SUPPORT VECTOR MACHINES

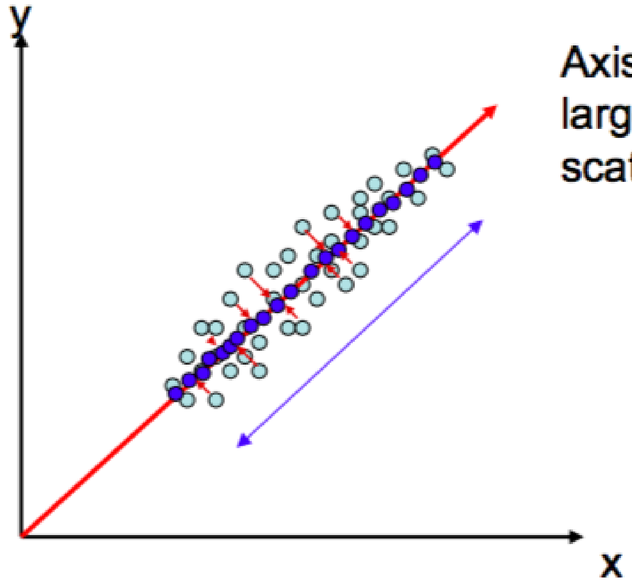
Many slides courtesy of Marios Savvides

Last time summary

- PCA
- LDA

What is PCA?

- We want to reduce the dimensionality but keep useful information
  - What is useful information? Variation
- We want to find a projection (a transformation) that describe maximum variation



Formulation

- Maximize the variance after projection ie
  - argmax $\mathrm{Var}(w^T x) = w^T \Sigma w$
- Subject to w is a unit vector
- Use Lagrangian multiplier to turn the constraint to a simple maximization
- $L(w, \lambda) = w^T \Sigma w - \lambda(w^T w - 1)$
- Take derivative with respect to w
- $\Sigma w = \lambda w$ <- eigenvector

Selecting eigenvectors

- Remember the variance of projected data is

$$\omega^T \Sigma\, \omega. \quad (1)$$

- And our solution yielded

$$\Sigma\omega = \lambda\omega \qquad (2)$$

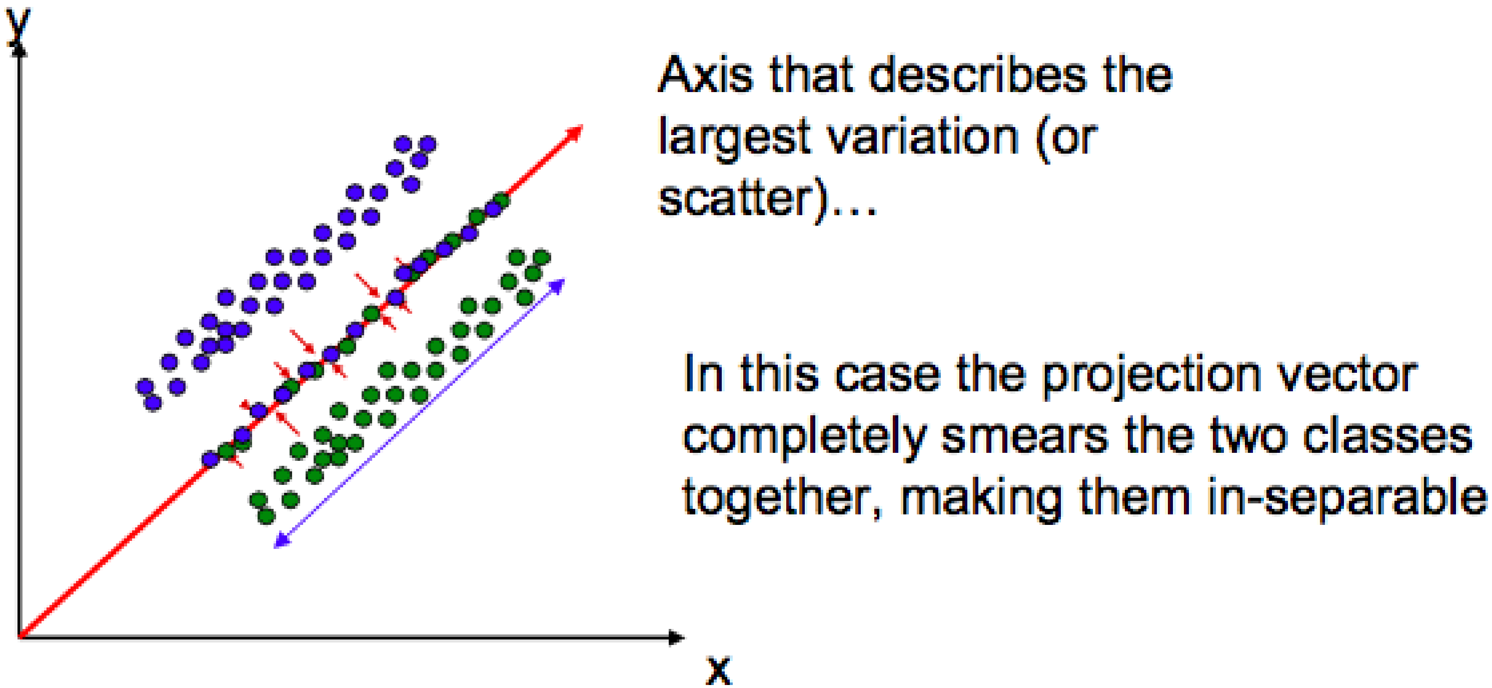
- Plug (2) in (1) and we get

$$\begin{aligned}\text{projected variance} &= \omega^T \Sigma\, \omega = \omega^T \lambda\omega \\ &= \lambda\, \omega^T \omega \qquad (\text{remember } \|\omega\|=1) \\ &= \lambda\end{aligned}$$

