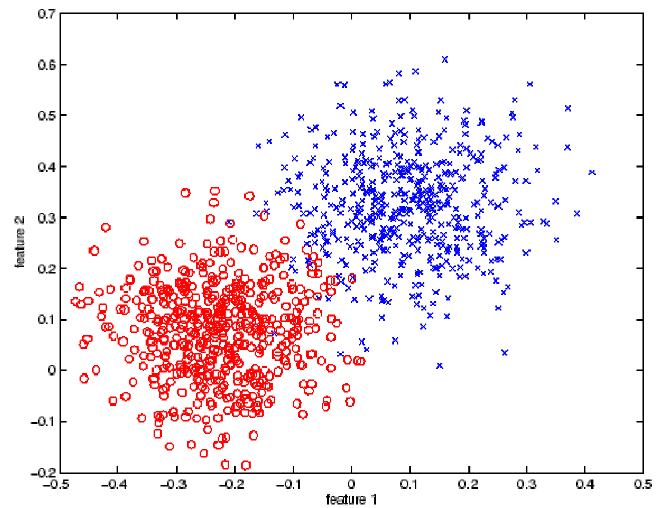
The Support Vector Machine

Nuno Vasconcelos
(Ken Kreutz-Delgado)

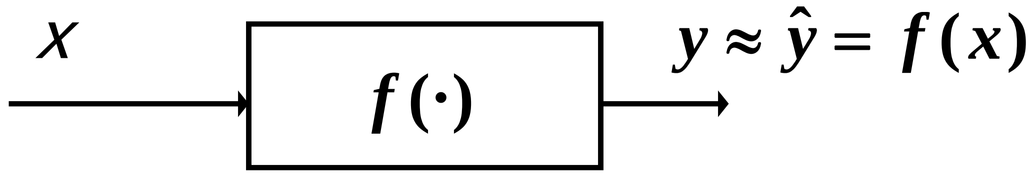
UC San Diego

Classification

- a Classification Problem has two types of variables
  - X - vector of observations (features) in the world
  - Y - state (class) of the world
- E.g.
  - $X \in \mathcal{X} \subset \mathbb{R}^2$, X = (fever, blood pressure)
  - $Y \in \mathcal{Y}$ = {disease, no disease}



- X, Y are stochastically related and this relationship can be well approximated by an "optimal" classifier function



- Goal: Design a "good" classifier $h \approx f \approx y$, $h: \mathcal{X} \rightarrow \mathcal{Y}$

Loss Functions and Risk

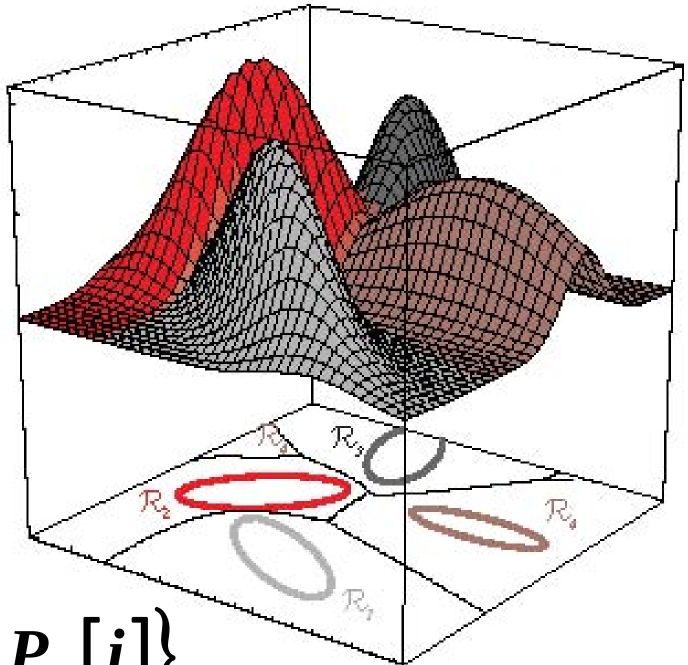
- Usually $h(\cdot)$ is a parametric function, $h(x,\alpha)$
- Generally it cannot estimate the *value* y arbitrarily well
  - Indeed, the best we can (optimistically) hope for is that h will well approximate the unknown optimal classifier f, $h \approx f$
- We define a loss function: $L[y, h(x,\alpha)]$
- Goal: Find the parameter values (equivalently, find the classifier) that minimize the expected value of the loss:

Risk = Average Loss = $R(\alpha) = E_{X,Y}\left\{L[y, h(x,\alpha)]\right\}$

- In particular, under the "0-1" loss the optimal solution is the Bayes Decision Rule (BDR):

$$h^*(x) = \arg\max_i P_{Y|X}\left[i \mid x\right]$$

Bayes Decision Rule



- The BDR carves up the observation space $\mathcal{X}$, assigning a label to each region
- Clearly, *h** depends on the class densities

$$h^*(x) = \arg\max_i \left\{ \log P_{X|Y}\left[x|i\right] + \log P_Y\left[i\right] \right\}$$

- Problematic! Usually we don't know these densities!!
- Key idea of discriminant learning:
  - First estimating the densities, followed by deriving the decision boundaries is a computationally intractable (hence bad) strategy
  - Vapnik's Rule: "When solving a problem avoid solving a more general (and thus usually much harder) problem as an intermediate step!"

Discriminant Learning

- Work directly with the decision function
  1. Postulate a (parametric) family of decision boundaries
  2. Pick the element in this family that produces the best classifier
- Q: What is a good family of decision boundaries?
- Consider two equal probability Gaussian class conditional densities of equal covariance:

$$h^*(x) = \arg\max_{i} \left\{ \log G(x, \mu_i, \Sigma_i) + \log \frac{1}{2} \right\}$$

$$= \arg\min_{i} \left\{ (x - \mu_i)^T \Sigma^{-1} (x - \mu_i) \right\}$$

$$= \begin{cases} 0, & \text{if} \quad (x-\mu_0)^T \Sigma^{-1} (x-\mu_0) < (x-\mu_1)^T \Sigma^{-1} (x-\mu_1) \\ 1, & \text{otherwise} \end{cases}$$

The Linear Discriminant Function

- The decision boundary is the set of points

$$(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)=(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)$$

  which, after some algebra, becomes

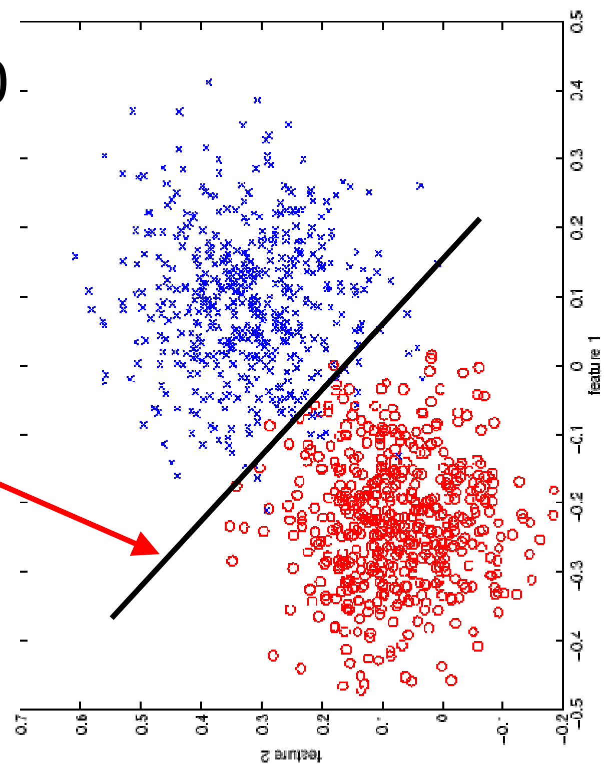
$$2(\mu_1-\mu_0)^T\Sigma^{-1}x+\mu_0{}^T\Sigma^{-1}\mu_0-\mu_1{}^T\Sigma^{-1}\mu_1=0$$

- This is the equation of the hyperplane

$$\boldsymbol{w}^T\boldsymbol{x}+\boldsymbol{b}=\boldsymbol{0}$$

  with

$$\boldsymbol{w}=2\Sigma^{-1}(\mu_1-\mu_0)$$
$$\boldsymbol{b}=\mu_0{}^T\Sigma^{-1}\mu_0-\mu_1{}^T\Sigma^{-1}\mu_1$$

