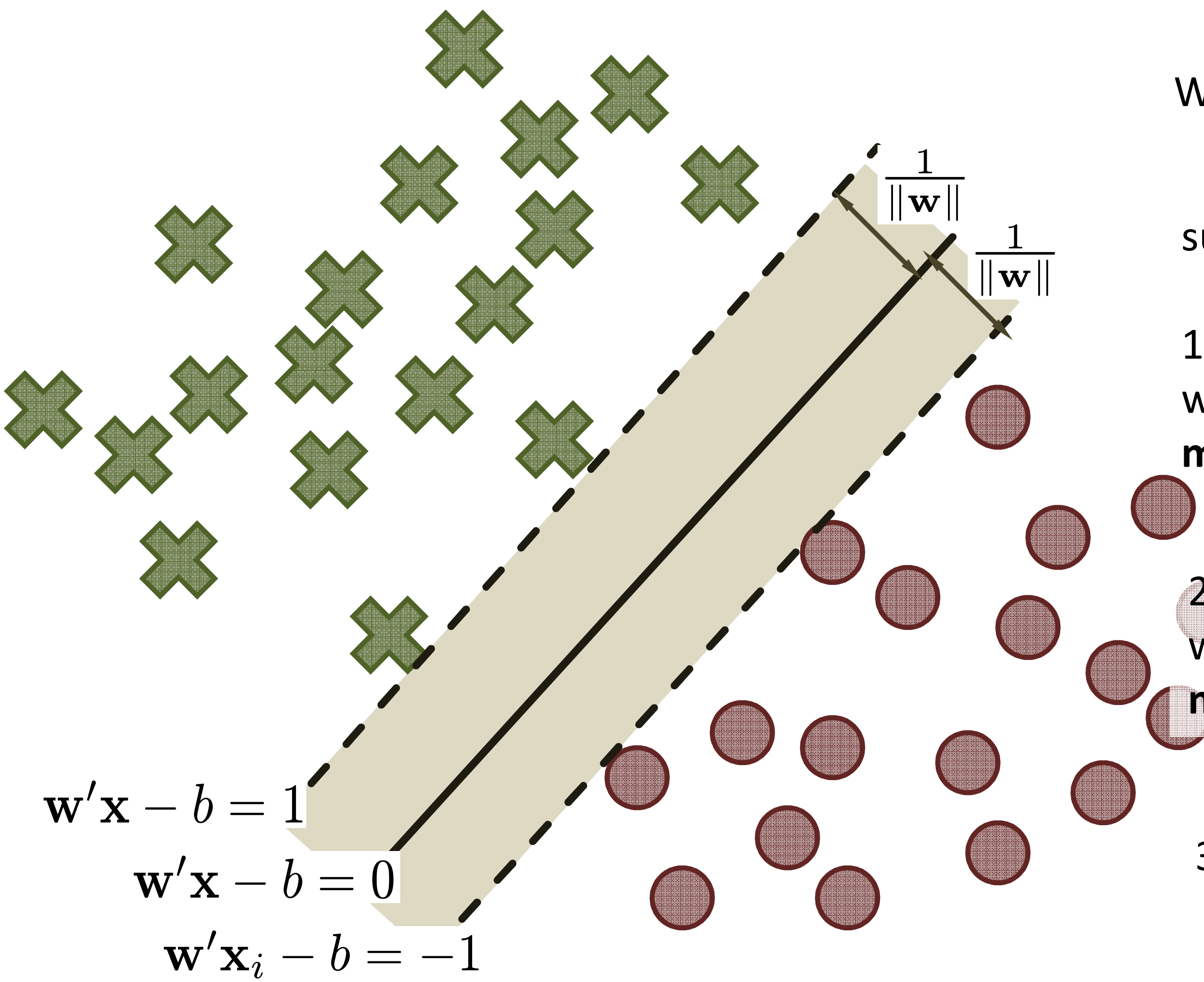
$$\gamma = \frac{|\alpha - \beta|}{\|\mathbf{w}\|}$$

Set $\alpha = b + 1$ and $\beta = b - 1$,, then the **margin** is

$$\gamma = \frac{2}{\|\mathbf{w}\|}$$

which should be maximized.

Formulating the SVM



We are given **labeled data points**

$$(\mathbf{x}_i,\ y_i)_{i=1}^{\ell}$$

We need to learn a hyperplane

$$\mathbf{w}'\mathbf{x} - b = 0$$

such that

1. all the points in class with labels $\mathbf{y_i}$ **= +1**, lie **above the margin**, that is

$$\mathbf{w}'\mathbf{x}_i - b \geq 1$$

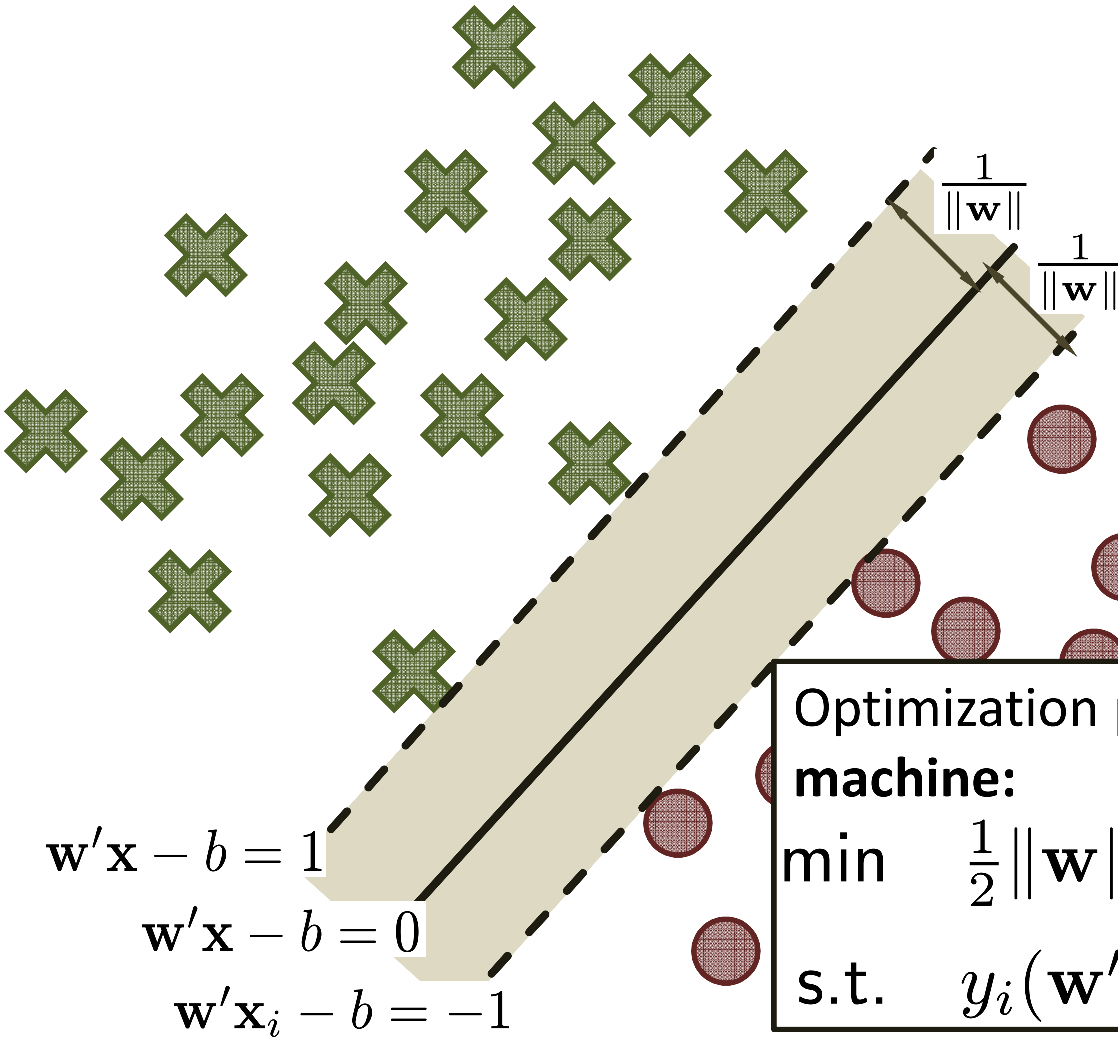
2. all the points in class with labels $\mathbf{y_i}$ **= -1**, lie **below the margin**, that is

$$\mathbf{w}'\mathbf{x}_i - b \leq -1$$

3. the margin is maximized

$$\gamma = \frac{2}{\|\mathbf{w}\|}$$

Formulating the SVM



Maximizing $\gamma = \frac{2}{\|\mathbf{w}\|}$ is **equivalent to minimizing**

$$\frac{1}{2}\|\mathbf{w}\|^2$$

with the constraints that

$\mathbf{w}'\mathbf{x}_i - b \geq 1$ when $y_i = +1$

$\mathbf{w}'\mathbf{x}_i - b \leq -1$ when $y_i = -1$

Optimization problem for a **support vector machine:**

min $\quad \frac{1}{2}\|\mathbf{w}\|_2^2$

s.t. $\quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1 \ldots \ell$

Optimization problem for a **support vector machine:**

$$\min \quad \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1 \ldots \ell$$

- **Convex, quadratic** minimization problem called the **primal problem.** Guaranteed to have a **global minimum**.
- Further properties of the formulation can be studied by deriving the **dual problem**
- Introduce **Lagrange multipliers,** $\{\alpha_i\}_{i=1}^{\ell}$, one for each constraint (hence data point). These are the **dual variables**.
- Construct the Lagrangian function of primal and dual variables (note that by definition all $\alpha_i \geq 0$)

$$L(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\mathbf{w}'\mathbf{w} - \sum_{i=1}^{\ell} \alpha_i \left[y_i(\mathbf{w}'\mathbf{x}_i - b) - 1\right]$$

Lagrangian function of a support vector machine

$$L(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\mathbf{w}'\mathbf{w} - \sum_{i=1}^{\ell} \alpha_i \left[y_i(\mathbf{w}'\mathbf{x}_i - b) - 1\right]$$

Differentiate the Lagrangian **with respect to the primal variables**

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \alpha_i) = 0 : \quad \mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$

$$\nabla_{b} L(\mathbf{w}, b, \alpha_i) = 0 : \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0$$

These are the **first order optimality conditions**. We can now **eliminate** the primal variables by **substituting the first order conditions** into the Lagrangian.

support vector machine **primal problem**

$$\min \quad \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1 \ldots \ell$$

support vector machine **dual problem**

$$\max \quad -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{\ell}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \quad \forall i = 1 \ldots \ell$$

Why bother with the dual, and solving for α?

- convex optimization problem. ***No duality gap***
- Dual has fewer constraints. ***Easier to solve***
- Dual solution is sparse. ***Easier to represent***

Outline

- Example of classification problems
- Formulating Support Vector Machines
- **SVM Properties**
- Soft-Margin SVMs
- Kernels and Nonlinear Data
- Regularization

Characteristics of the Solution

Recall the first order condition:

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$

The final solution is a **linear combination of the training data**!