Eigenfaces

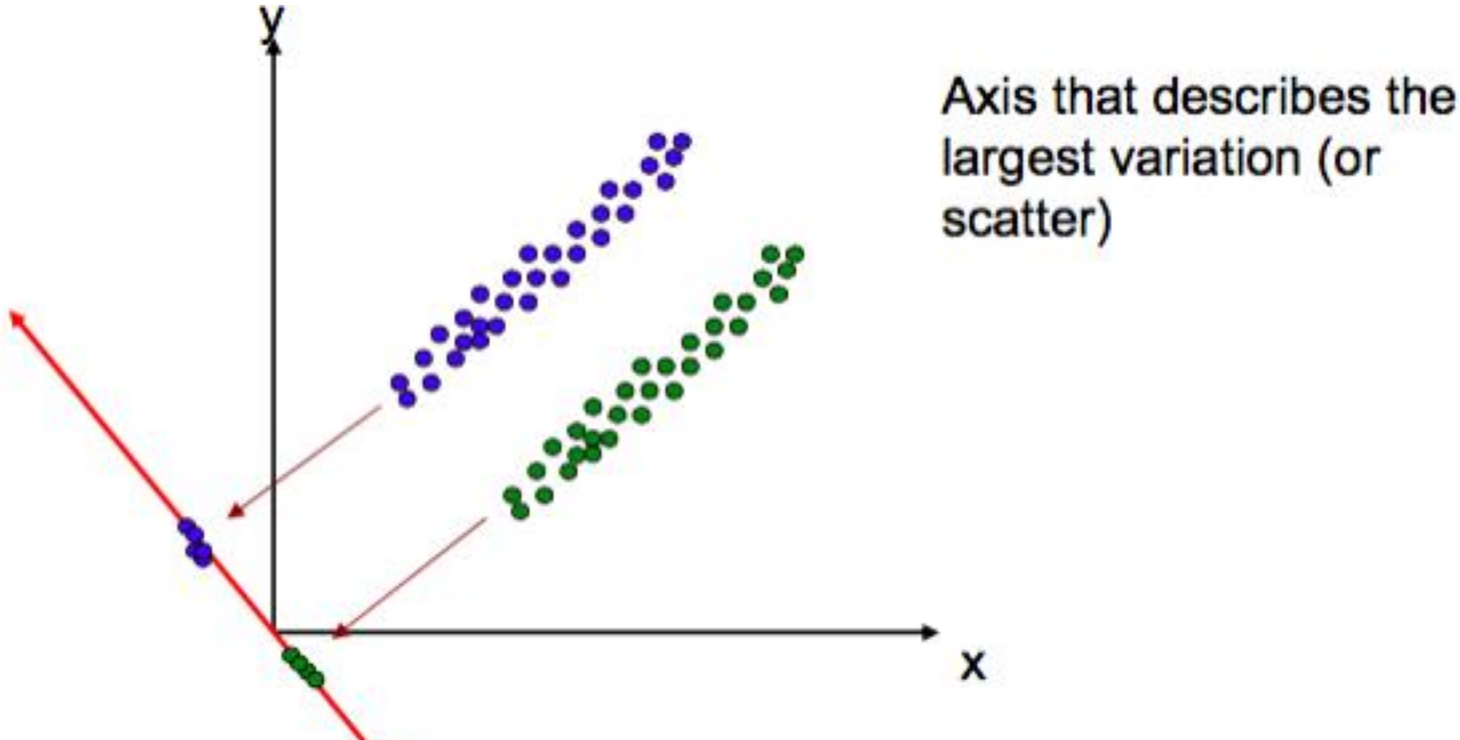
PCA for classification

- PCA does not cares about the class labels



What is LDA

- Find the projections that separate the classes.
- Assumes unimodal Gaussian model for each class
  - Maximize the distance between the means and minimize the variance of each class -> best classification performance



Fisher Linear Discriminant Criterion

- We want to maximize between class scatter
- We want to minimize within class scatter

- We have an objective function as a ratio so we can achieve both!

$$J(\mathbf{w}) = \frac{|(\tilde{\mu}_1 - \tilde{\mu}_2)|^2}{\tilde{s}_1{}^2 + \tilde{s}_2{}^2}$$

$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathbf{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathbf{W}}\mathbf{w}}$$

LDA solution

- If you do calculus

$$\mathbf{S_B w} = \lambda \mathbf{S_W w}$$

$$\mathbf{S_W}^{-1}\mathbf{S_B w} = \lambda \mathbf{w}$$

If S_w is non-singular and invertible.

- Generalized eigenvalue problem. The number of solutions is min(rankS_B, rankS_W) = C-1 or N-C
- For 2 class this simplifies to
- Note this is only one projection direction

$$\mathbf{w} = \mathbf{S_W}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

LDA+PCA

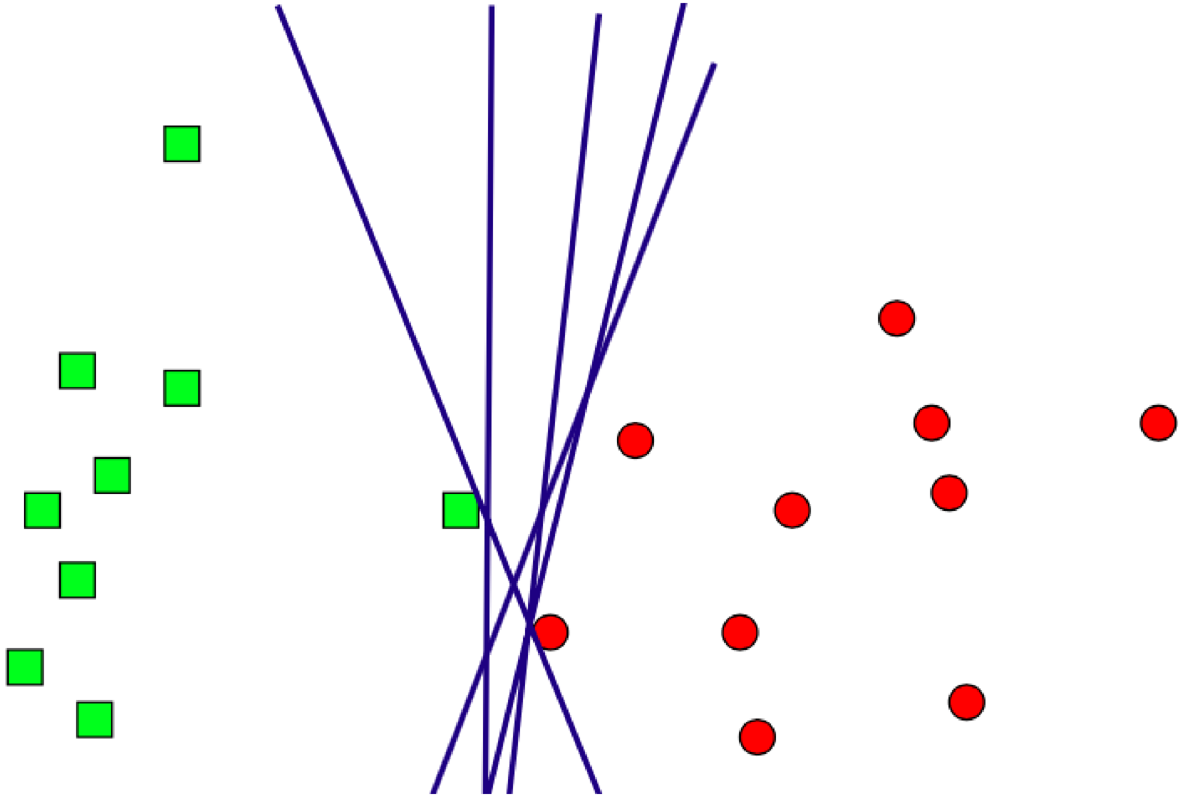
- First do PCA to reduce dimension
- Then do LDA to maximize classification ability
- How many dimensions to PCA?
  - Do PCA to keep N-C eigenvectors -> Makes S_w full rank and invertible
  - Then, do LDA and compute C-1 projections in this N-C subspace
- PCA+LDA = Fisher projection