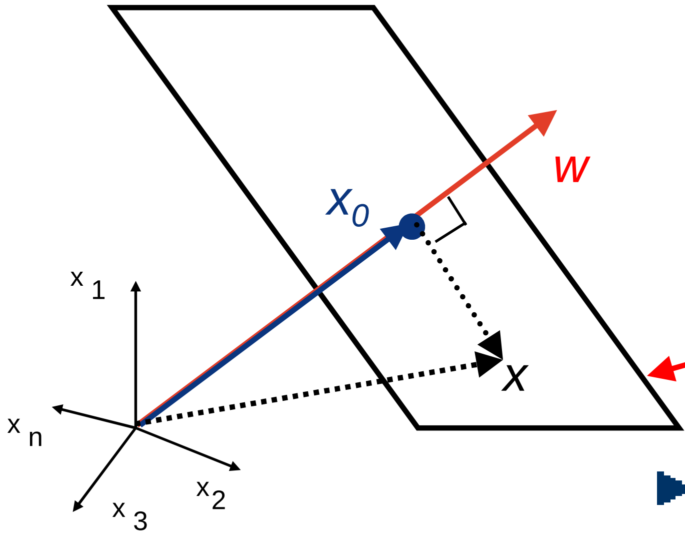


- This is a linear discriminant

Linear Discriminants

- The hyperplane equation can also be written as

$$w^T x + b = 0 \Leftrightarrow w^T \left(x + \frac{w}{\|w\|^2} b \right) = 0 \Leftrightarrow$$



$$w^T \left(x - x_0 \right) = 0$$ with $$x_0 = -b \frac{w}{\|w\|^2}$$

- Geometric interpretation
  - Hyperplane of normal w
  - Hyperplane passes through x_0
  - Hyperplane point x_0 is the point closest to the origin

Linear Discriminants

- For the given model, the quadratic discriminant function

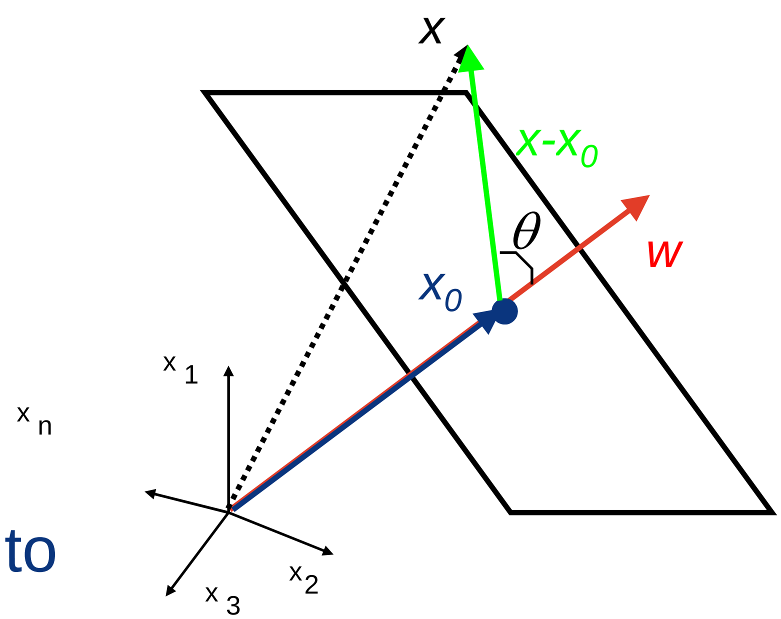
$$h^*(x) = \begin{cases} 0, & \text{if} \quad (x-\mu_0)^T \Sigma^{-1} (x-\mu_0) < (x-\mu_1)^T \Sigma^{-1} (x-\mu_1) \\ 1, & \text{if} \quad (x-\mu_0)^T \Sigma^{-1} (x-\mu_0) > (x-\mu_1)^T \Sigma^{-1} (x-\mu_1) \end{cases}$$

- is equivalent to the linear discriminant function

$$h^*(x) = \begin{cases} 0 & \text{if } g(x) > 0 \\ 1 & \text{if } g(x) < 0 \end{cases}$$

- where

$$\begin{aligned} g(x) &= w^T (x - x_0) \\ &= \|\mathbf{w}\| \cdot \|x - x_0\| \cdot \cos\theta \end{aligned}$$



- $g(x) > 0$ if x is on the side w points to ("w points to the positive side")

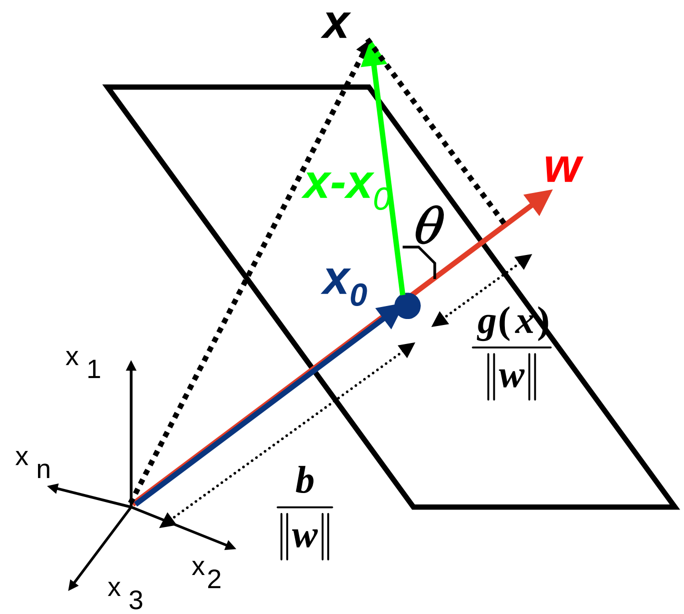
Linear Discriminants

- Finally, note that

$$\frac{g(x)}{\|w\|} = \frac{w^T}{\|w\|}(x - x_0)$$

is:

  - The projection of x-x_0 onto the unit vector in the direction of w
  - The length of the component of x-x_0 orthogonal to the plane



- I.e. $g(x)/||w||$ = perpendicular distance from x to the plane
- Similarly, $|b|/||w||$ is the distance from the plane to the origin, since:

$$x_0 = -b\frac{w}{\|w\|^2}$$

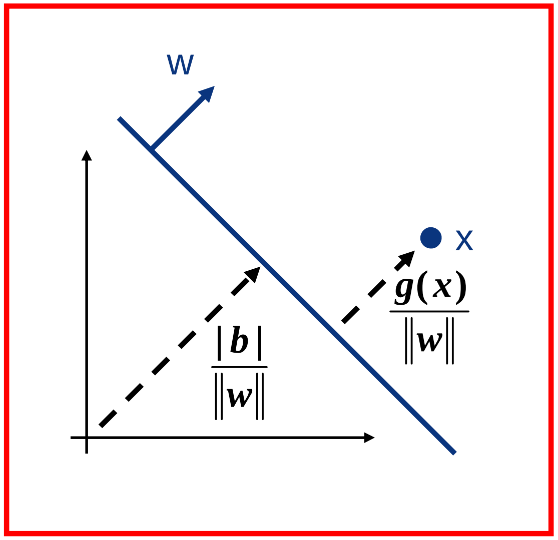
Geometric Interpretation

- Summarizing, the linear discriminant decision rule

$$h^*(x) = \begin{cases} 0 & \text{if } g(x) > 0 \\ 1 & \text{if } g(x) < 0 \end{cases}$$ with $$g(x) = w^T x + b$$

has the following properties

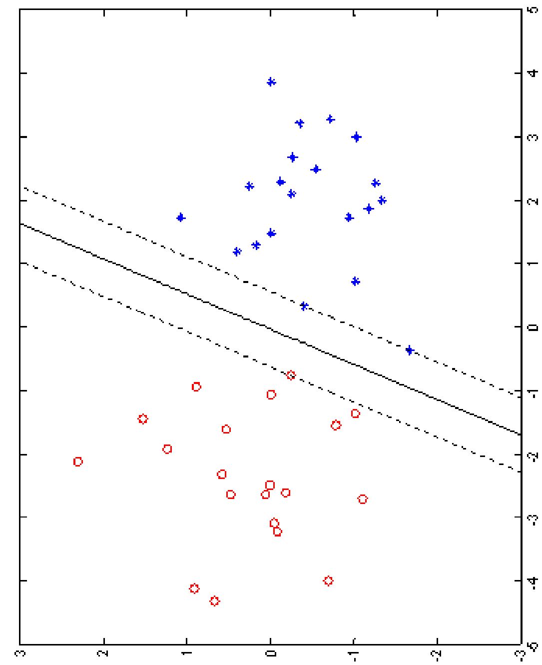
- It divides $\mathcal{X}$ into two “half-spaces”
- The boundary is the hyperplane with:
  - normal *w*
  - distance to the origin *b*/||*w*||
- g(*x*)/||*w*|| gives the signed distance from point *x* to the boundary
  - g(x) = 0 for points on the plane
  - g(x) > 0 for points on the side *w* points to (“positive side”)
  - g(x) < 0 for points on the “negative side”