Additional optimality condition: **complementarity slackness**

$$0 \leq \alpha_i \quad \perp \quad y_i(\mathbf{w}'\mathbf{x} - b) - 1 \geq 0$$

$\alpha_i = 0$ and $y_i(\mathbf{w}'\mathbf{x}_i - b) > 1$ (point not on hyperplane)

or

$\alpha_i > 0$ and $y_i(\mathbf{w}'\mathbf{x}_i - b) = 1$ (point on hyperplane)

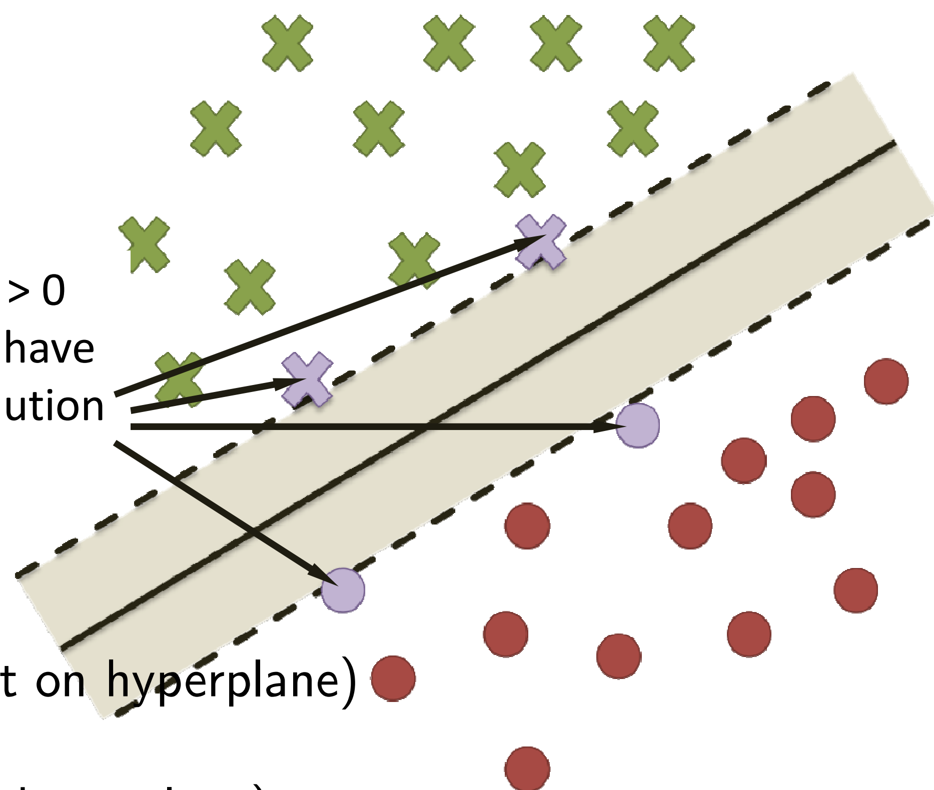
Characteristics of the Solution

Recall the first order condition:

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$

The final solution is a **sparse linear combination of the training data**!

only **support vectors** have $\alpha_i > 0$ (non-zero). All other vectors have $\alpha_i = 0$, and this makes the solution **sparse!**



$\alpha_i = 0$ and $y_i(\mathbf{w}'\mathbf{x}_i - b) > 1$ (point not on hyperplane)

or

$\alpha_i > 0$ and $y_i(\mathbf{w}'\mathbf{x}_i - b) = 1$ (point on hyperplane)

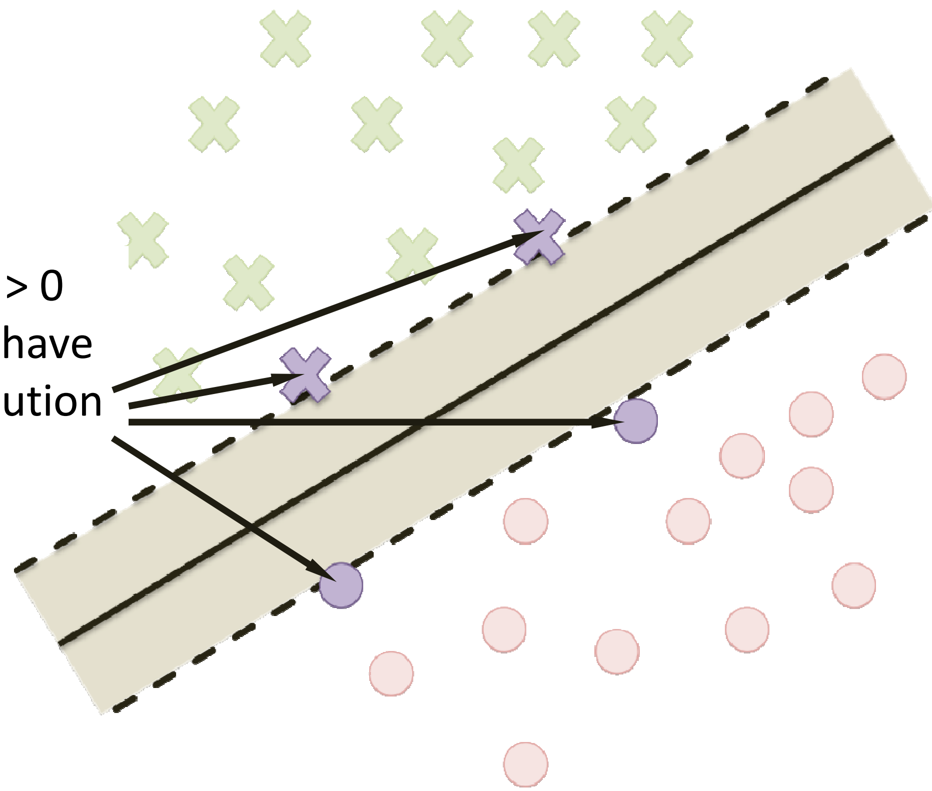
Characteristics of the Solution

Recall the first order condition:

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{x}_i$$

The final solution is a **sparse linear combination of the training data**!

only **support vectors** have $\alpha_i > 0$ (non-zero). All other vectors have $\alpha_i = 0$, and this makes the solution **sparse!**

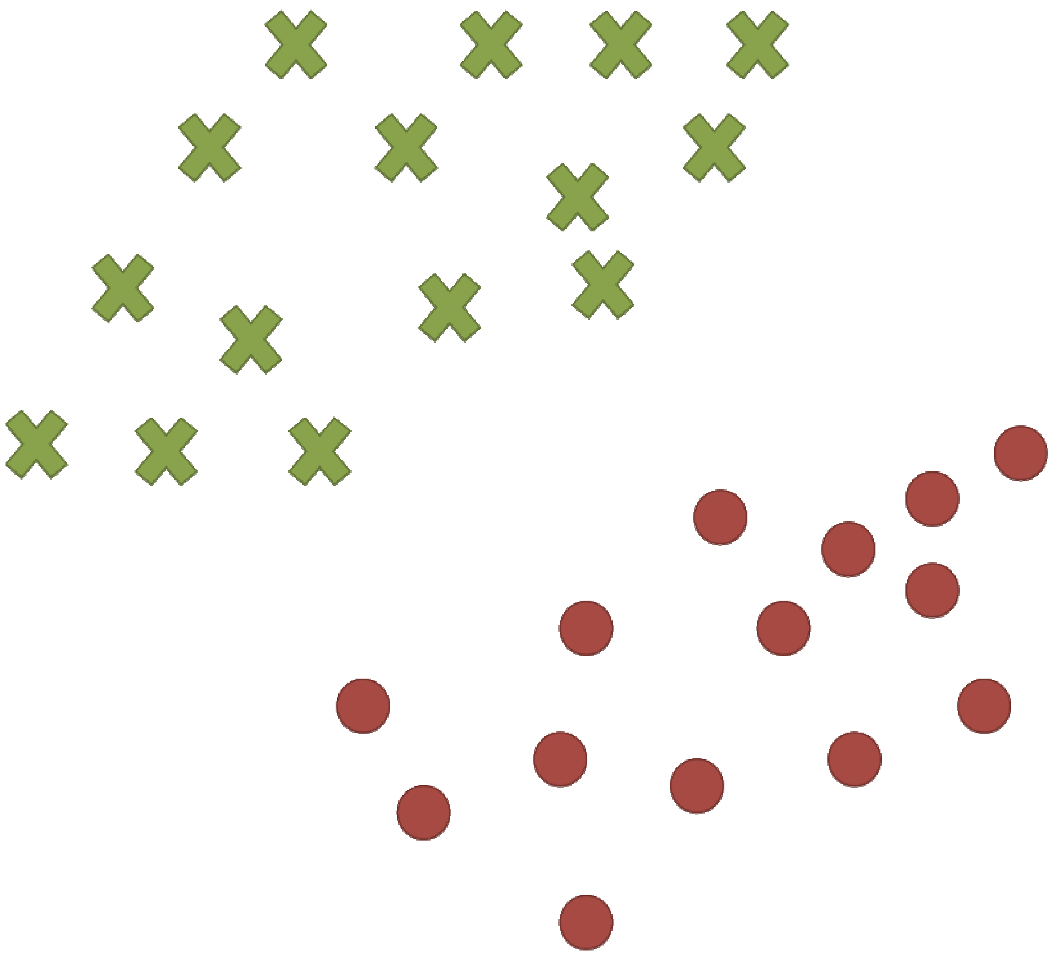


solution **does not change if** all other vectors with $\alpha_i = 0$ **are deleted!**

Outline

- Example of classification problems
- Formulating Support Vector Machines
- SVM Properties
- **Soft-Margin SVMs**
- Kernels and Nonlinear Data
- Regularization

Linearly Inseparable Data

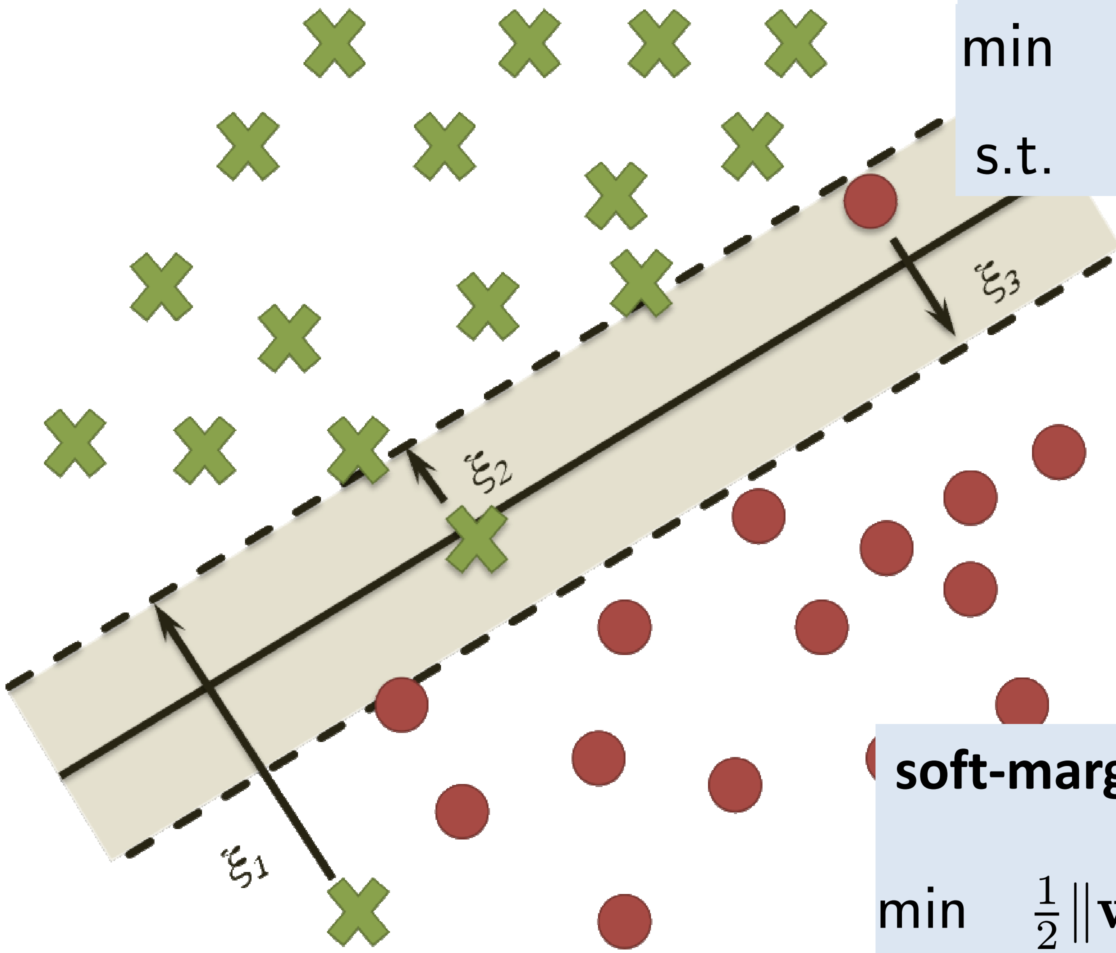


So far, assumed that the data is **linearly separable**. However, this assumption is not valid in most **real-world applications**.