SUPPORT VECTOR MACHINES

https://www.analyticsvidhya.com/blog/2020/10/the-mathematics-behind-svm/

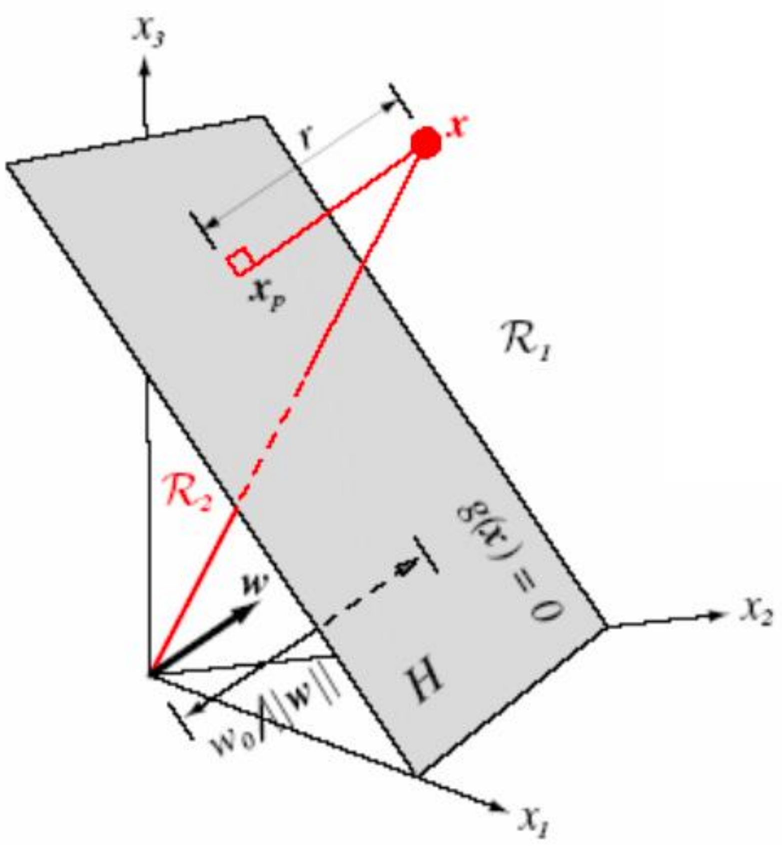
Linear classification problem

- Find a line that separates two classes
- Many solutions exist!
- Which one is the best?



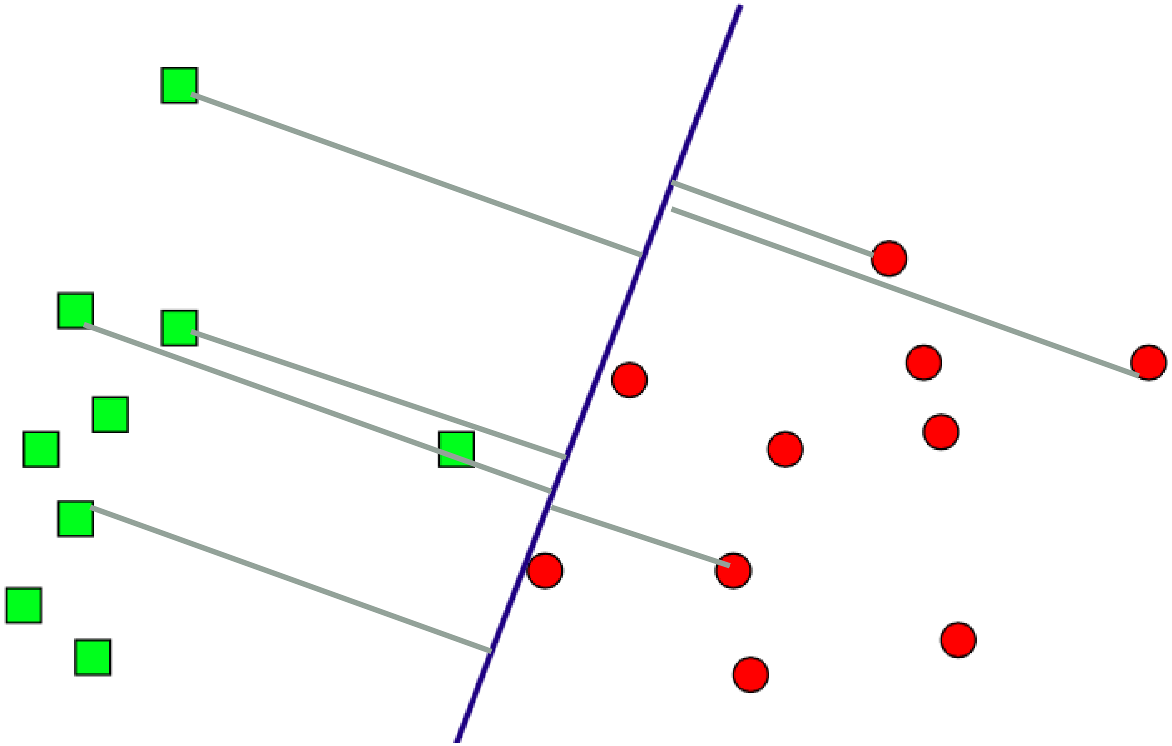
Geometric interpretation of a decision boundary

- Recall a linear classifier (without the logistic part)
  - $g(x) = \mathbf{w}^T\mathbf{x} + w_0$
- If $\mathbf{x}_1$ and $\mathbf{x}_2$ is on the decision boundary, then
  - $\mathbf{w}^T\mathbf{x}_1 + w_0 = \mathbf{w}^T\mathbf{x}_2 + w_0 = 0$
  - $w^T(\mathbf{x}_1\text{-}\mathbf{x}_2) = 0$
    - w is normal to any vector lying in the decision boundary
    - w is normal to the decision boundary hyperplane
    - If $w_0 = 0$, the hyperplane passes through the origin
- Note we can scale w and w_0 without affecting the hyperplane