The Linear Discriminant Function

- When is it a good decision function?
- We've just seen that it is optimal for
  - Gaussian classes having equal class probabilities and covariances

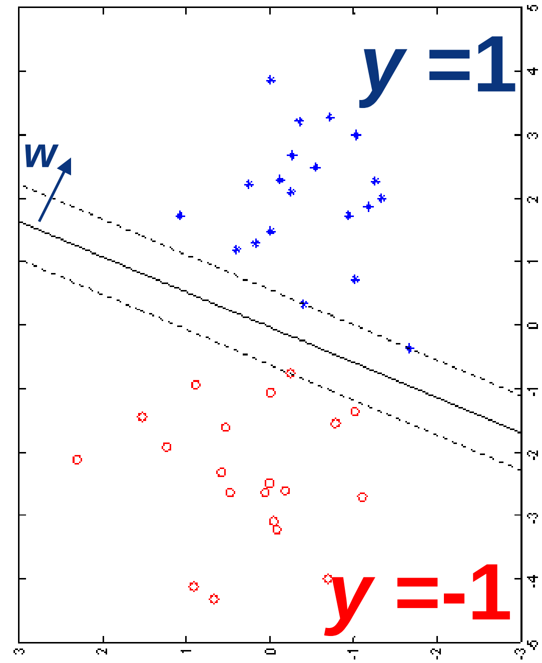
  But, this sounds too much like an artificial, toy problem
- However, it is also optimal if the data is linearly separable
  - I.e., if there is a hyperplane which has
    - all "class 0" data on one side
    - all "class 1" data on the other



- Note: this holding on the training set only guarantees optimality in the minimum training error sense, not in the sense of minimizing the true risk

Linear Discriminants

- For now, our goal is to explore the simplicity of the linear discriminant
- let's assume linear separability of the training data
- One handy trick is to use class labels $y \in \{-1,1\}$ instead of $y \in \{0,1\}$, *where*
  - y = 1 for points on the positive side
  - y = -1 for points on the negative side
- The decision function then becomes



$$h^*(x) = \begin{cases} 1 & \text{if } g(x) > 0 \\ -1 & \text{if } g(x) < 0 \end{cases} \Leftrightarrow \boxed{h^*(x) = \operatorname{sgn}\left[g(x)\right]}$$

Linear Discriminants & Separable Data

- We have a classification error if
  - $y = 1$ and $g(x) < 0$ or $y = -1$ and $g(x) > 0$
  - i.e., if $y\,g(x) < 0$
- We have a correct classification if
  - $y = 1$ and $g(x) > 0$ or $y = -1$ and $g(x) < 0$
  - i.e., if $y\,g(x) > 0$
- Note that, if the data is linearly separable, given a training set

$$D = \{(x_1,y_1), \cdots , (x_n,y_n)\}$$

we can have zero training error.

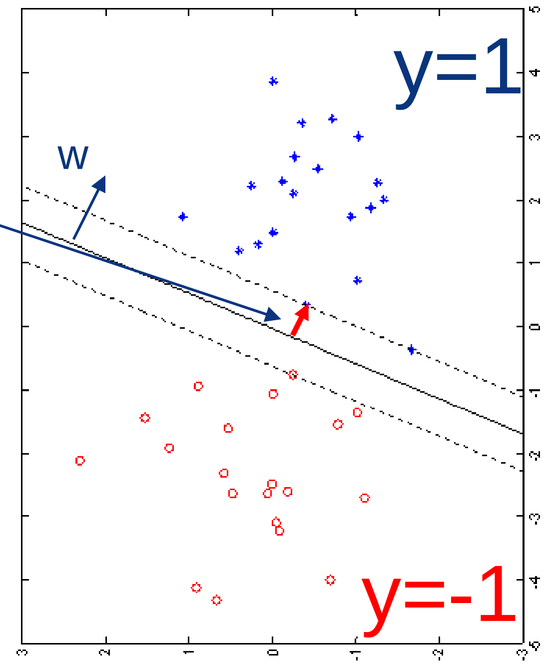
- The necessary & sufficient condition for this is that

$$y_i\left(w^T x_i + b\right) > 0, \quad \forall i = 1,\cdots,n$$

The Margin

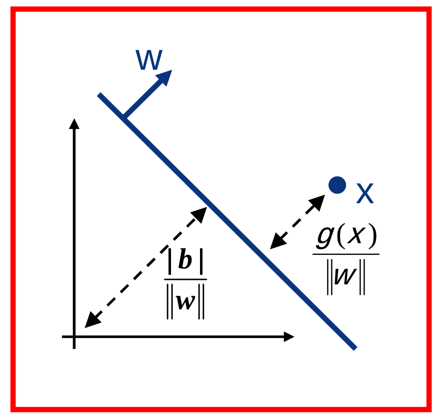
- The margin is the distance from the boundary to the closest point

$$\gamma = \min_i \frac{\left| \boldsymbol{w}^T \boldsymbol{x}_i + \boldsymbol{b} \right|}{\left\| \boldsymbol{w} \right\|}$$



- There will be no error on the training set if it is strictly greater than zero:

$$y_i \left(\boldsymbol{w}^T \boldsymbol{x}_i + \boldsymbol{b} \right) > 0, \ \forall i \quad \Leftrightarrow \quad \gamma > 0$$



- Note that this is ill-defined in the sense that γ does not change if both *w* and *b* are scaled by a common scalar λ
- We need a normalization

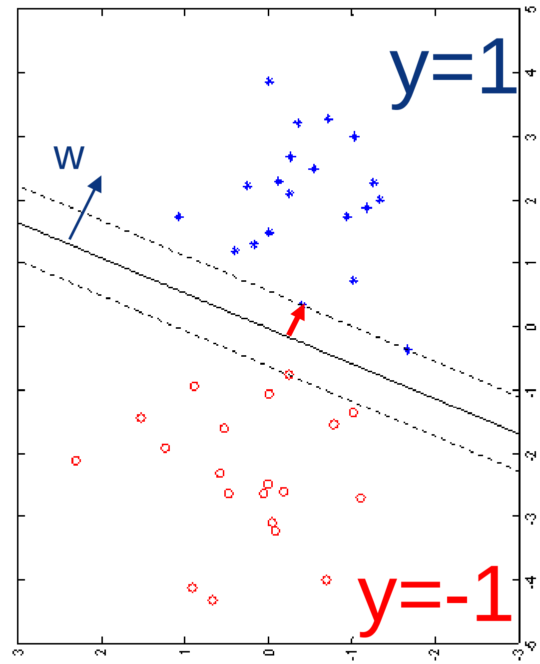
Support Vector Machine (SVM)

- A convenient normalization is to make $|g(x)| = 1$ for the closest point, i.e.