Need to extend **hard-margin support vector machines** to be able to handle noisy data

This results in the **soft-margin support vector machine**

Soft-margin Support Vector Machine



hard-margin support vector machine

min $\quad \frac{1}{2}\|\mathbf{w}\|_2^2$

s.t. $\quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1 \ldots \ell$

soft-margin support vector machine

min $\quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i$

s.t. $\quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1 \ldots \ell$

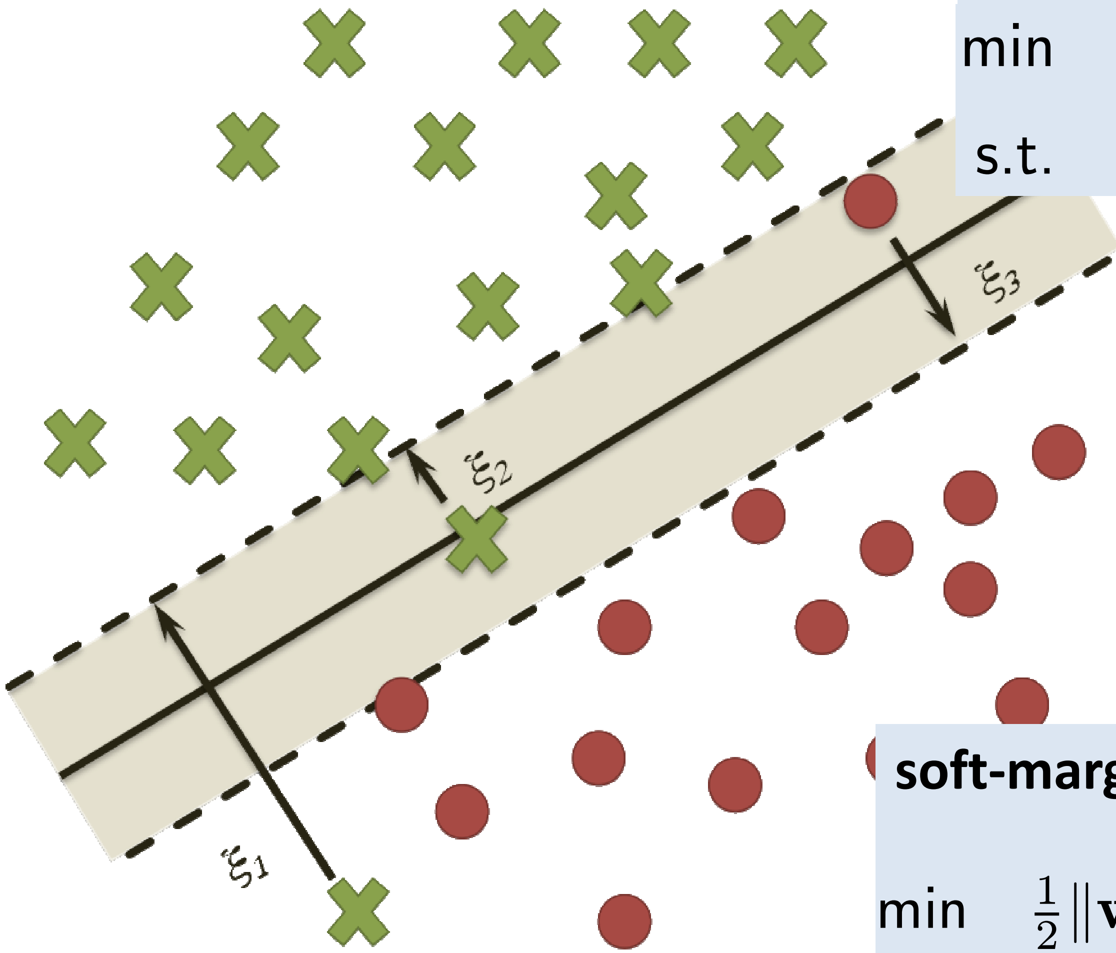
$\xi_i \geq 0$

Soft-margin Support Vector Machine

hard-margin support vector machine

min $\quad \frac{1}{2}\|\mathbf{w}\|_2^2$

s.t. $\quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1 \ldots \ell$



soft-margin support vector machine

min $\quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i$

s.t. $\quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1 \ldots \ell$

$\xi_i \geq 0$

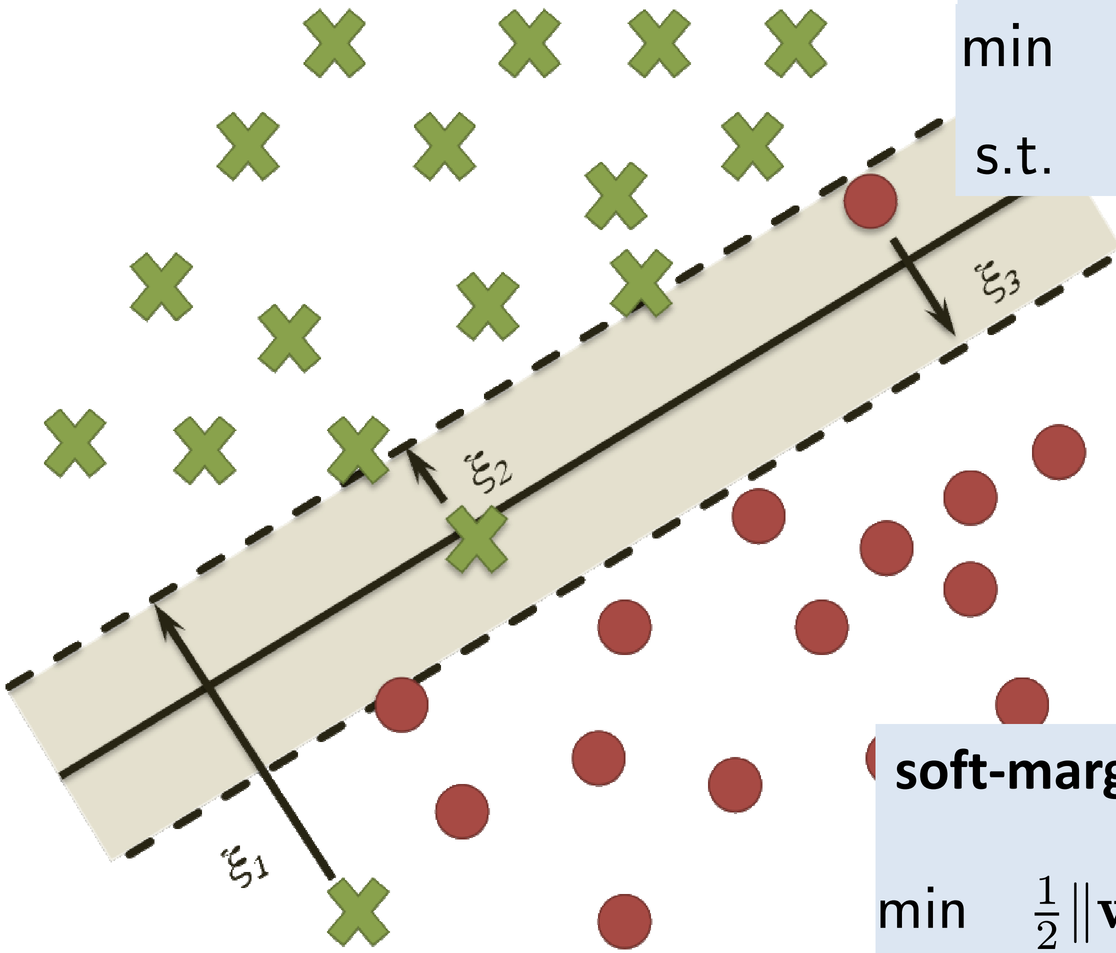
errors (slack variables) to measure **loss** of misclassified data points

Soft-margin Support Vector Machine

hard-margin support vector machine

$$\min \quad \tfrac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 \quad \forall i = 1 \ldots \ell$$



regularization constant that trades off between complexity and loss

soft-margin support vector machine

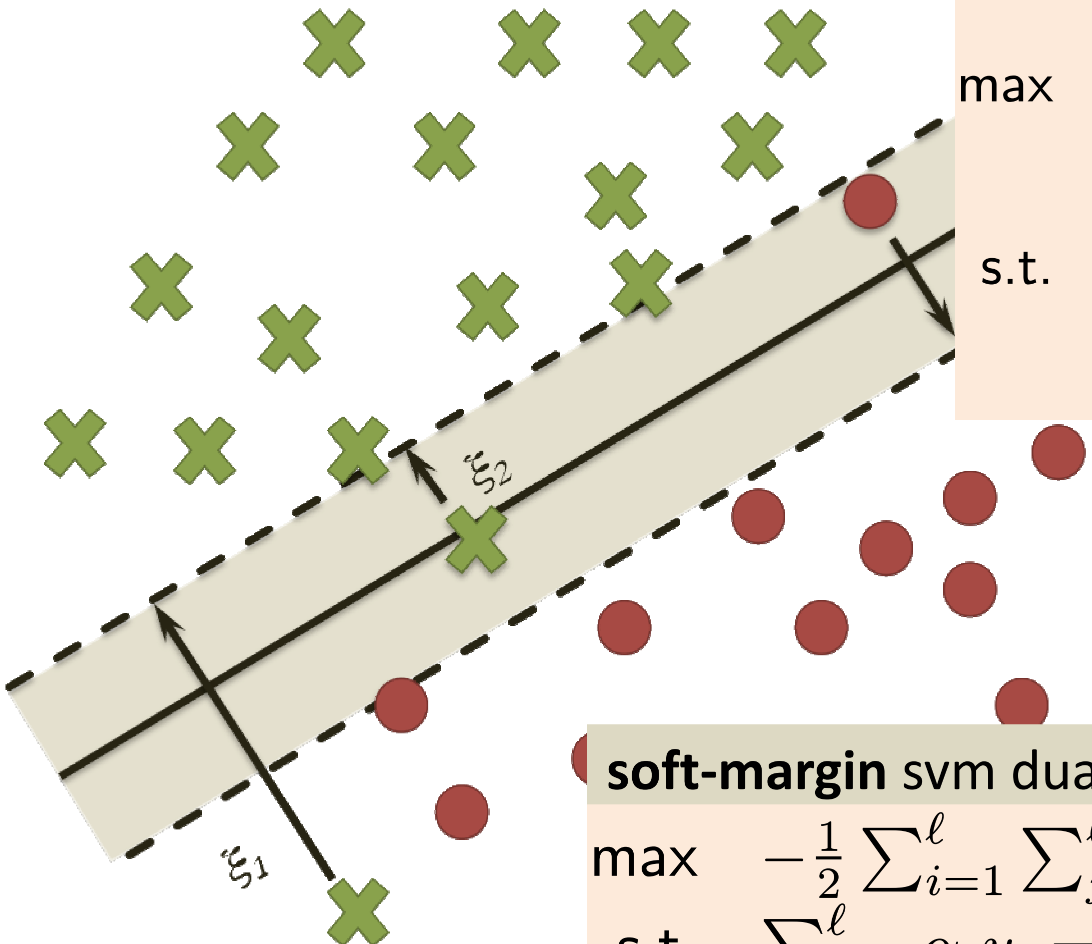
$$\min \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1 \ldots \ell$$

$$\xi_i \geq 0$$

errors (slack variables) to measure **loss** of misclassified data points

Soft-margin Dual



hard-margin svm dual

$$\max \quad -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{\ell}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0,$$

$$\alpha_i \geq 0, \quad \forall i = 1 \ldots \ell$$

soft-margin svm dual

$$\max \quad -\tfrac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{\ell}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \quad \forall i = 1 \ldots \ell$$

Some things to note about the Dual

$$\begin{array}{ll} \text{max} & -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{\ell}\alpha_i \\ \text{s.t.} & \sum_{i=1}^{\ell}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \ \ \forall i = 1\ldots\ell \end{array}$$

Note that the **regularization constant** is set by the user.

This is an important parameter that can cause **dramatically different behaviors on the same data set.**

Note that the dual solution depends **only on the inner products** of the training data.