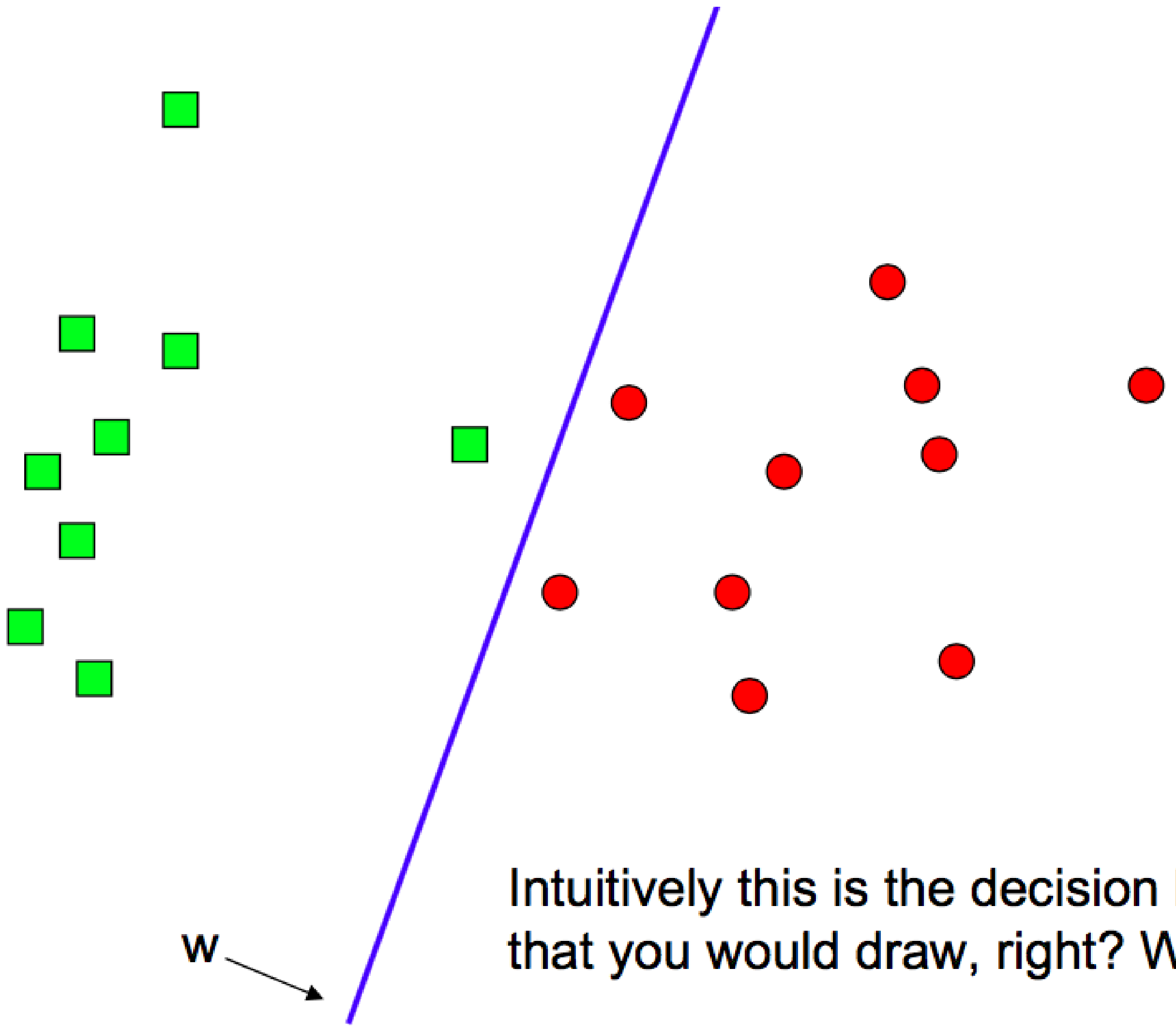


Logistic Regression

- Minimizes sum of L2 distance (square error) between all points to the line
- Also have probabilistic interpretation (assume noise is Gaussian)



Support vectors

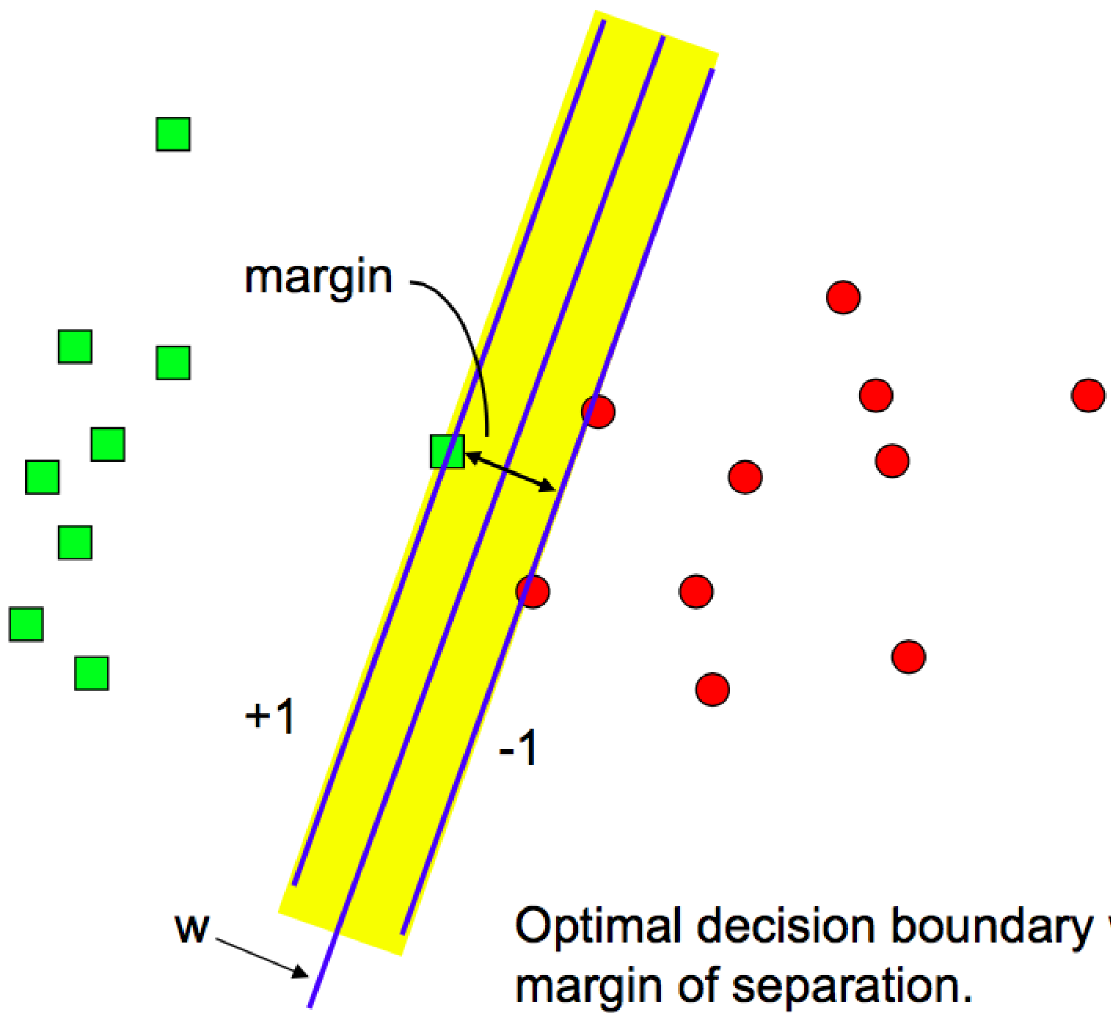


Intuitively this is the decision boundary that you would draw, right? Why?