$$\min_i \left| w^T x_i + b \right| \equiv 1$$

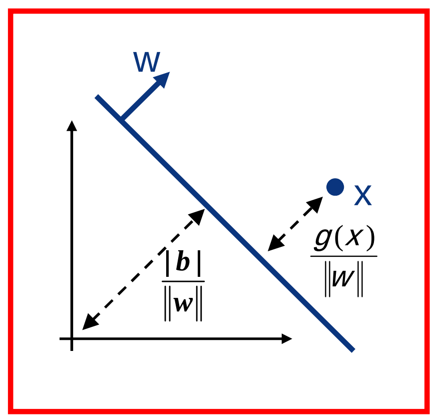
under which

$$\gamma = \frac{1}{\|w\|}$$



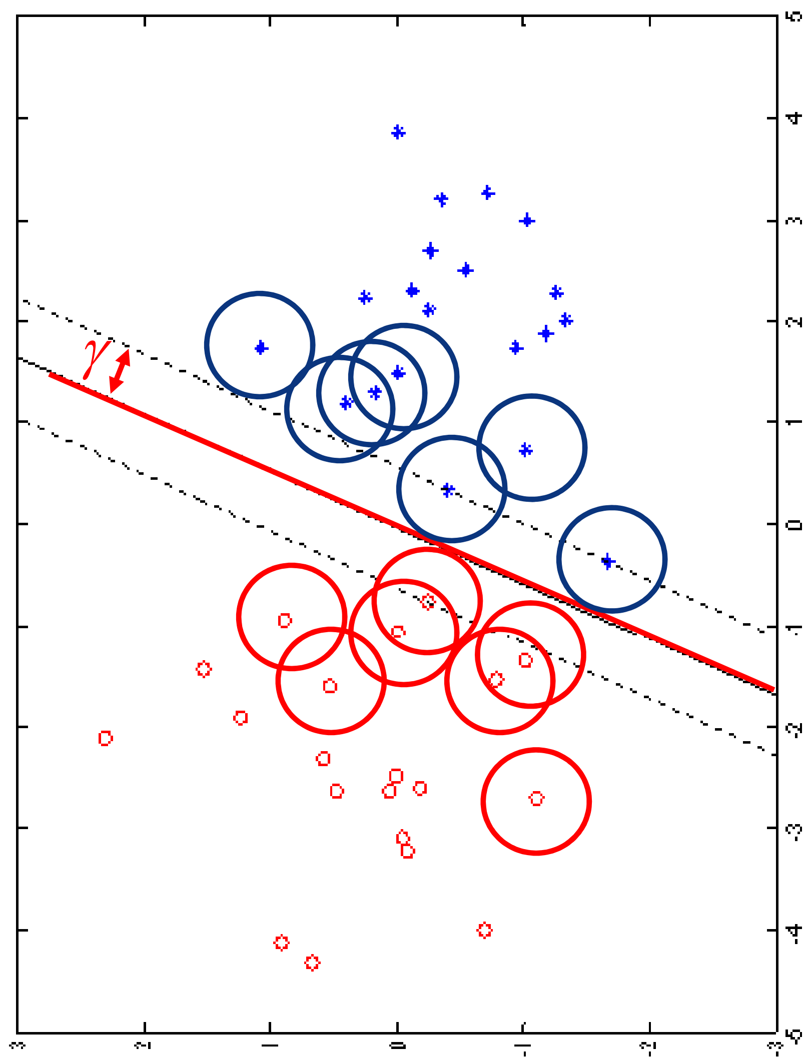
- The Support Vector Machine (SVM) is the linear discriminant classifier that maximizes the margin subject to these constraints:

$$\min_{w,b} \|w\|^2 \quad \text{subject to} \quad y_i\left(w^T x_i + b\right) \geq 1 \ \ \forall i$$



Maximizing the Margin

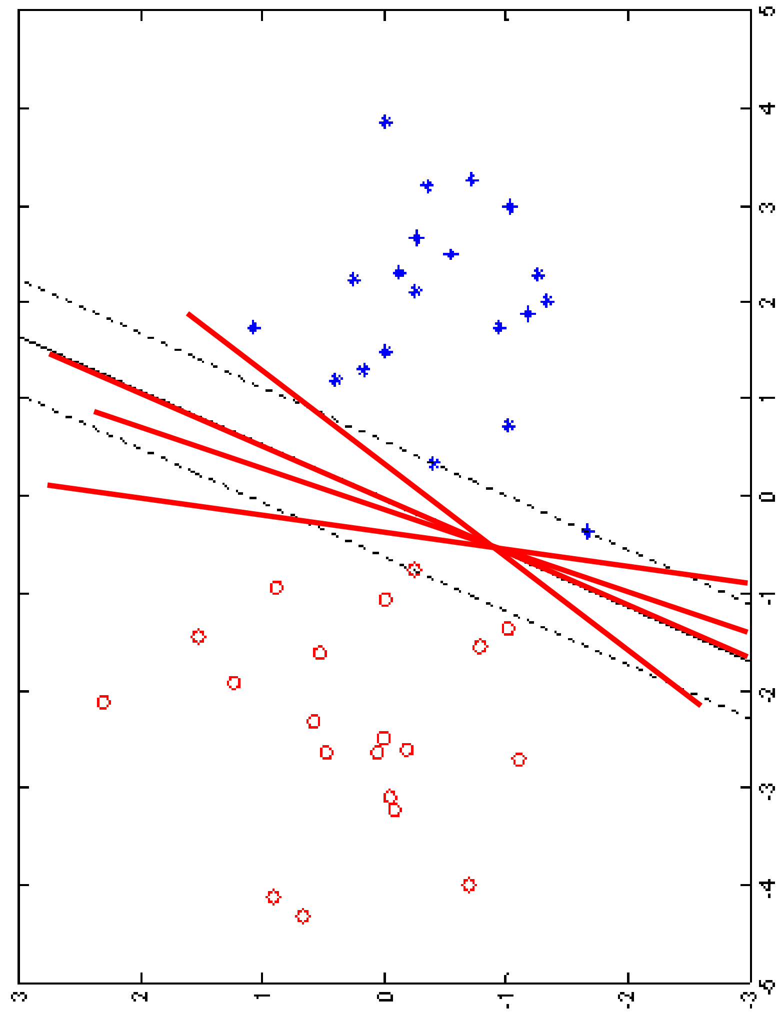
- **Intuition 1:**
  - Think of each point in the training set as a sample from a probability density centered on it
  - If we draw another sample, we will not get the same points
  - Thus each point is represents a pdf with a certain variance
  - The sum of all such “point-centerd pdfs” provides a density estimate (a so-called “kernel estimate”)
  - If we leave a margin of γ on the training set, we are safe against this “resampling” uncertainty (as long as the radius of support of a point pdf is smaller than γ)



  - Thus, the larger the value of γ, the more robust is the classifier when applied to new data!

Maximizing the Margin

- Intuition 2:
  - Think of the hyper plane as an uncertain estimate because it is learned from random data samples
  - Since the sample changes from draw to draw, the hyperplane parameters are random variables of non-zero variance
  - Instead of a single hyperplane we have a probability distribution over possible hyperplanes
  - The larger the margin, the larger the number of hyperplanes that will not originate errors on the data
  - The larger the value of γ, the larger the variance allowed on the plane parameter estimates!



Duality

- We must solve an optimization problem with constraints
- There is a rich theory on how to solve such problems
  - We will not get into it here (take 271B if interested)
  - The main result is that we can often formulate a dual problem which is easier to solve
  - In the dual formulation we introduce a vector of Lagrange multipliers $\alpha_i > 0$, one for each constraint, and solve

$$\max_{\alpha \geq 0} q(\alpha) = \max_{\alpha \geq 0} \left\{ \min_{\mathbf{w}} L(w, b, \alpha) \right\}$$

  - where

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \left[y_i \left(w^T x_i + b \right) - 1 \right]$$

is the Lagrangian

The Dual Optimization Problem

- For the SVM, the dual problem can be simplified into

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \right\}$$

$$\text{subject to } \sum_i y_i \alpha_i = 0$$

- Once this is solved, the vector

$$w^* = \sum_i \alpha_i y_i x_i$$

is the normal to the maximum margin hyperplane

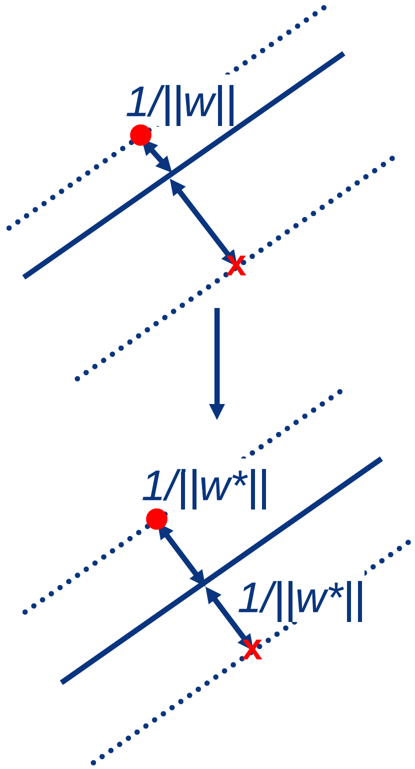
- Note: the dual solution does not determine the optimal *b*,* since *b* drops out when we solve

$$\min_{\mathbf{w}} L(w, b, \alpha)$$

The Dual Problem

- There are various possibilities for determining *b**.
  For example:
  - Pick one point x^+ on the margin on the $y = 1$ side and one point x^- on margin on the $y = -1$ side
  - Then use the margin constraint

$$\left.\begin{aligned} \mathbf{w}^T \mathbf{x}^+ + b &= 1 \\ \mathbf{w}^T \mathbf{x}^- + b &= -1 \end{aligned}\right\} \Leftrightarrow \boxed{b^* = -\frac{\mathbf{w}^T(\mathbf{x}^+ + \mathbf{x}^-)}{2}}$$