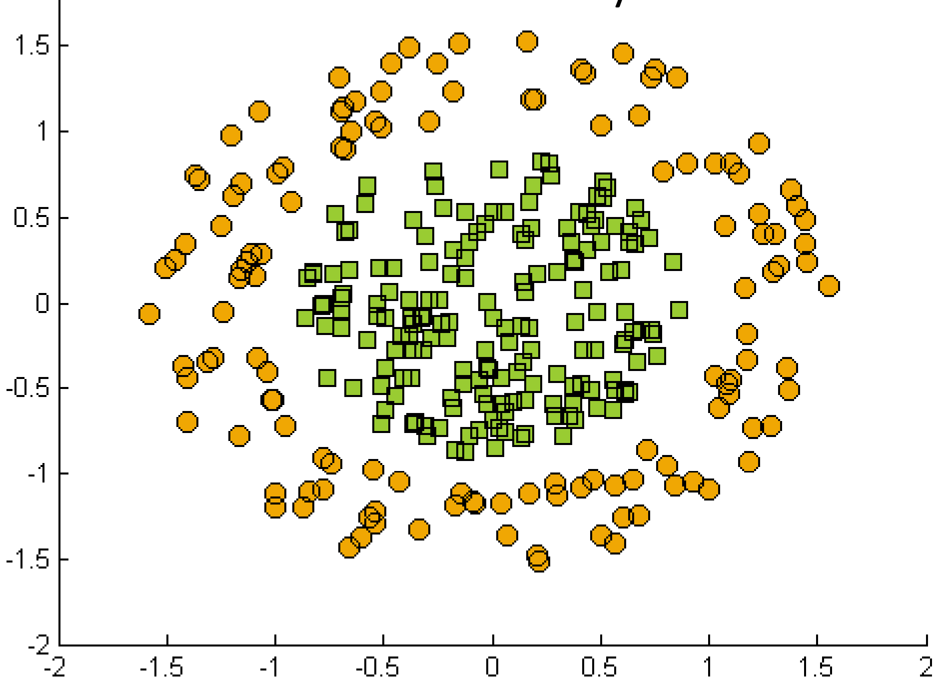
This is an important observation that allows us to extend linear SVMs to handle **nonlinear data.**

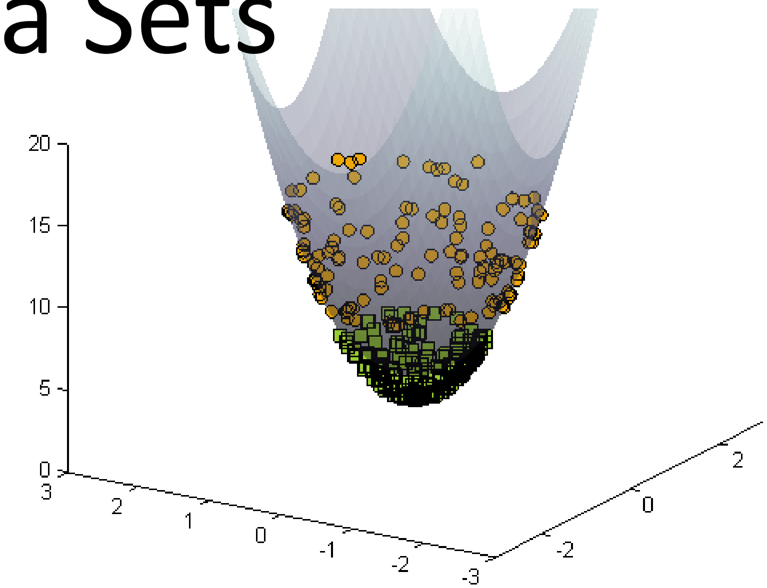
Outline

- Example of classification problems
- Formulating Support Vector Machines
- SVM Properties
- Soft-Margin SVMs
- **Kernels and Nonlinear Data**
- Regularization

Nonlinear Data Sets

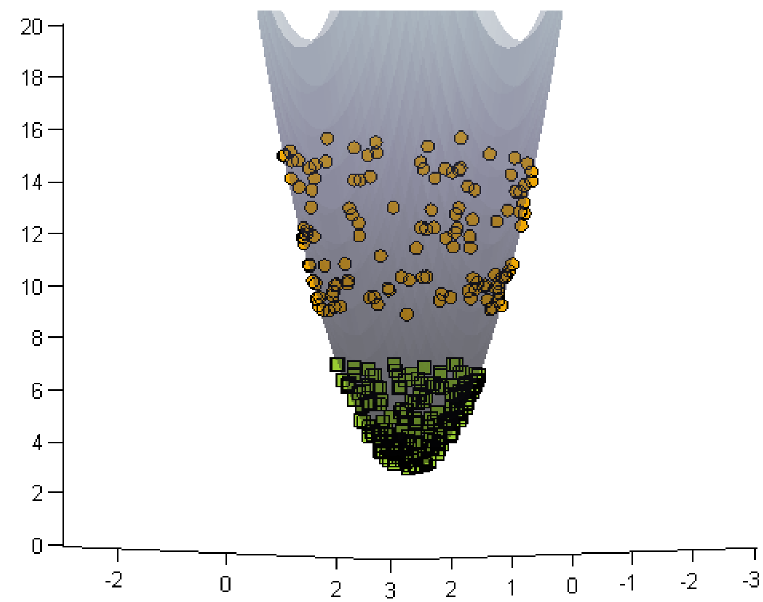
No linear classifier can classify this nonlinear data.



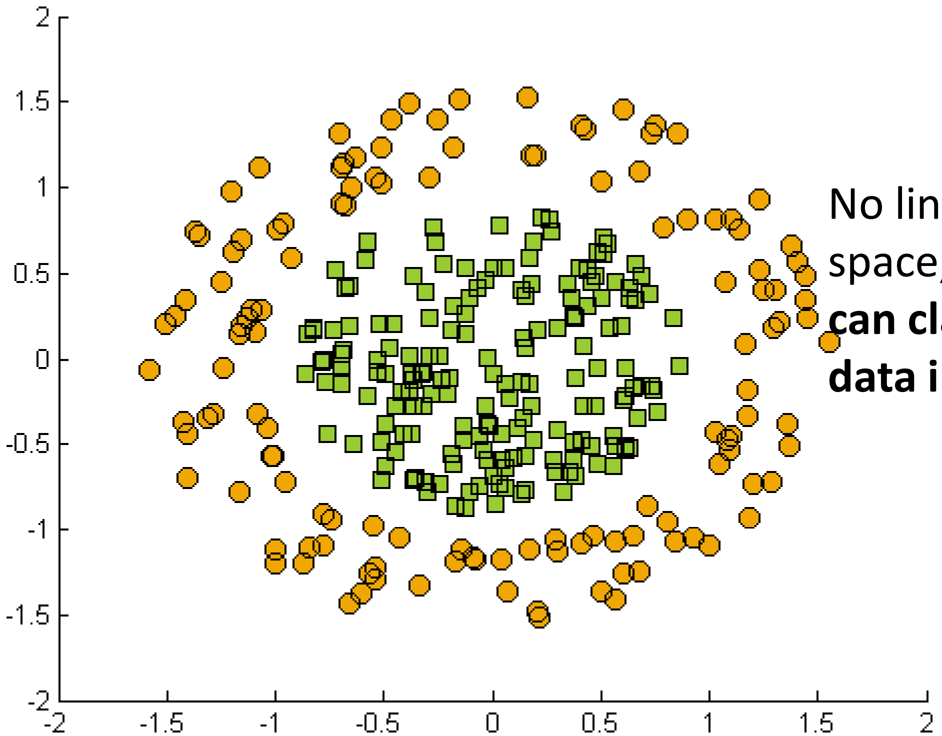


Naïve solution: Transform the input data into a higher dimension using the following nonlinear transformation:

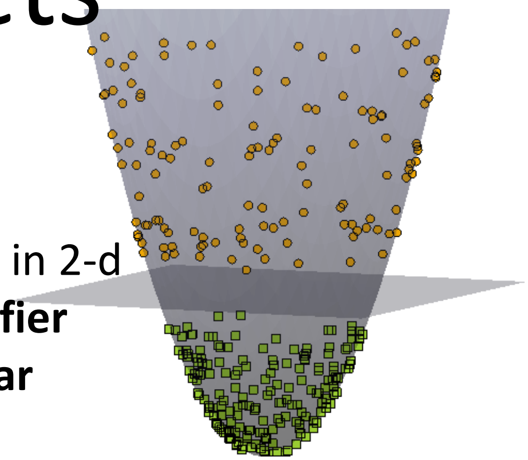
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{bmatrix}$$

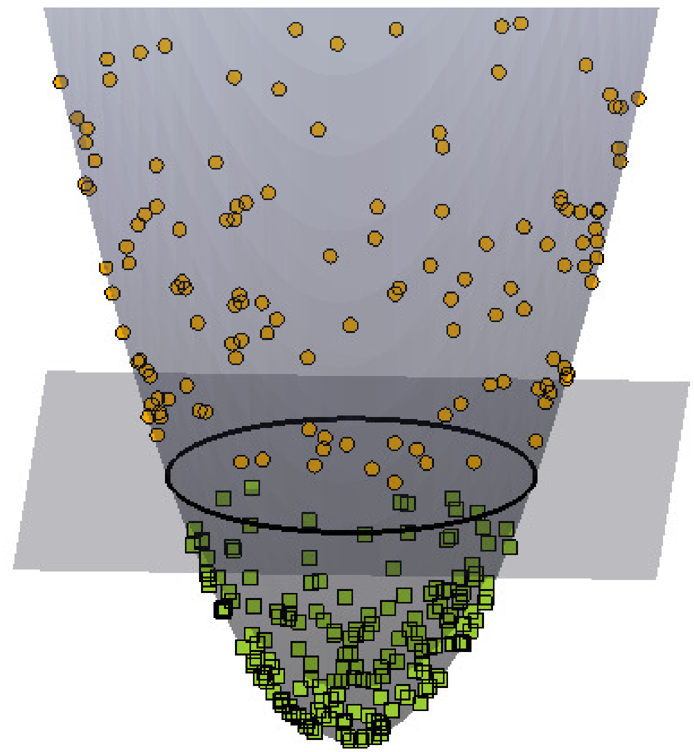


Nonlinear Data Sets



No linear classifier exists in 2-d space, but a **linear classifier can classify this nonlinear data in 3-d space!**

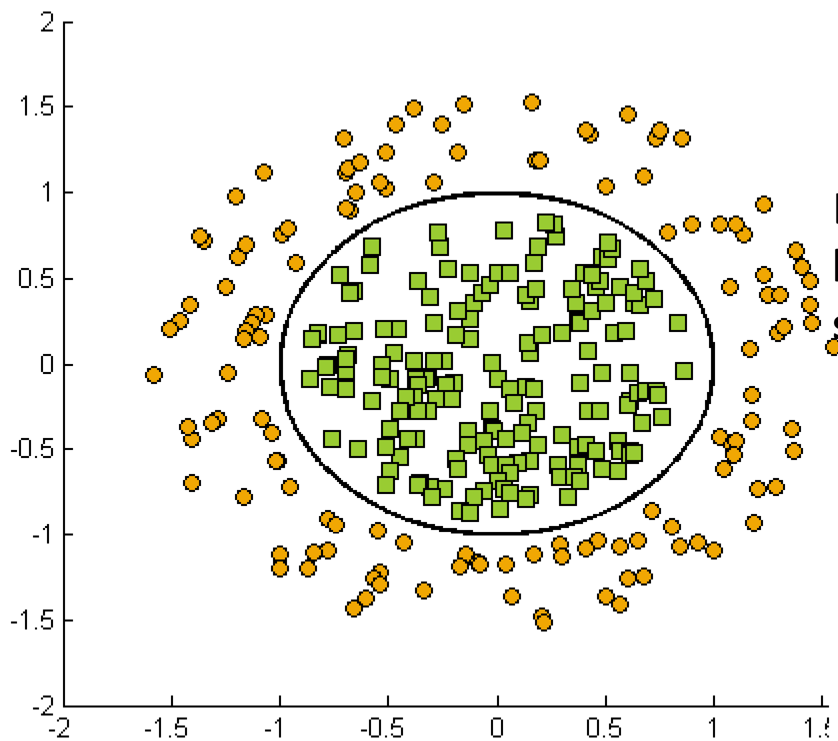




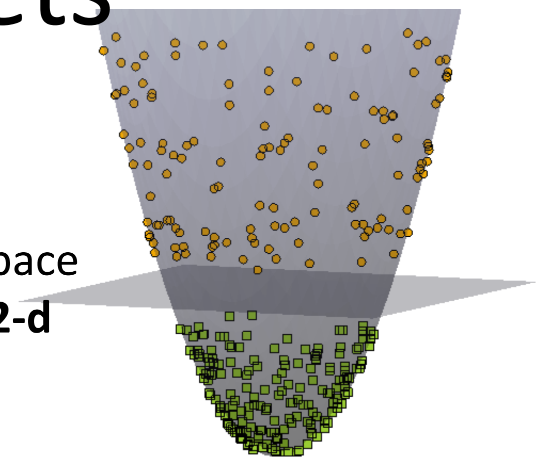
Naïve solution: Transform the input data into a higher dimension using the following nonlinear transformation**:**