Support Vector Machines (SVM)

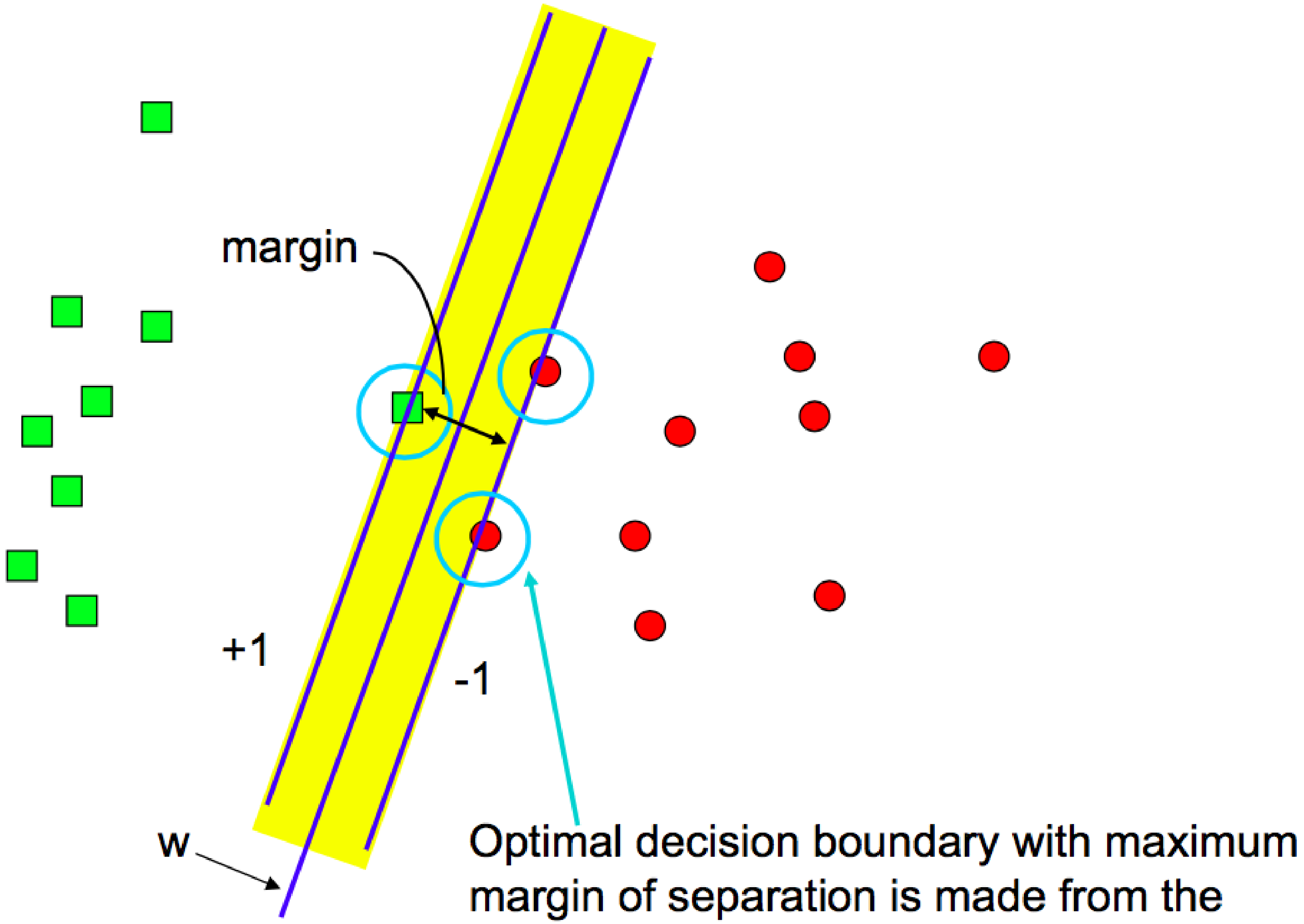
- Goal: improve generalization!
  - Care more about reducing classifier variance than reducing classifier bias
- How?
- Find the decision boundary that gives the most “slack” in classification
  - Don’t care about easy cases, care about borderline cases!
    - Focus on the margin
  - Maximize the “margin of error” between two classes

Support Vectors

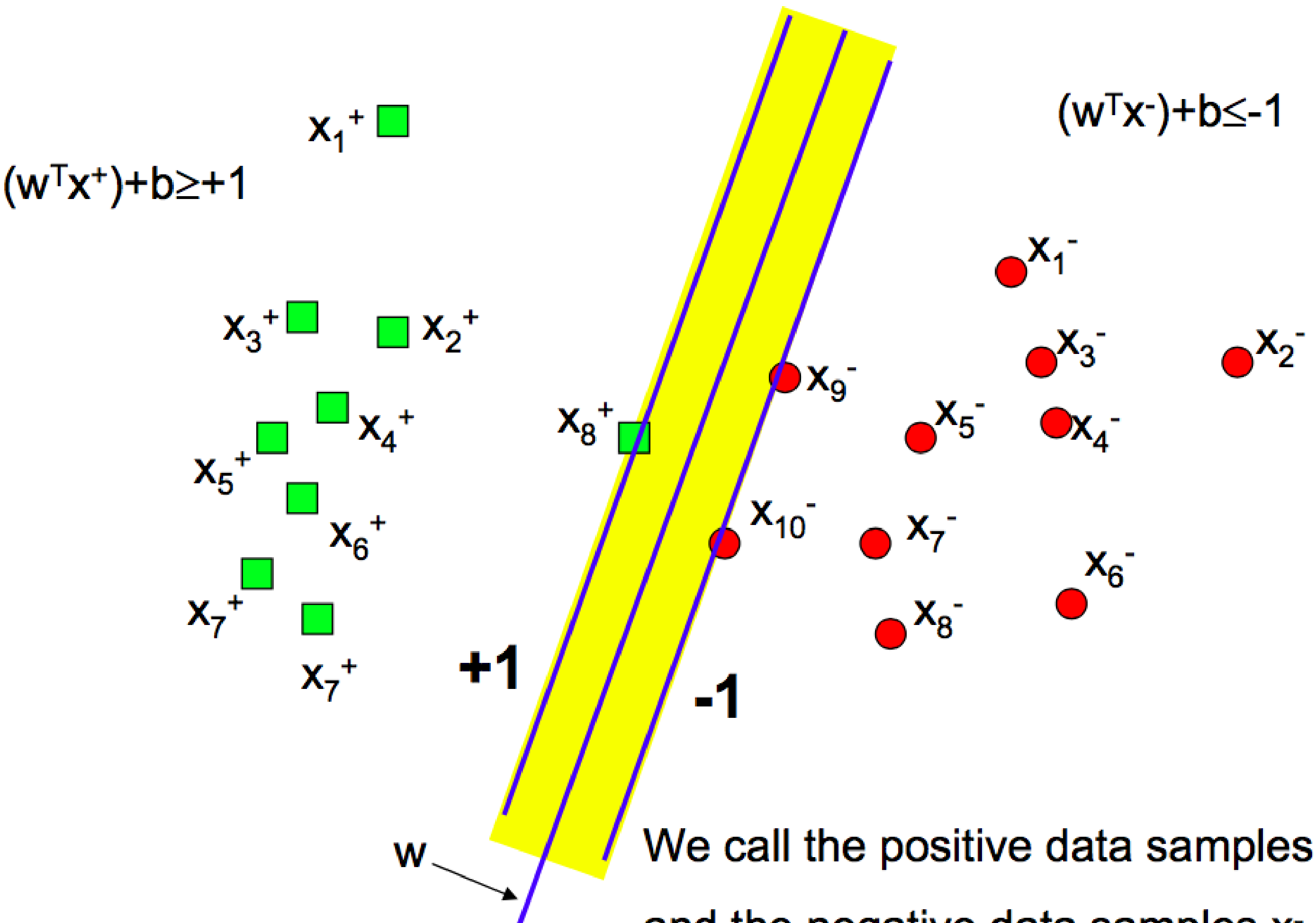


Optimal decision boundary with maximum margin of separation.