- Note:
  - The maximum margin solution guarantees that there is always at least one point “on the margin” on each side
  - If not, we could move the hyperplane and get an even larger margin (see figure on the right)

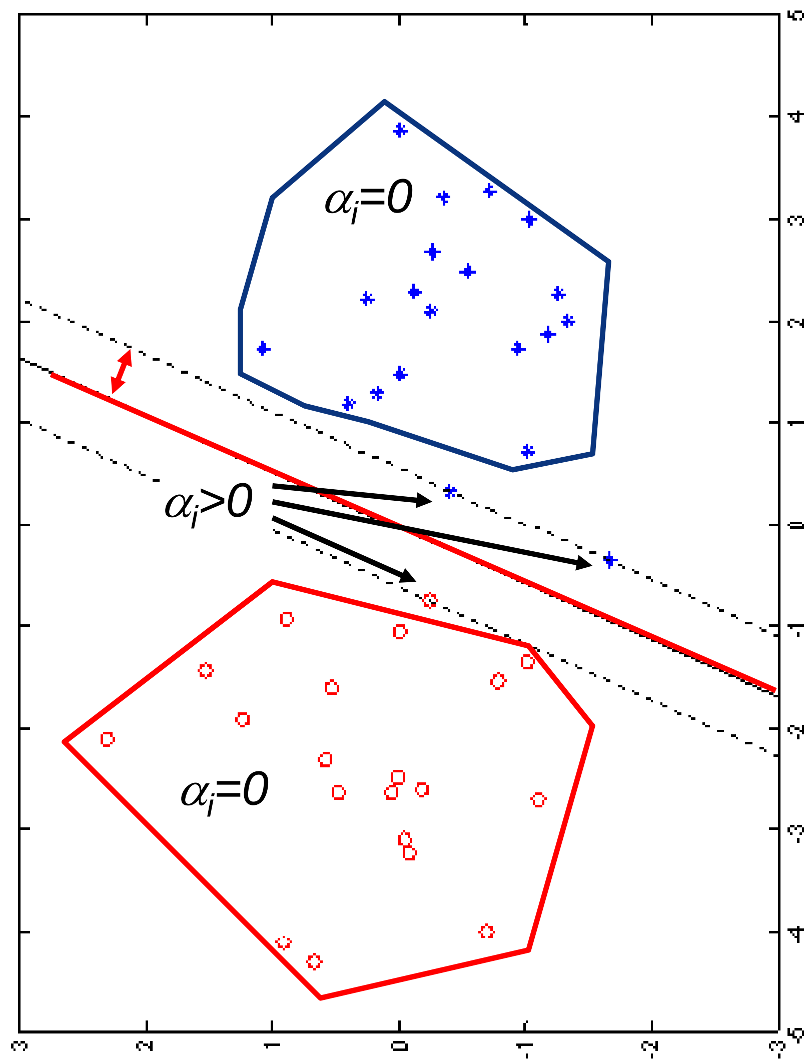
Support Vectors

It turns out that:

- An inactive constraint always has zero Lagrange multiplier α_i
- That is,
  - i) $\alpha_i > 0$ and $y_i(w^{*T}x_i + b^*) = 1$ or
  - ii) $\alpha_i = 0$ and $y_i(w^{*T}x_i + b^*) > 1$
- Hence $\alpha_i > 0$ only for points

$$|w^{*T}x_i + b^*| = 1$$

which are those that lie at a distance equal to the margin (i.e., those that are "on the margin"). These points are the "Support Vectors"



Support Vectors

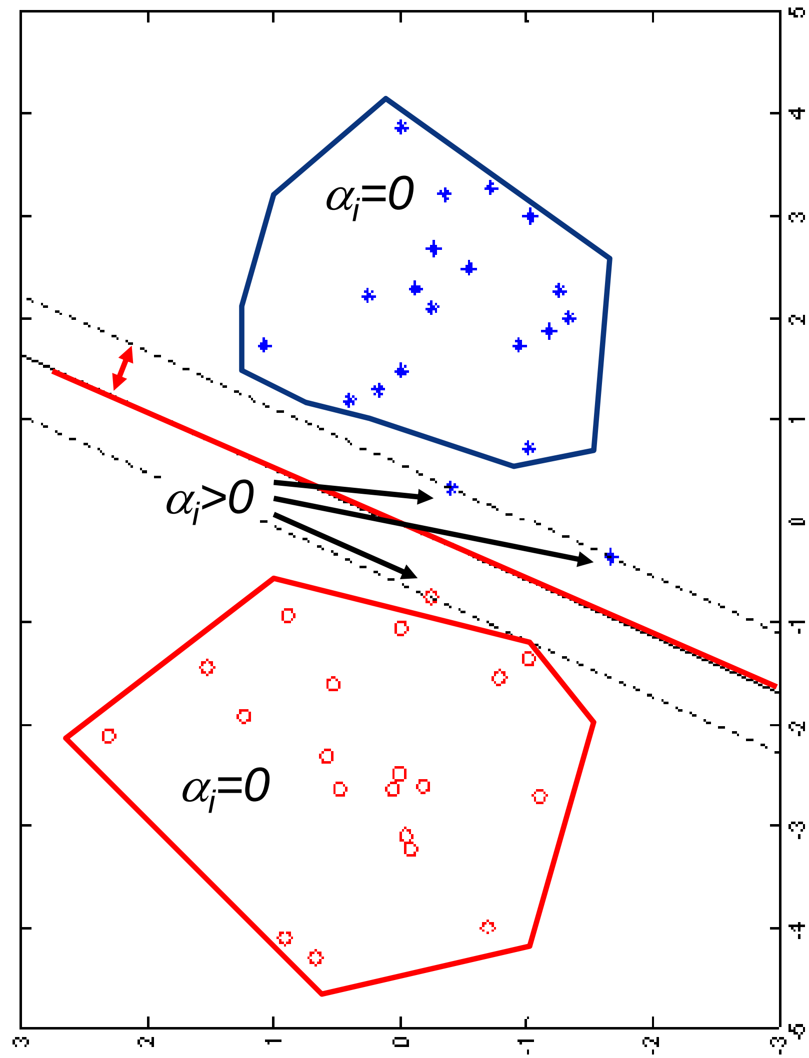
- The points with $\alpha_i > 0$ "support" the optimal hyperplane (w^*,b^*).
- This why they are called "Support Vectors"
- Note that the decision rule is

$$f(x) = \text{sgn}\left[w^{*T} x + b^* \right]$$

$$= \text{sgn}\left[\sum_i y_i \alpha_i^* x_i^T \left(x - \frac{x^+ + x^-}{2} \right) \right]$$

$$= \text{sgn}\left[\sum_{i \in SV} y_i \alpha_i^* x_i^T \left(x - \frac{x^+ + x^-}{2} \right) \right]$$