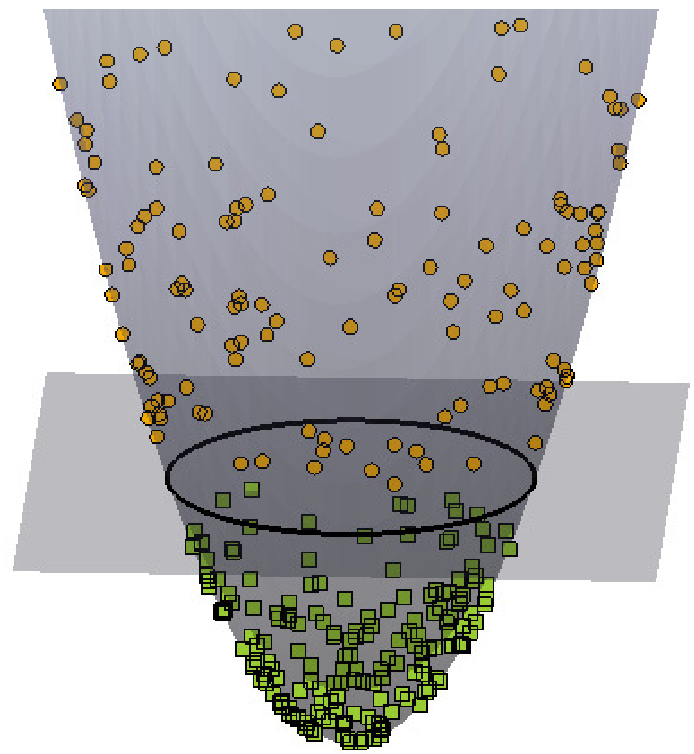
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1 x_2 \\ x_2^2 \end{bmatrix}$$

Nonlinear Data Sets



Linear classifier in 3-d space **becomes non-linear in 2-d space!**

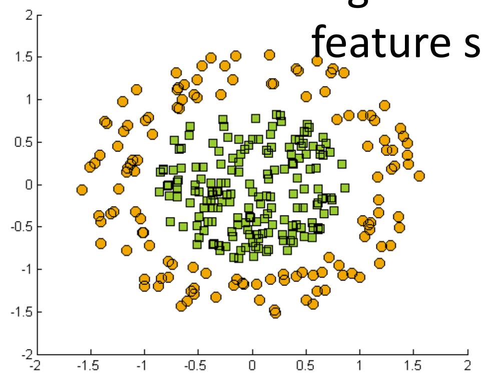




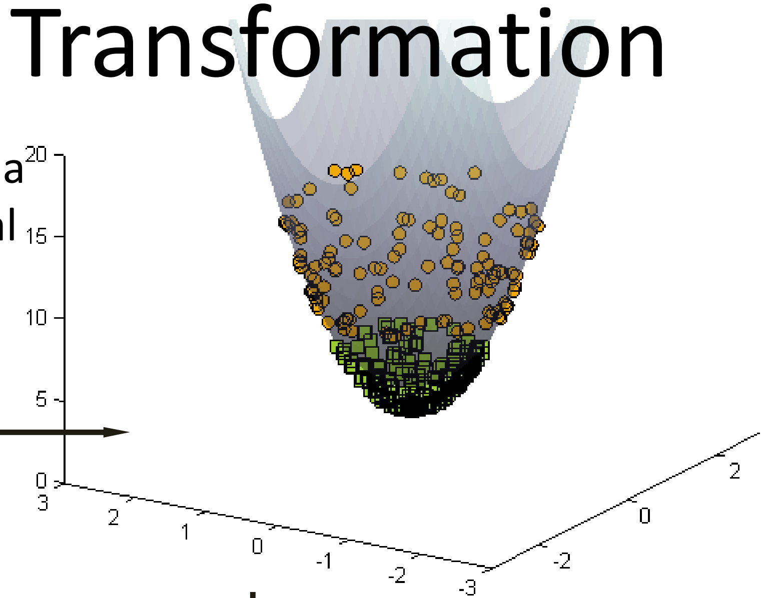
Naïve solution: Transform the input data into a higher dimension using the following nonlinear transformation:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

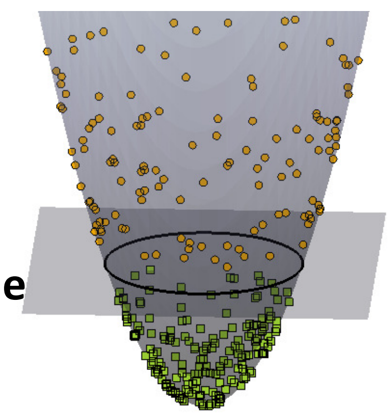
Naïve Approach: Explicit Transformation



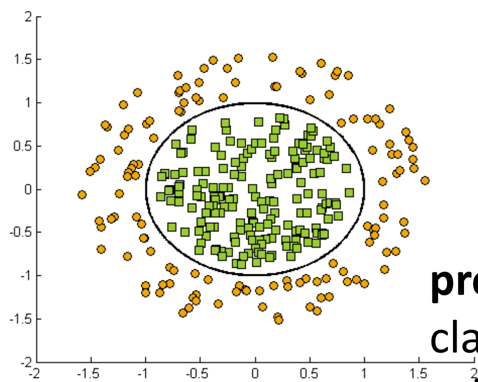
transform data to a higher dimensional feature space



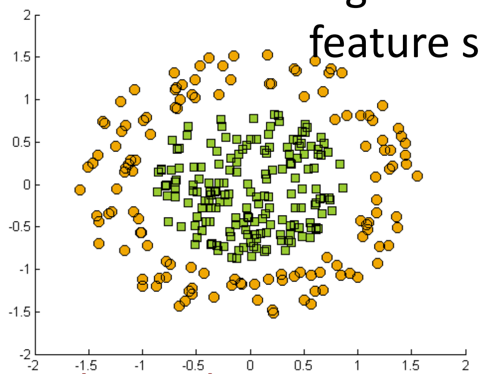
train a **classifier in the high dimensional space** using **support vector machines**



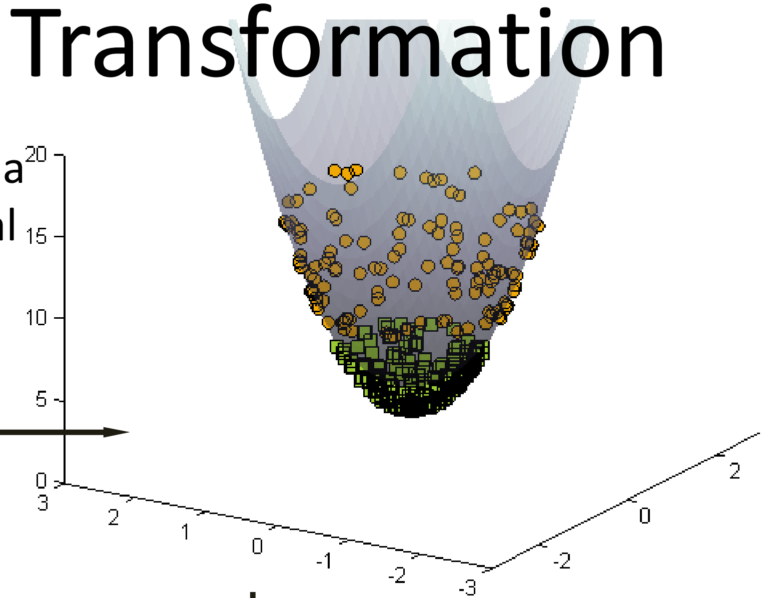
project the learned classifier **back to the original space**



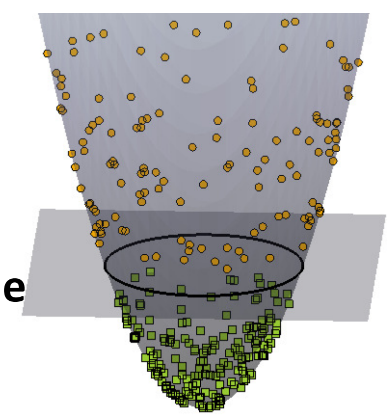
Naïve Approach: Explicit Transformation



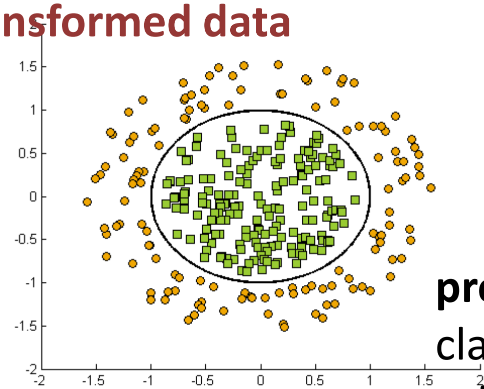
transform data to a higher dimensional feature space



train a classifier in the high dimensional space using **support vector machines**



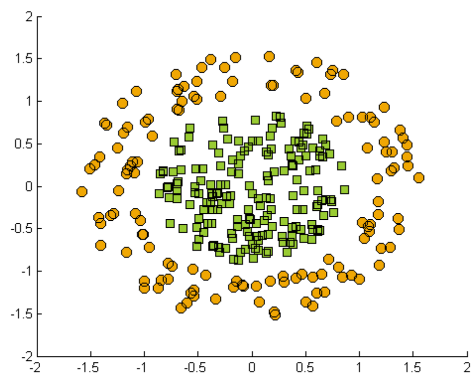
project the learned classifier **back to the original space**

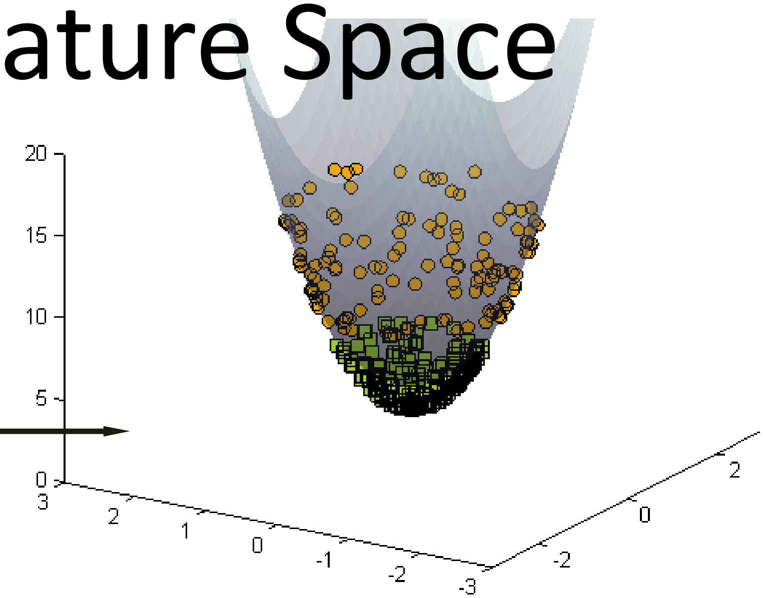


Recall that SVM training relies only on **inner products of the training data**.