Support Vectors



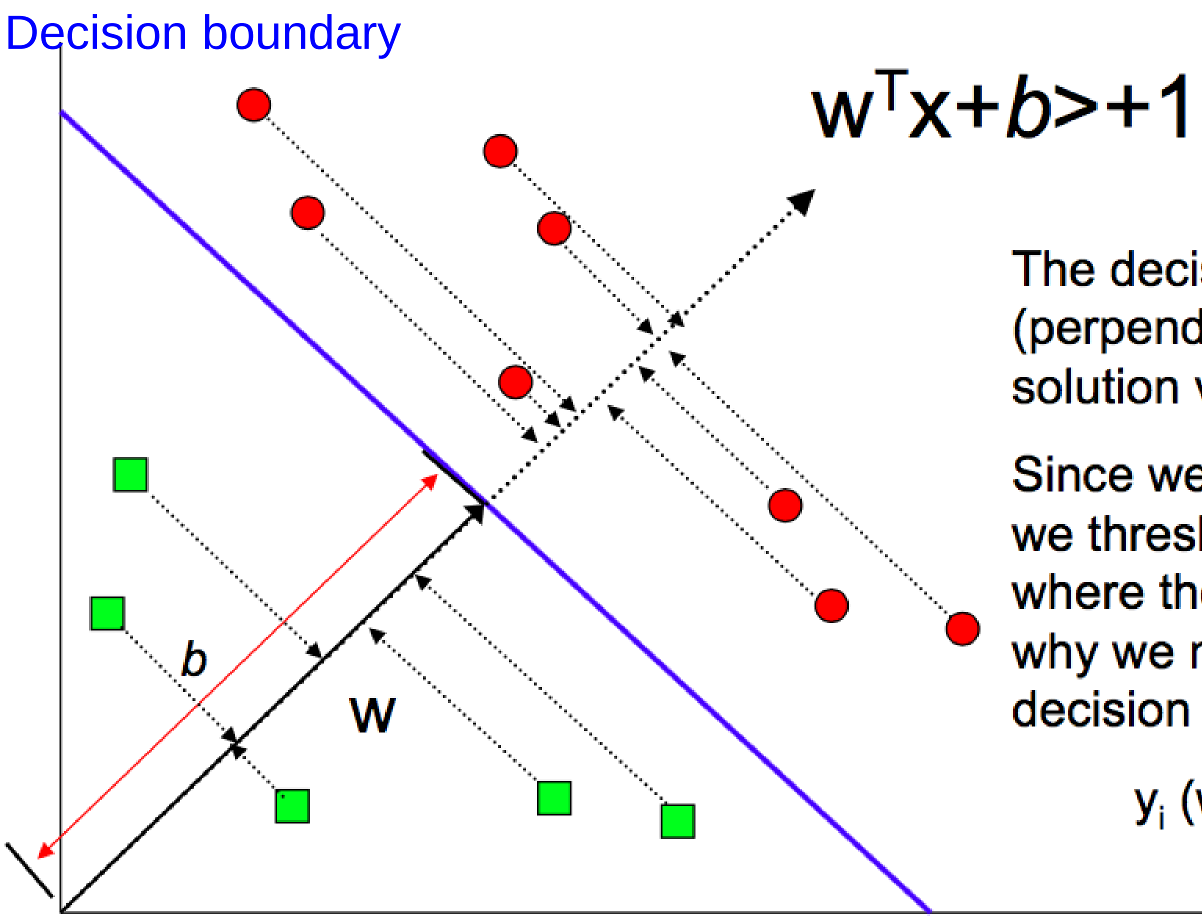
Support Vectors



We call the positive data samples x^+ i.e. $(w^T x^+)+b \geq +1$

and the negative data samples x^- i.e. $(w^T x^-)+b \leq -1$

Geometric interpretation

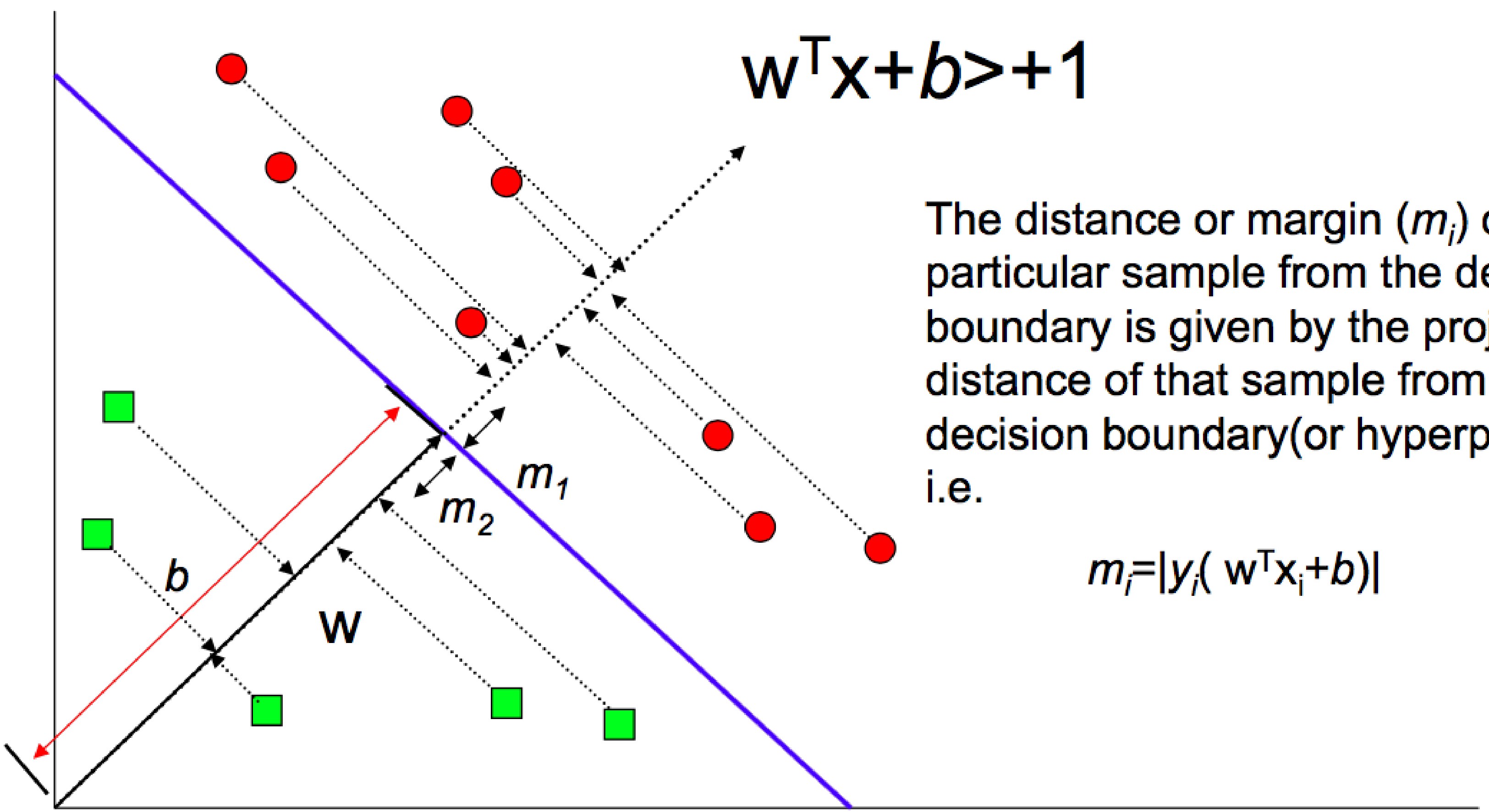


The decision boundary is orthogonal (perpendicular/normal) to the solution vector w.

Since we project samples onto w, we threshold the distance/position where these samples fall. That is why we need offset *b to* shift the decision boundary in the w direction.

$y_i\ (w^tx+b) \geq 0 \quad y_i=[-1, +1]$

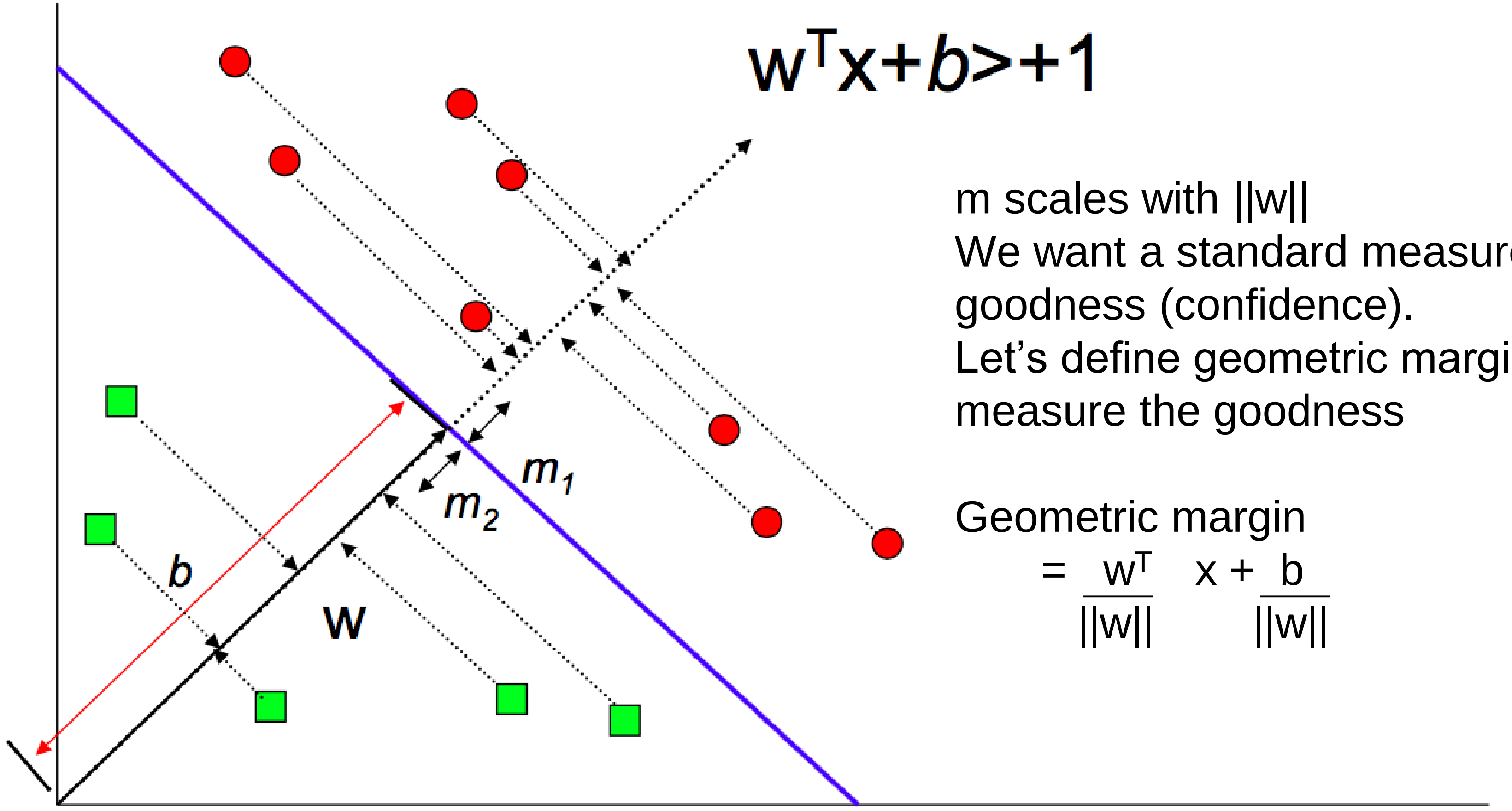