where SV = $\{i \mid \alpha^*_i > 0\}$ indexes the set of support vectors



Support Vectors and the SVM

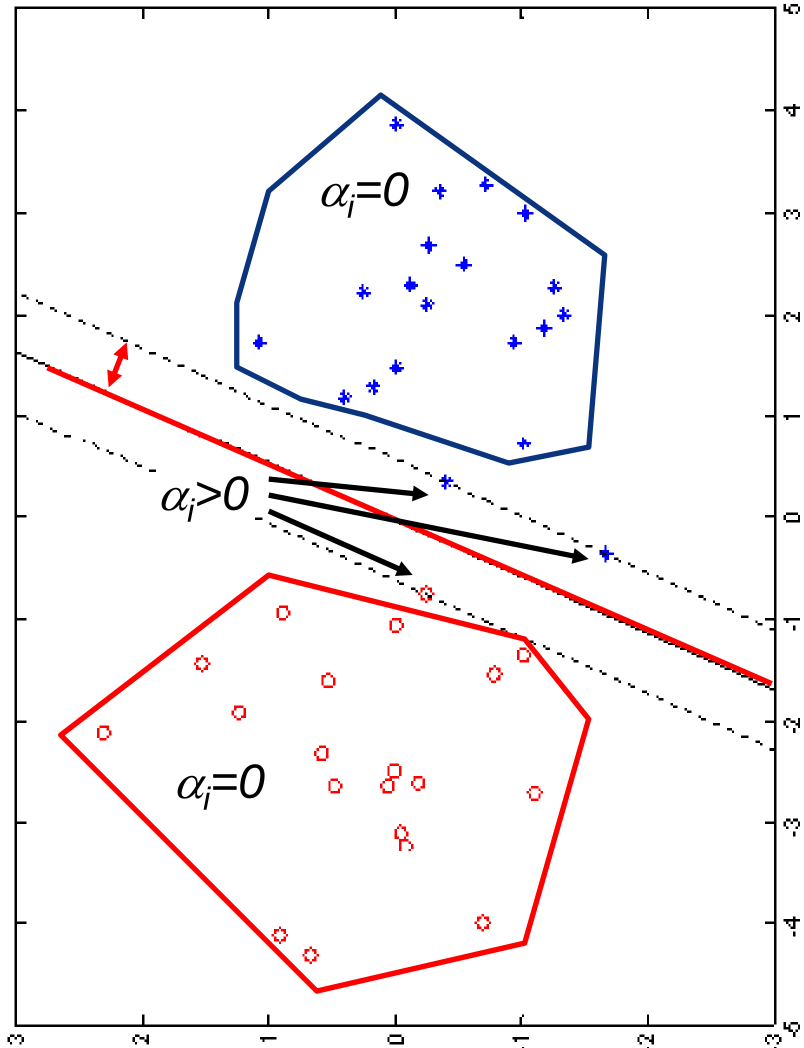
- Since the decision rule is

$$f(x) = \mathrm{sgn}\left[\sum_{i\,\in\,\mathrm{SV}} y_i \alpha_i^* x_i^T \left(x - \frac{x^+ + x^-}{2} \right)\right]$$

  where x^+ and x^- are support vectors, we see that we only need the support vectors to completely define the classifier!

- We can literally throw away all other points!!

- The Lagrange multipliers can also be seen as a measure of importance of each point



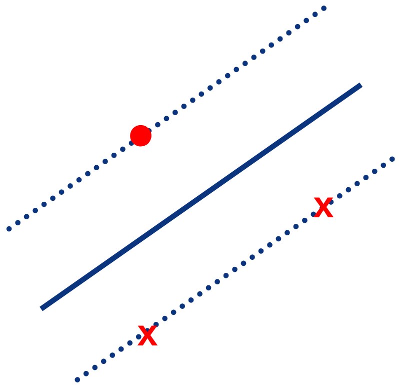
- Points with $\alpha_i = 0$ have no influence—a small perturbation does not change the solution

The Robustness of SVMs

- We talked a lot about the "curse of dimensionality"
  - In general, the number of examples required to achieve certain precision of pdf estimation, and pdf-based classification, is exponential in the number of dimensions
- It turns out that SVMs are remarkably robust to the dimensionality of the feature space
  - Not uncommon to see successful applications on 1,000D+ spaces
- Two main reasons for this:
  - 1) All that the SVM has to do is to learn a hyperplane.

    Although the number of dimensions may be large, the number of parameters is relatively small and there is not much room for overfitting

    In fact, *d+1* points are enough to specify the decision rule in R^d !!



Robustness: SVMs as Feature Selectors

- The second reason for robustness is that the data/feature space *effectively* is not *really* that large
  - 2) This is because the SVM is a feature selector

To see this let's look at the decision function

$$f(x) = \mathrm{sgn}\left[\sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^*\right]$$

This is a thresholding of the quantity

$$\sum_{i \in SV} y_i \alpha_i^* x_i^T x$$

Note that each of the terms $x_i^T x$ is the projection (actually, inner product) of the vector which we wish to classify, x, onto the training (support) vector x_i

SVMs as Feature Selectors

- Define *z* to be the vector of the projection of *x* onto all of the support vectors

$$z(x) = \left(x^T x_{i_1}, \cdots, x^T x_{i_k} \right)^T$$

- The decision function is a hyperplane in the *z*-space

$$f(x) = \operatorname{sgn}\left[\sum_{i \in \mathrm{SV}} y_i \alpha_i^* x_i^T x + b^* \right] = \operatorname{sgn}\left[\sum_k w_k^* z_k(x) + b^* \right]$$

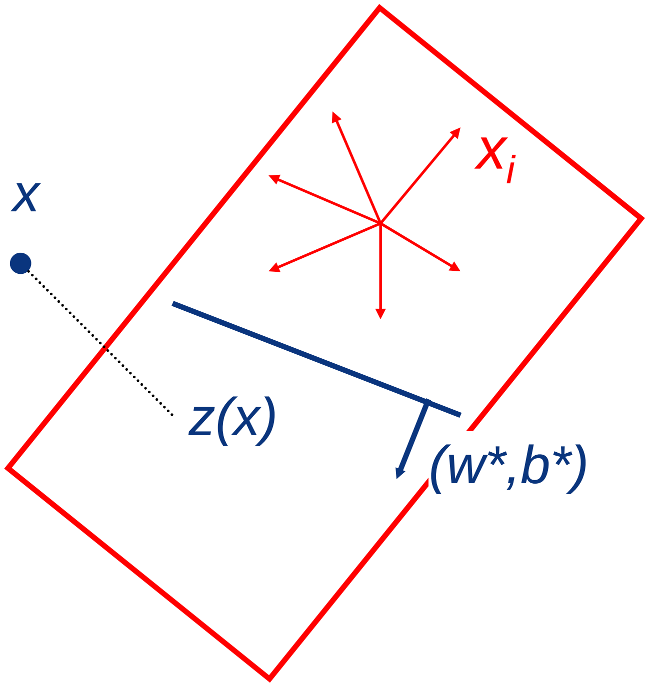
with

$$w^* = \left(\alpha_{i_1}^* y_{i_1}, \cdots, \alpha_{i_k}^* y_{i_k} \right)^T$$

- This means that
  - The classifier operates only on the span of the support vectors!
  - The SVM performs feature selection automatically.

SVMs as Feature Selectors

- Geometrically, we have:
  - 1) Projection of new data point x on the span of the support vectors
  - 2) Classification on this (sub)space



$$w^* = \left(\alpha_{i_1}^* y_{i_1}, \cdots, \alpha_{i_k}^* y_{i_k}\right)^T$$

- The effective dimension is |SV| and, typically, |SV| << n !!

Summary of the SVM

- SVM training:
  - 1) Solve the optimization problem:

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \right\}$$

$$\text{subject to} \quad \sum_i y_i \alpha_i = 0$$

  - 2) Then compute the parameters of the "large margin" linear discriminant function:

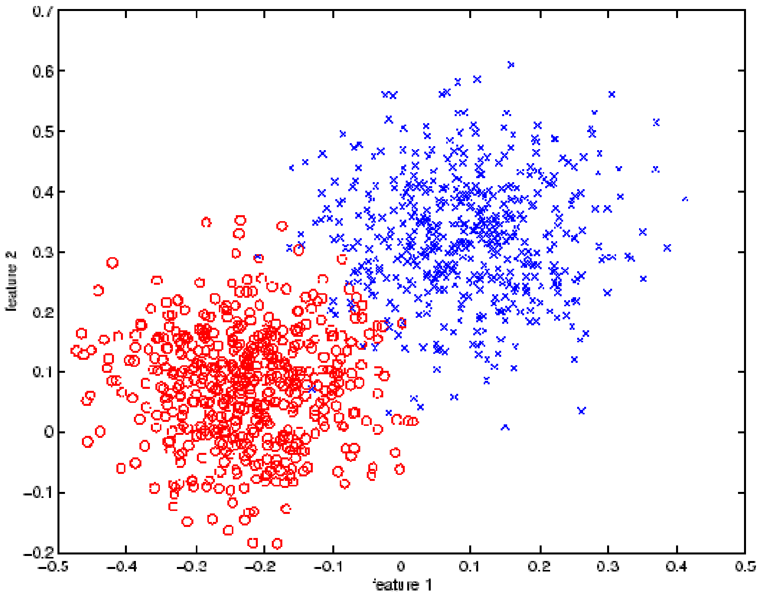
$$w^* = \sum_{i \in SV} \alpha_i^* y_i x_i \qquad b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* \left(x_i^T x^+ + x_i^T x^- \right)$$