In this case, it will rely on the inner-products of the **transformed data**

Inner Products in Feature Space





Let two points in the **original input space** be

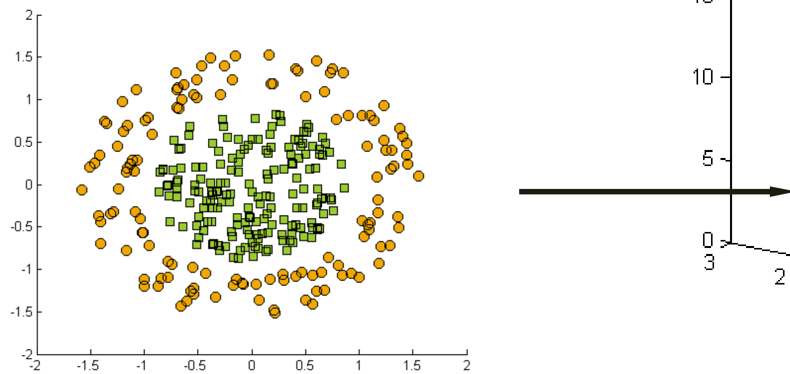
$$\mathbf{x} = (x_1,\, x_2)$$
$$\mathbf{z} = (z_1,\, z_2)$$

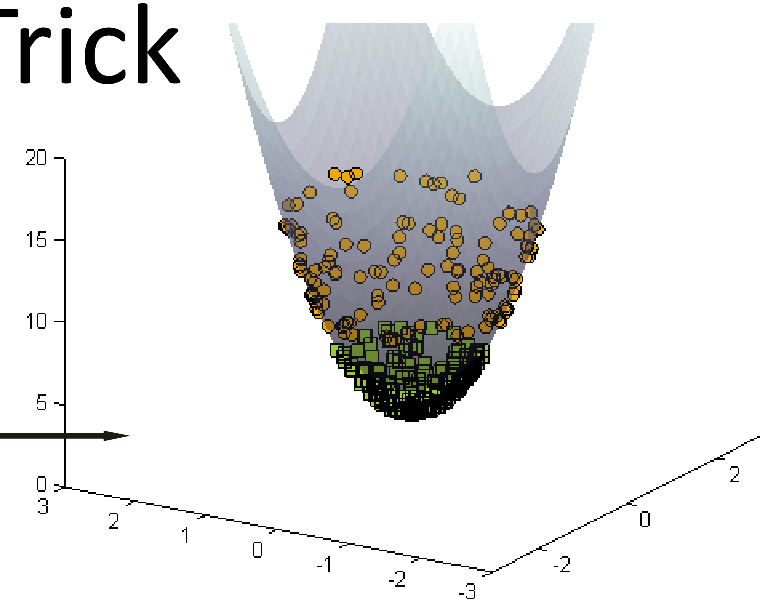
After transformation, in the **high-dimensional feature space**, they become

$$\phi(\mathbf{x}) = (x_1^2,\, \sqrt{2}x_1x_2,\, x_2^2)$$
$$\phi(\mathbf{z}) = (z_1^2,\, \sqrt{2}z_1z_2,\, z_2^2)$$

What do inner products in this **transformed feature space** look like?

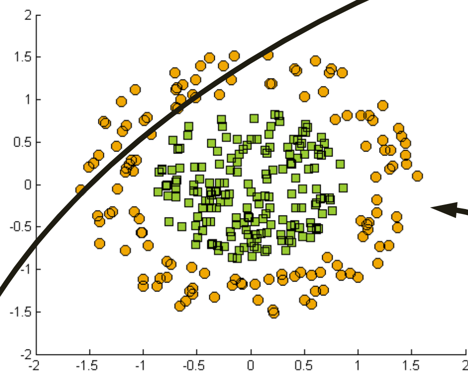
The Kernel Trick

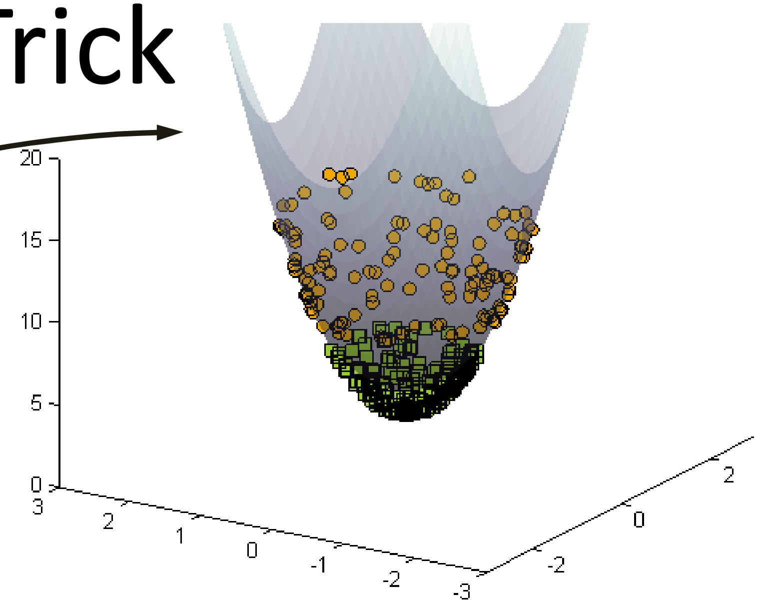




$$
\begin{aligned}
\langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2} x_1 x_2), (w_1^2, w_2^2, \sqrt{2} w_1 w_2) \rangle \\
&= x_1^2 w_1^2 + x_2^2 w_2^2 + 2 x_1 w_1 x_2 w_2 \\
&= (x_1 w_1 + x_2 w_2)^2 \\
&= \langle \mathbf{x}, \mathbf{w} \rangle^2
\end{aligned}
$$

The Kernel Trick





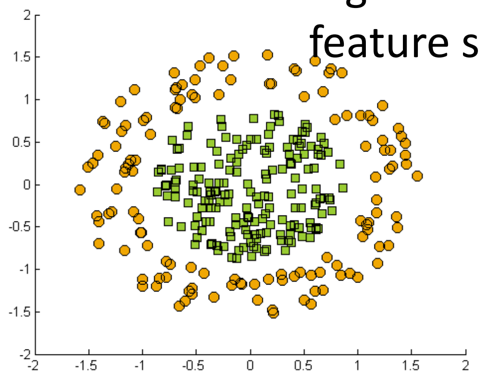
inner products in high-dimensional **feature space**

inner products in the original **input space**

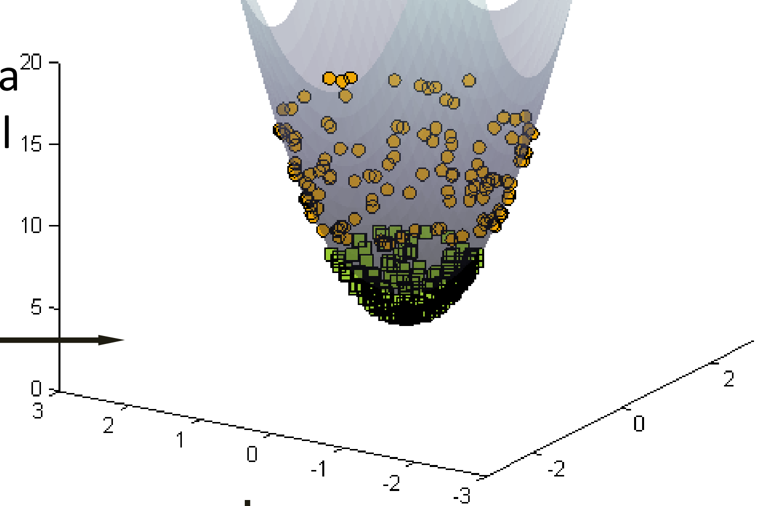
$$\langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle = \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (w_1^2, w_2^2, \sqrt{2}w_1w_2) \rangle$$
$$= x_1^2w_1^2 + x_2^2w_2^2 + 2x_1w_1x_2w_2$$
$$= (x_1w_1 + x_2w_2)^2$$
$$= \langle \mathbf{x}, \mathbf{w} \rangle^2$$

the two inner products are **related to each other through the kernel function**

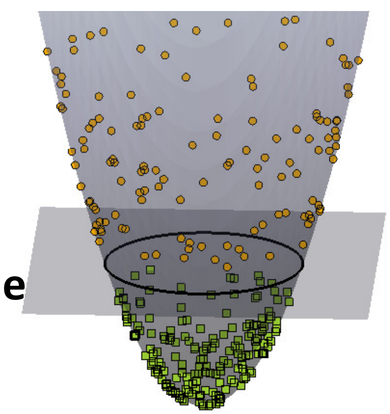
Better Approach: The Kernel Trick



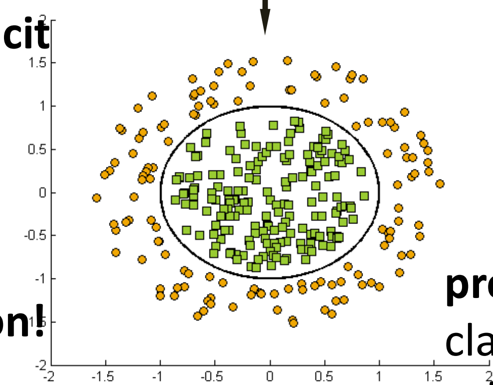
transform data to a higher dimensional feature space



train a **classifier in the high dimensional space** using **support vector machines**



project the learned classifier **back to the original space**

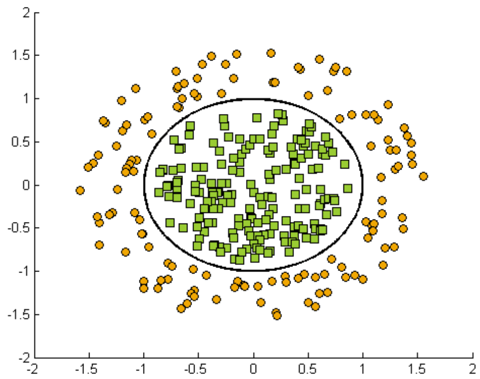


Use a **kernel function** to directly learn a nonlinear classifier!

No need for explicit transformations

Can use existing approaches with slight modification!

The Kernel Trick



Use a **kernel function** to directly learn a nonlinear classifier!

No need for explicit transformations

Can use existing approaches with slight modification!

linear support vector machine

$$\max \quad -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \mathbf{x}_i'\mathbf{x}_j + \sum_{i=1}^{\ell}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \quad \forall i = 1\ldots\ell$$

kernel support vector machine

$$\max \quad -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \kappa(\mathbf{x}_i,\, \mathbf{x}_j) + \sum_{i=1}^{\ell}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C \quad \forall i = 1\ldots\ell$$