Functional Margin



The distance or margin (m_i) of a particular sample from the decision boundary is given by the projected distance of that sample from the decision boundary(or hyperplane) i.e.

$$m_i=|y_i(w^Tx_i+b)|$$

Geometric Margin

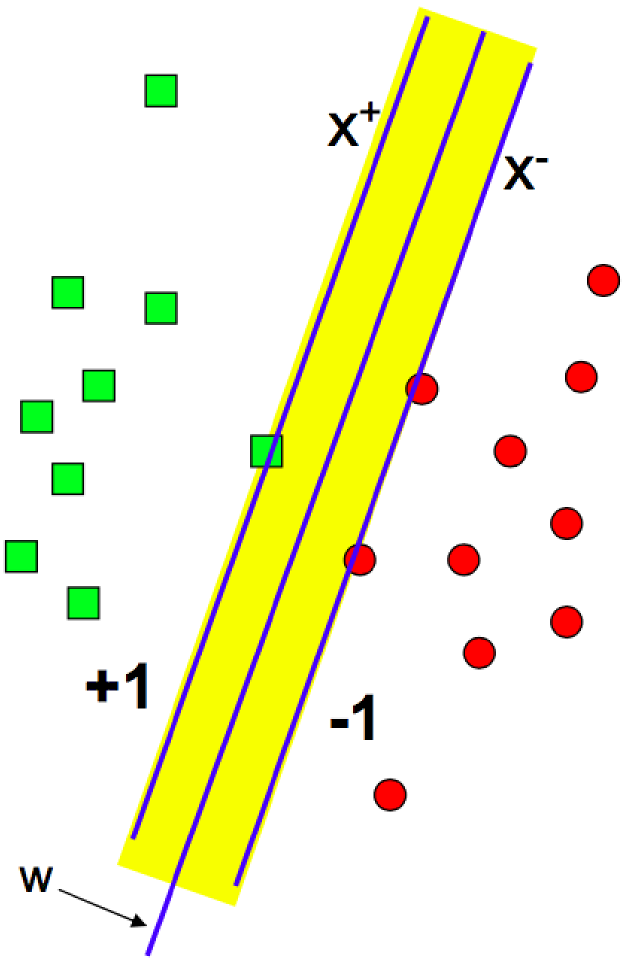


m scales with ||w||
We want a standard measure of goodness (confidence).
Let's define geometric margin to measure the goodness

Geometric margin

$$= \frac{w^T}{||w||} x + \frac{b}{||w||}$$

Support Vectors



Let x^+ denote a positive point with functional margin of 1 and x^- denote a negative point respectively.