- SVM Linear Discriminant Decision Function:

$$f(x) = \text{sgn}\left[\sum_{i \in SV} y_i \alpha_i^* x_i^T x + b^* \right]$$

Non-Separable Problems

- So far we have assumed linearly separable classes
- This is rarely the case in practice
- A separable problem is “easy” most classifiers will do well
- We need to be able to extend the SVM to the non-separable case

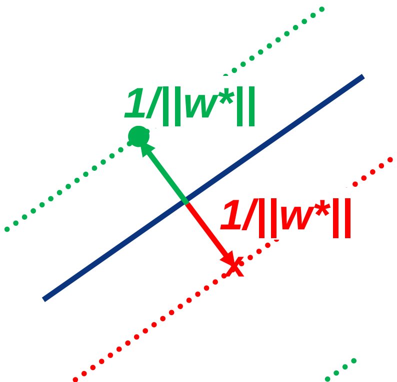


- Basic idea:
  - With class overlap we cannot enforce a (“hard”) margin.
  - But we can enforce a “soft margin”
  - For most points there *is* a margin. But there are a few outliers that cross-over, or are closer to the boundary than the margin. So how do we handle the latter set of points?

Soft Margin Optimization

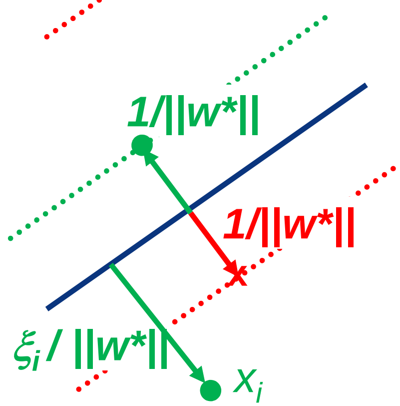
- Mathematically this is done by introducing slack variables
- Rather than solving the "hard margin" problem

$$\min_{w,b} \|w\|^2 \quad \text{subject to } y_i\left(w^T x_i + b\right) \geq 1 \quad \forall i$$



instead we solve the "soft margin" problem

$$\min_{w,\xi,b} \|w\|^2 \quad \text{subject to } y_i\left(w^T x_i + b\right) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0, \forall i$$

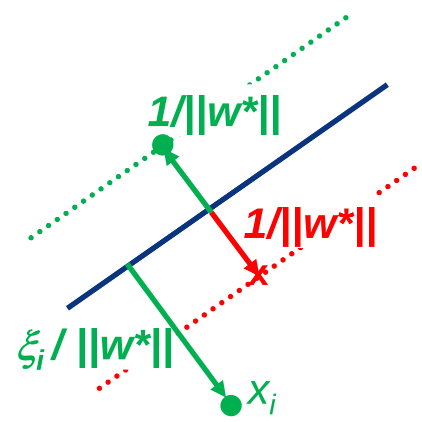


- The ξ_i are called slack variables
- Basically, the same optimization as before but points with $\xi_i > 0$ are allowed to violate the margin

Soft Margin Optimization

- Note that, as it stands, the problem is not well defined
- By making ξ_i arbitrarily large, $w \approx 0$ is a solution!
- Therefore, we need to penalize large values of ξ_i
- Thus, instead we solve the penalized, or regularized, optimization problem:

$$\min_{w,\xi,b} \ \|w\|^2 + C\sum_i \xi_i$$
$$\text{subject to} \quad y_i\left(w^T x_i + b\right) \ge 1 - \xi_i \quad \forall i$$
$$\xi_i \ge 0, \forall i$$



- The quantity $C\sum_i \xi_i$ is the penalty, or regularization, term. The positive parameter C controls how harsh it is.

The Soft Margin Dual Problem

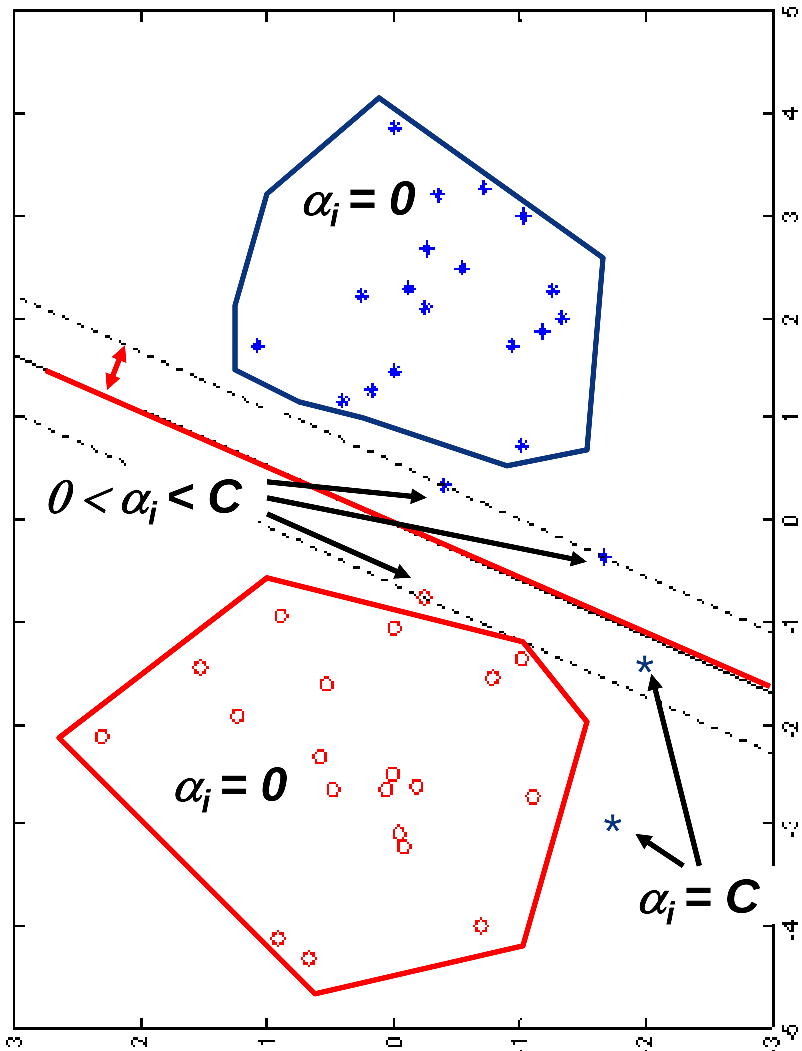
- The dual optimization problem:

$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i \alpha_i \right\}$$

$$\textbf{subject to} \quad \sum_i y_i \alpha_i = 0,$$

$$0 \leq \alpha_i \leq C$$

- The only difference with respect to the hard margin case is the “box constraint” on the Lagrange multipliers α_i
- Geometrically we have this