Some Popular Kernels

Some popular kernels

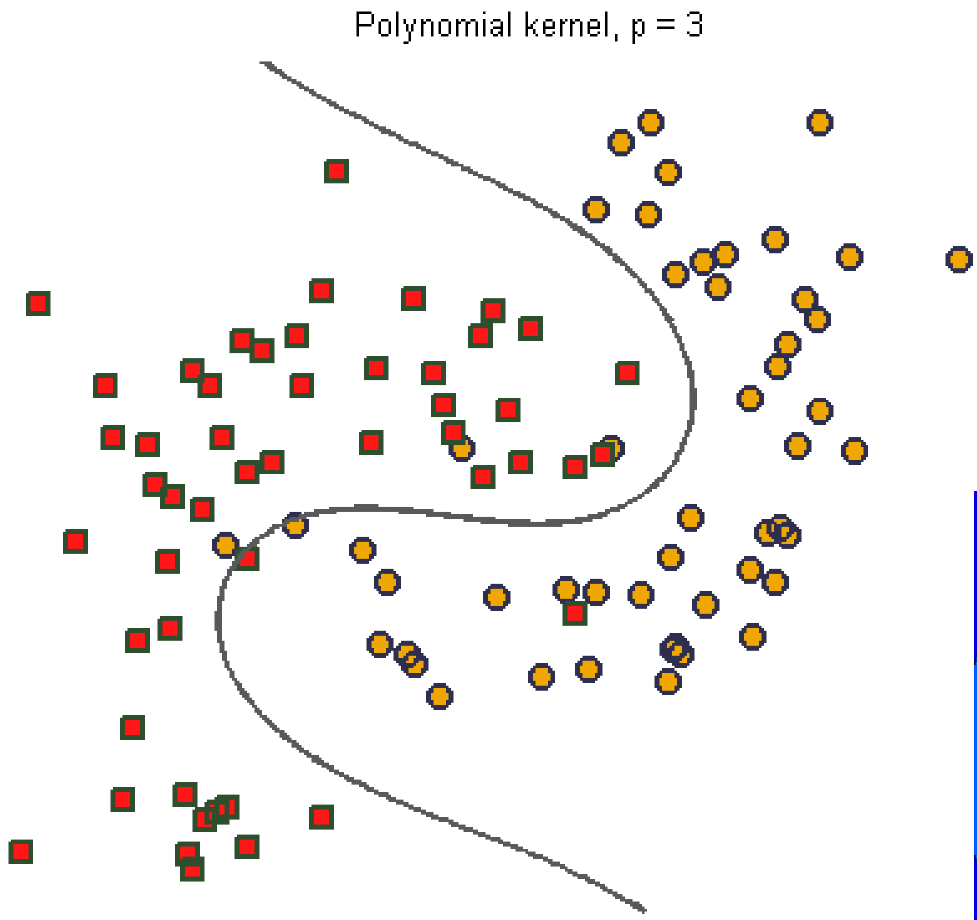
- Linear kernel: $\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$
- Polynomial kernel: $\kappa(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + c)^d$, $c, d \geq 0$
- Gaussian kernel: $\kappa(\mathbf{x}, \mathbf{z}) = e^{-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma}}$, $\sigma > 0$
- Sigmoid kernel: $\kappa(\mathbf{x}, \mathbf{z}) = \tanh^{-1} \eta \langle \mathbf{x}, \mathbf{z} \rangle + \theta$

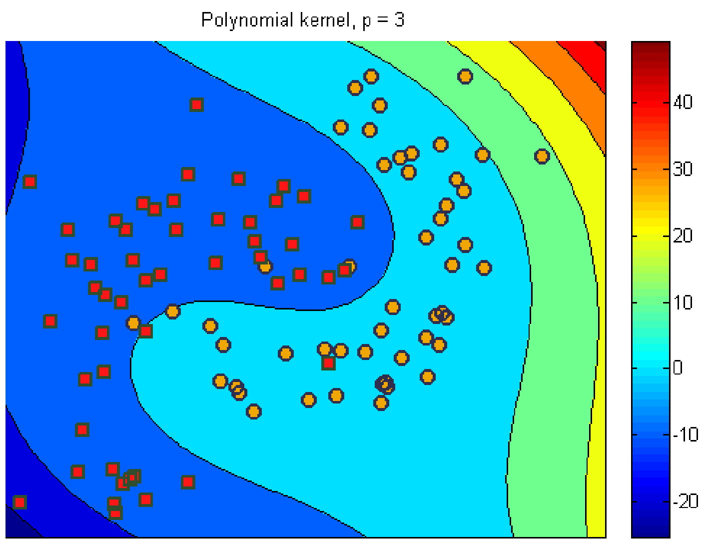
Kernels can also be constructed from other kernels:

- Conical (not linear) combinations, $\kappa(\mathbf{x}, \mathbf{z}) = a_1 \kappa_1(\mathbf{x}, \mathbf{z}) + a_2 \kappa_2(\mathbf{x}, \mathbf{z})$
- Products of kernels, $\kappa(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z}) \kappa_2(\mathbf{x}, \mathbf{z})$
- Products of functions, $\kappa(\mathbf{x}, \mathbf{z}) = f_1(\mathbf{x}) f_2(\mathbf{z})$, f_1, f_2 are real valued functions.

Polynomial Kernels

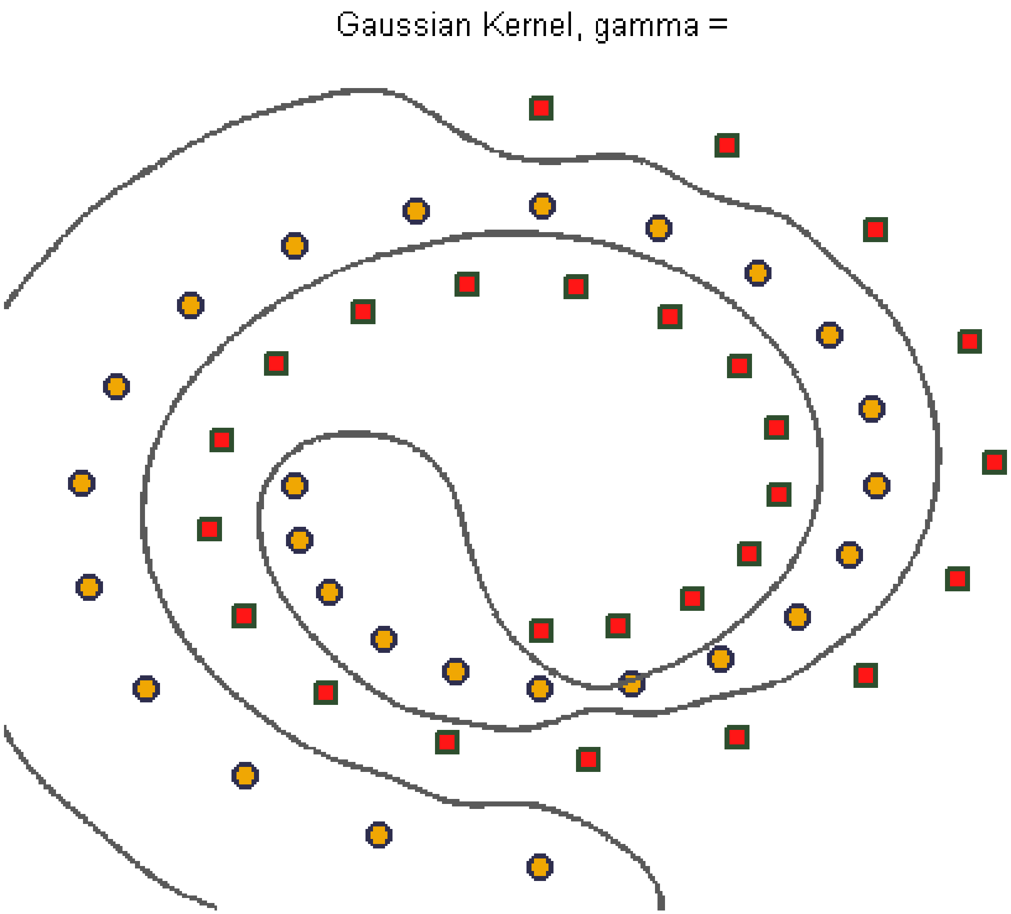
$$\kappa(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + 1)^d$$

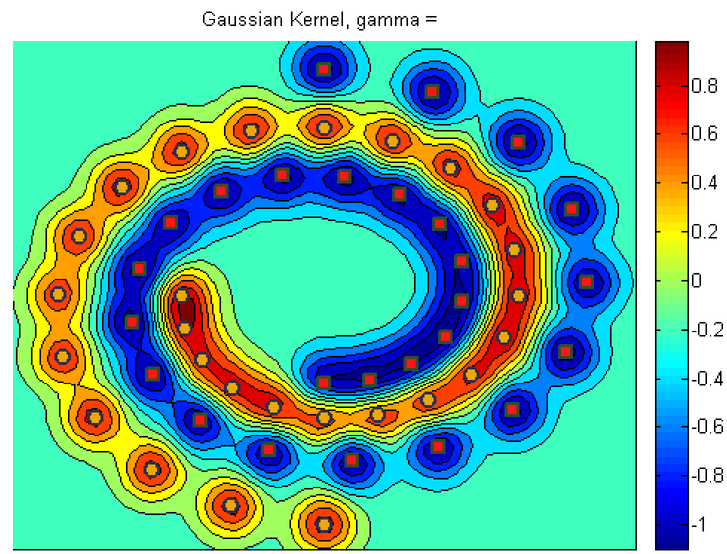




Gaussian Kernels

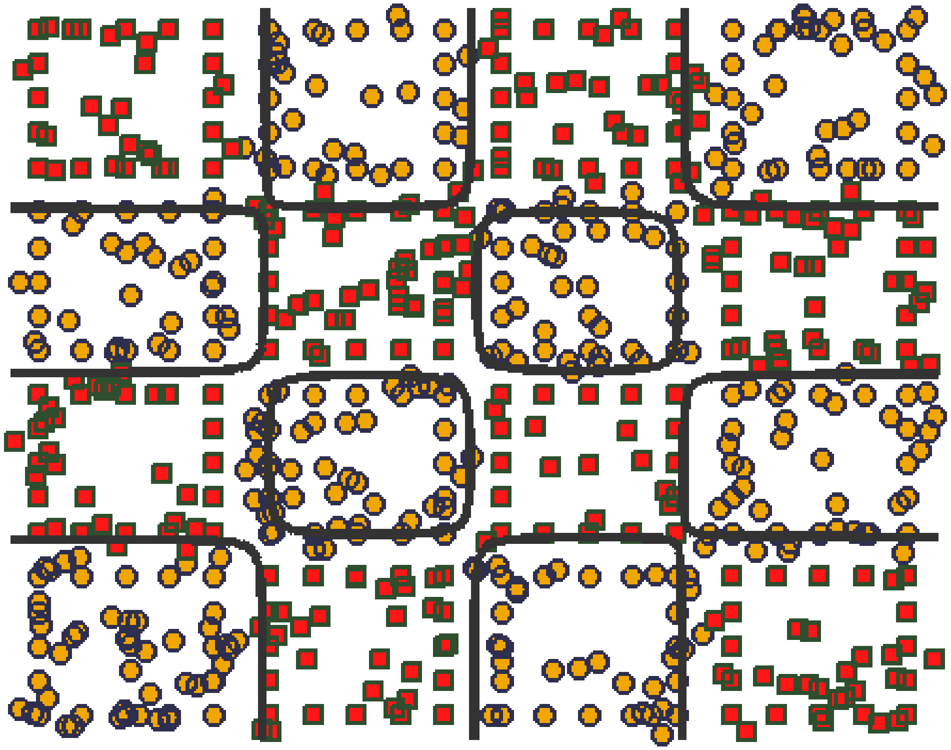
$$\kappa(\mathbf{x}, \mathbf{z}) = \exp -\frac{\|\mathbf{x}-\mathbf{z}\|^2}{\sigma}$$

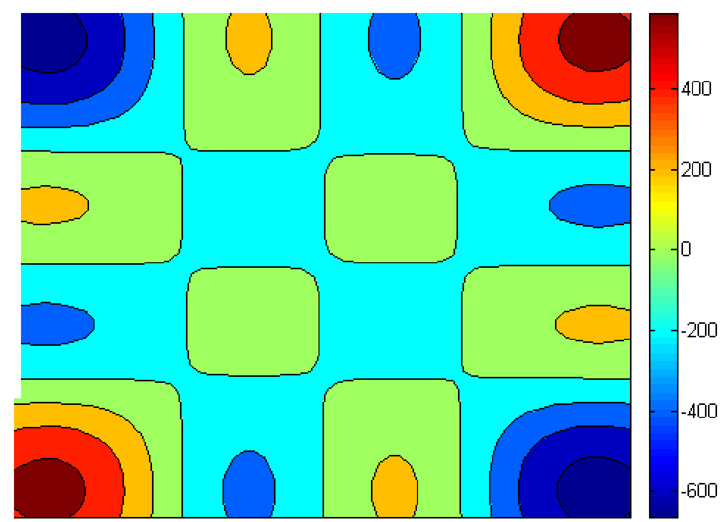




Gaussian Kernels

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp - \frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma}$$





Outline

- Example of classification problems
- Formulating Support Vector Machines
- SVM Properties
- Soft-Margin SVMs
- Kernels and Nonlinear Data
- **Regularization**

Regularization and Over-fitting

soft-margin support vector machine

$$\min \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}\xi$$

$$\text{s.t.} \quad y_i(\mathbf{w}'\mathbf{x}_i - b) \geq 1 - \xi_i \quad \forall i = 1 \ldots \ell$$

$$\xi \geq 0$$

kernel support vector machine

$$\max \quad -\frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\alpha_i\alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i=1}^{\ell}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{\ell}\alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C \quad \forall i = 1 \ldots \ell$$