Support Vectors

- They are the points with $\alpha_i > 0$
- As before, the decision rule is

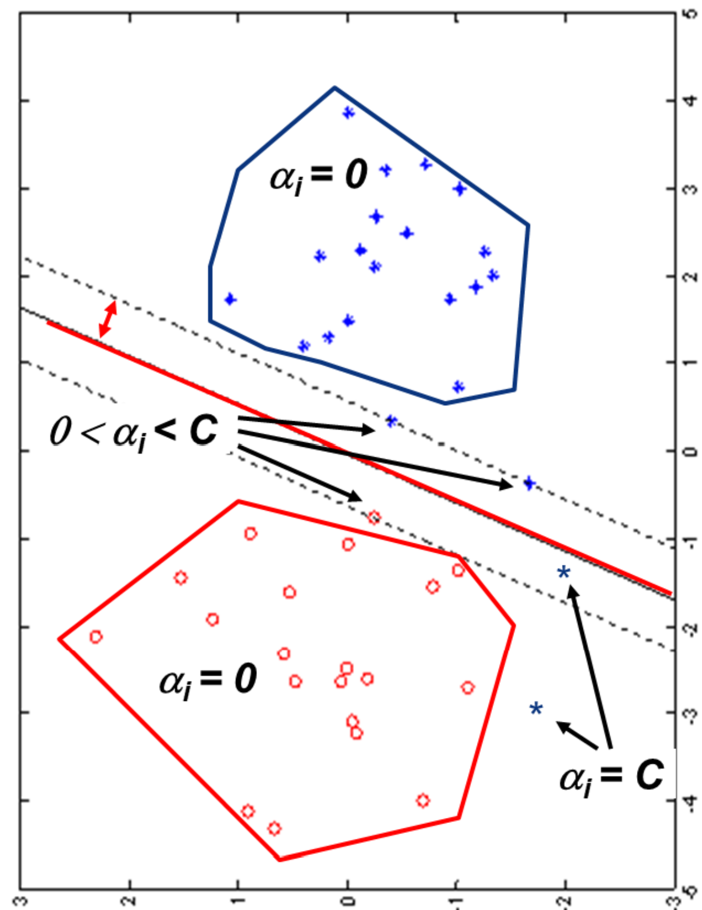
$$f(x) = \mathrm{sgn}\left[\sum_{i\,\in\,SV} y_i \alpha_i^* x_i^T x + b^*\right]$$

where SV = $\{ i \mid \alpha^*_i > 0 \}$

and b^* is chosen s.t.

  - $y_i g(x_i) = 1$, for all x_i s.t. $0 < \alpha_i < C$

- The box constraint on the Lagrange multipliers:
  - makes intuitive sense as it prevents any single support vector outlier from having an unduly large impact in the decision rule.



Kernelization of the SVM

- Note that all SVM equations depend only on $x_i^T x_j$
- The kernel trick is trivial: replace by $K(x_i, x_j)$
  - 1) Training:

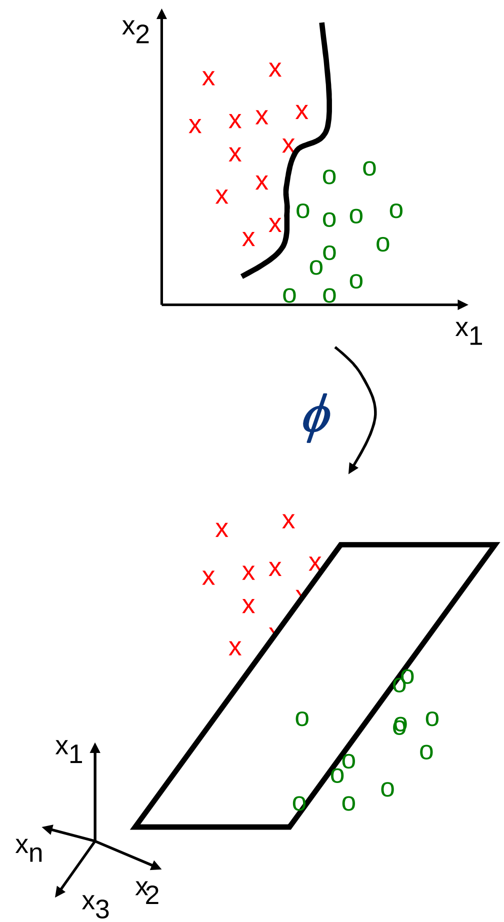
$$\max_{\alpha \geq 0} \left\{ -\frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j k\left(x_i, x_j\right) + \sum_{i} \alpha_i \right\}$$

$$\text{subject to} \quad \sum_{i} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C$$

$$b^* = -\frac{1}{2} \sum_{i \in SV} y_i \alpha_i^* \left(K\left(x_i, x^+\right) + K\left(x_i, x^-\right) \right)$$

  - 2) Decision function:

$$f(x) = \operatorname{sgn}\left[\sum_{i \in SV} y_i \alpha_i^* K\left(x_i, x\right) + b^* \right]$$



Kernelization of the SVM

Notes:

- As usual, nothing we did really requires us to be in R^d.
  - We could have simply used $<x_i,x_j>$ to denote for the inner product on a infinite dimensional space and all the equations would still hold
- The only difference is that we can no longer recover w^* explicitly without determining the feature transformation ϕ, since

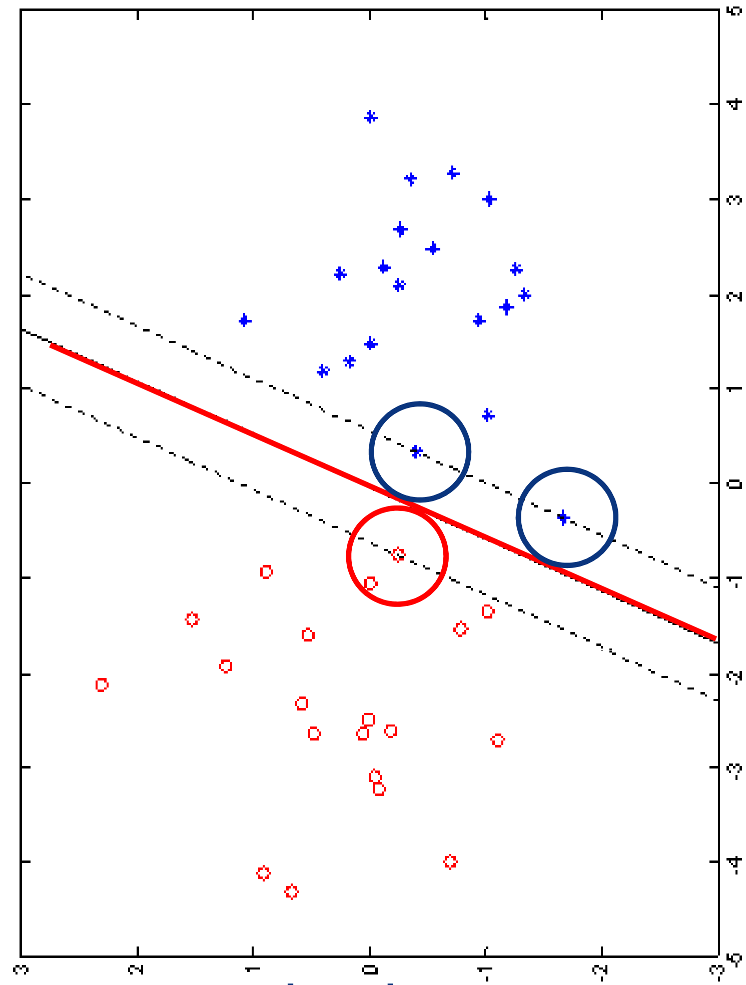
$$w^* = \sum_{i \in SV} \alpha_i^* y_i \, \phi(x_i)$$

- This can be an infinite dimensional object. E.g., it is a sum of Gaussians ("lives" in an infinite dimensional function space) when we use the Gaussian kernel
- Luckily, we don't need w^*, only the SVM decision function

$$f(x) = \operatorname{sgn}\left[\sum_{i \in SV} y_i \alpha_i^* K(x_i, x) + b^*\right]$$

Limitations of the SVM

- The SVM is appealing, but there are some limitations:
  - A major problem is the selection of an appropriate kernel. There is no generic “optimal” procedure to find the kernel or its parameters
    - Usually we pick an arbitrary kernel, e.g. Gaussian
    - Then, determine kernel parameters, e.g. variance, by trial and error
  - *C* controls the importance of outliers (larger *C* = less influence)
    - Not really intuitive how to choose *C*



- SVM is usually tuned and performance-tested using cross-validation. There is a need to cross-validate with respect to both *C* and kernel parameters

Practical Implementation of the SVM

- **In practice, we need an algorithm for solving the optimization problem of the training stage**
  - **This is a complex problem**
  - **There has been a large amount of research in this area**
    - **Therefore, writing "your own" algorithm is not going to be competitive**
  - **Luckily there are various packages available, e.g.:**
    - **libSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/**
    - **SVM light: http://www.cs.cornell.edu/People/tj/svm_light/**
    - **SVM fu: http://five-percent-nation.mit.edu/SvmFu/**
    - **various others (see http://www.support-vector.net/software.html)**
  - **There are also many papers and books on algorithms (see e.g. B. Schölkopf and A. Smola. Learning with Kernels. MIT Press, 2002)**

END