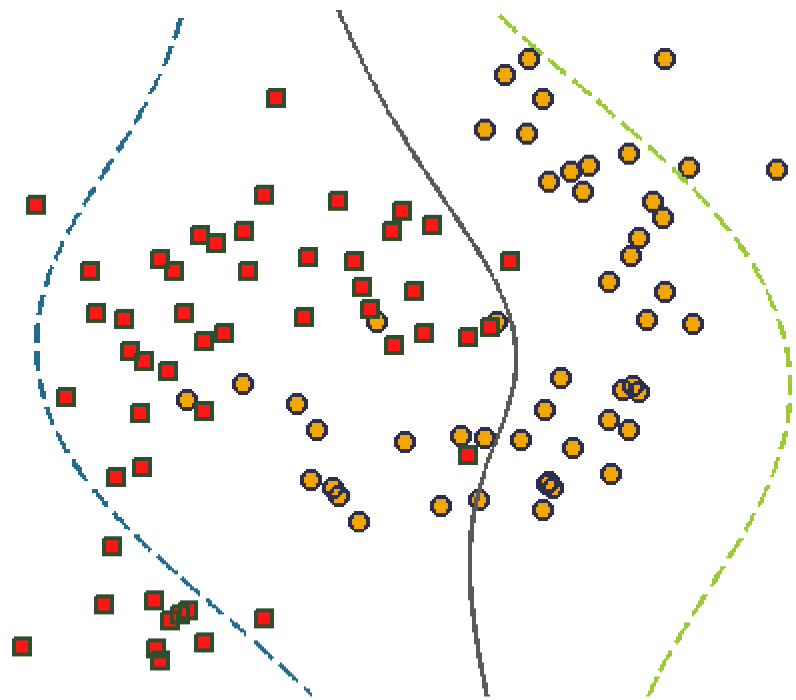
The **regularization parameter, C**, is chosen **a priori**, defines the relative **trade-off between norm** (complexity / smoothness / capacity) **and loss** (error penalization)

We want to find classifiers that *minimize* (**regularization** + ***C*** • **loss**)

Regularization
- introduces **inductive bias** over solutions
- controls the **complexity** of the solution
- imposes **smoothness restriction** on solutions

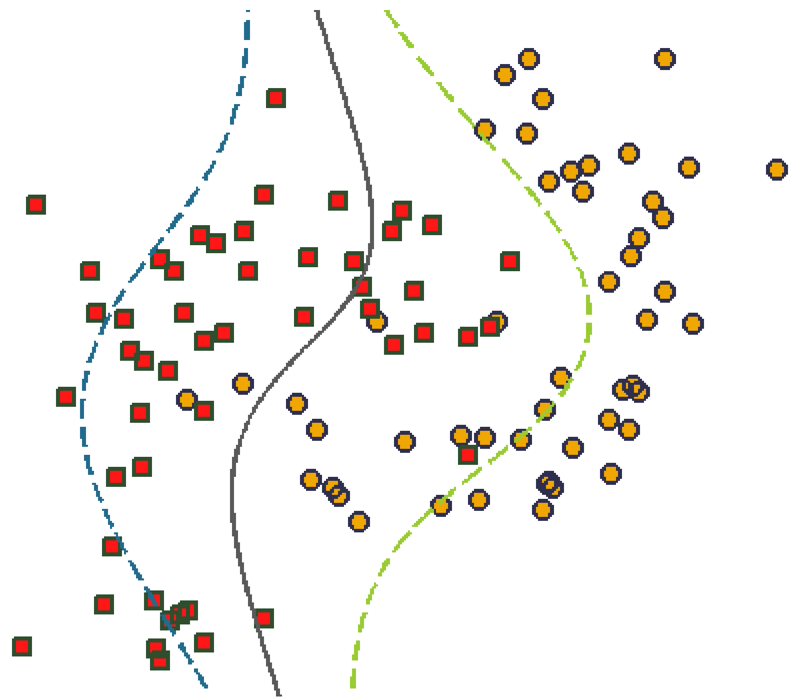
As **C increases**, the effect of the **regularization decreases** and the **SVM tends to overfit the data**

The Effect of C on Classification



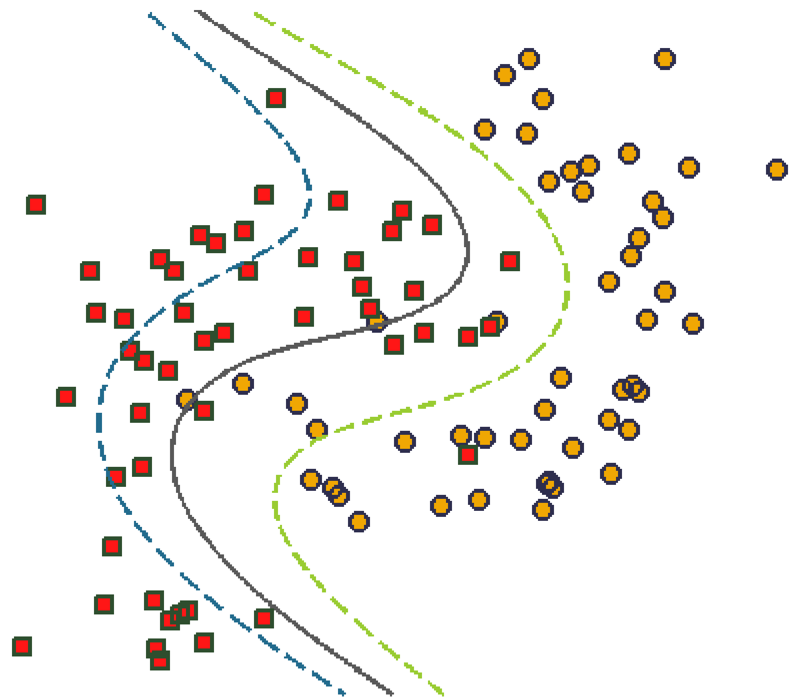
C = 0.001

The Effect of C on Classification



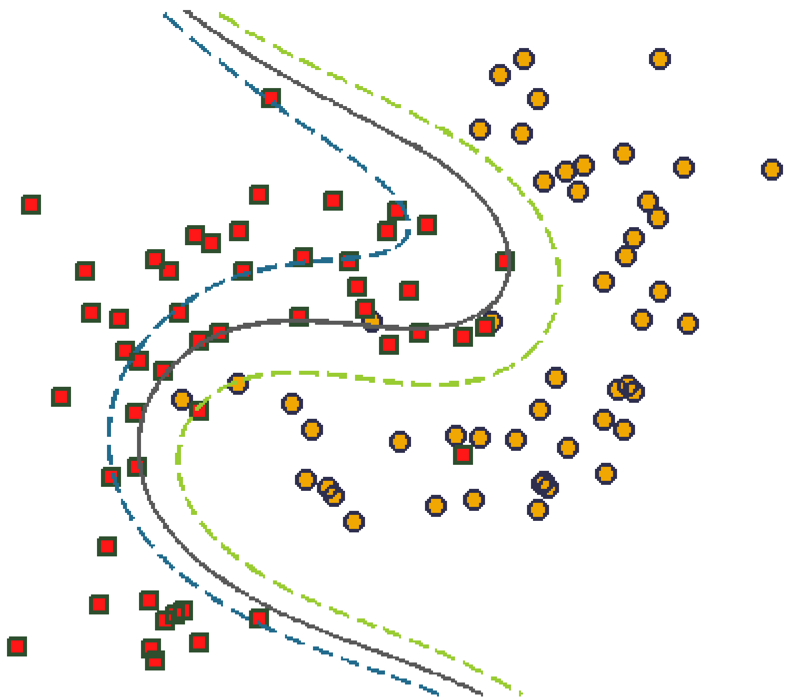
C = 0.01

The Effect of C on Classification



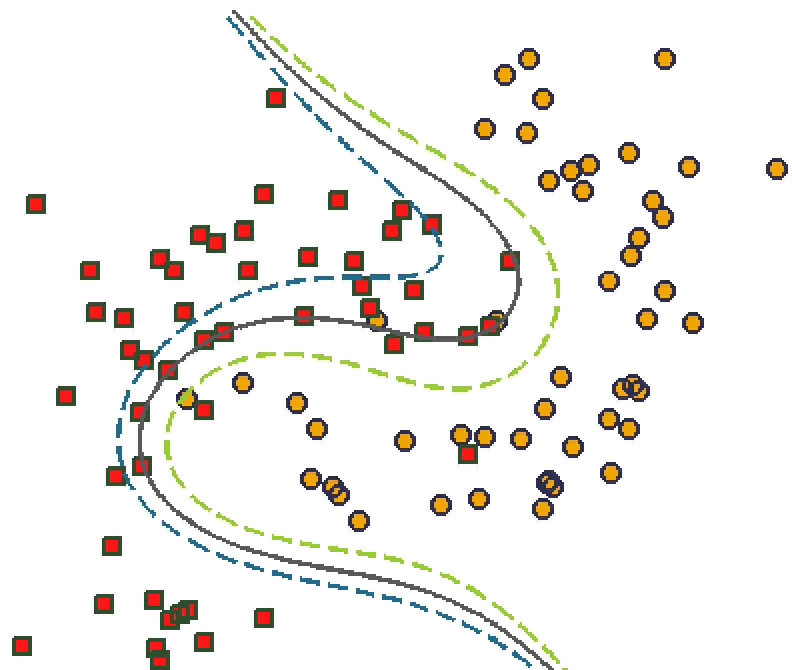
C = 0.1

The Effect of C on Classification



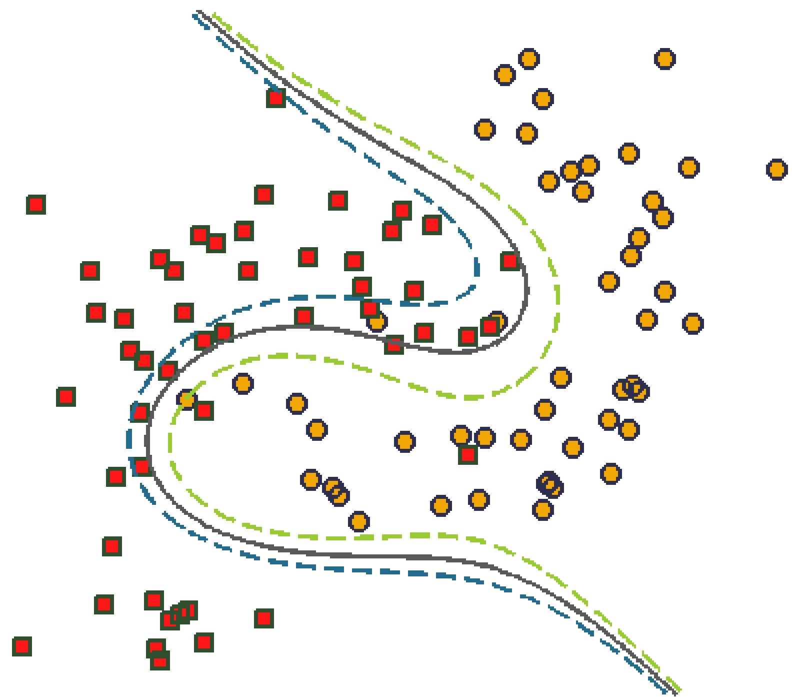
C = 1

The Effect of C on Classification



***C* = 10**

The Effect of C on Classification



C = 100

SVM Algorithms over the Years

- **Earliest solution approaches**: Quadratic Programming Solvers (CPLEX, LOQO, Matlab QP, SeDuMi)
- **Decomposition methods**: SVM chunking (Osuna et. al., 1997); implementation: SVMlight (Joachims, 1999)
- **Sequential Minimization Optimization** (Platt, 1999); implementation: LIBSVM (Chang et. al., 2000)
- **Interior Point Methods** (Munson and Ferris, 2006), **Successive Over-relaxation** (Mangasarian, 2004)
- **Co-ordinate Descent Algorithms** (Keerthi et. al., 2009), **Bundle Methods** (Teo et. al., 2010)