This implies:

$w^T x^+ + b = +1$

$w^T x^- + b = -1$

The difference between the two geometric margins is

$$m = \left(\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}^+ \rangle - \langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, \mathbf{x}^- \rangle \right)$$

$$= \frac{1}{\|\mathbf{w}\|} \left(\langle \mathbf{w}, \mathbf{x}^+ \rangle - \langle \mathbf{w}, \mathbf{x}^- \rangle \right)$$

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

< > denotes dot product

Max margin

- We want to maximize the margin
  - Maximize $\dfrac{2}{\|\mathbf{w}\|}$
- Same as minimize $\langle \mathbf{w}, \mathbf{w} \rangle = \mathbf{w}^{\mathrm{T}}\mathbf{w}$

SVM objective function

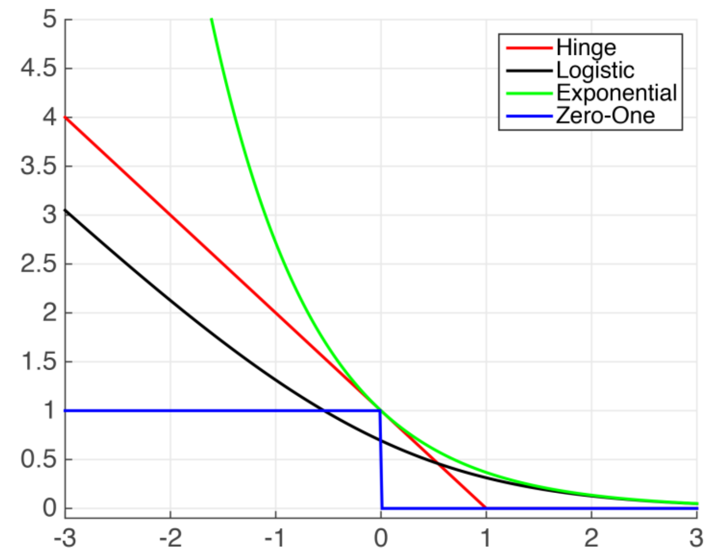
- Minimize $\mathbf{w}^T\mathbf{w}$
- Subject to
  - $y_i(<\mathbf{w}, \mathbf{x}_i> + b) = y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

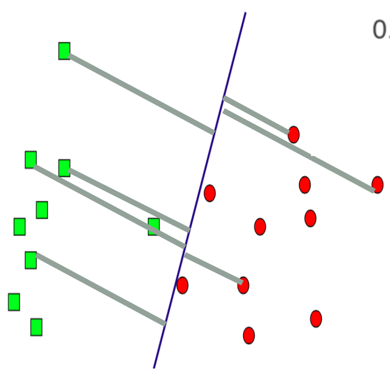
- $y_i = \{+1,-1\}$ depending on the binary class
  - Positive class must fall on the positive side of the boundary
  - Negative class must fall on the negative size

- Convex optimization (No local minimas)
- Can be solved by Quadratic Programing (QP)

Notes on the Losses

- Logistic regression optimizes for the log loss (binary cross entropy loss) https://www.analyticsvidhya.com/blog/2020/11/binary-cross-entropy-aka-log-loss-the-cost-function-used-in-logistic-regression
- SVM optimize for the hinge loss
  - $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$
  - Or $0 \geq 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)$
  - We don't want this inequality to be broken so our effective loss is
    - $\max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$

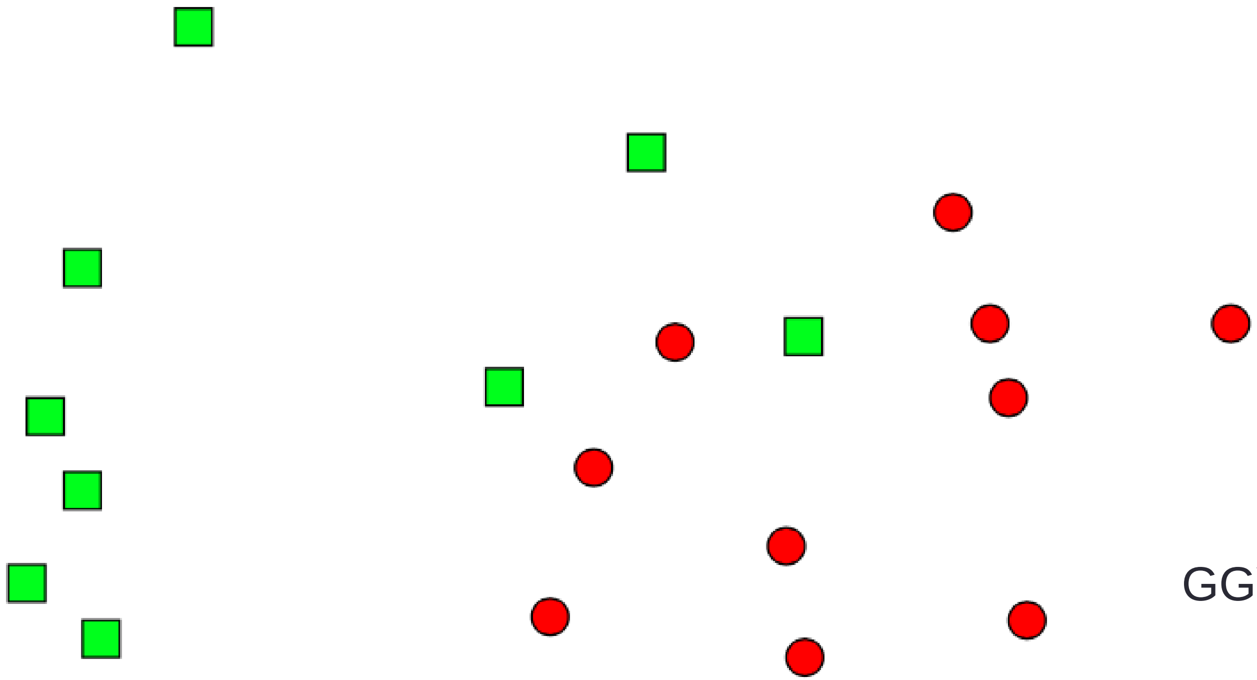




https://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote10.html

Linearly non-separable

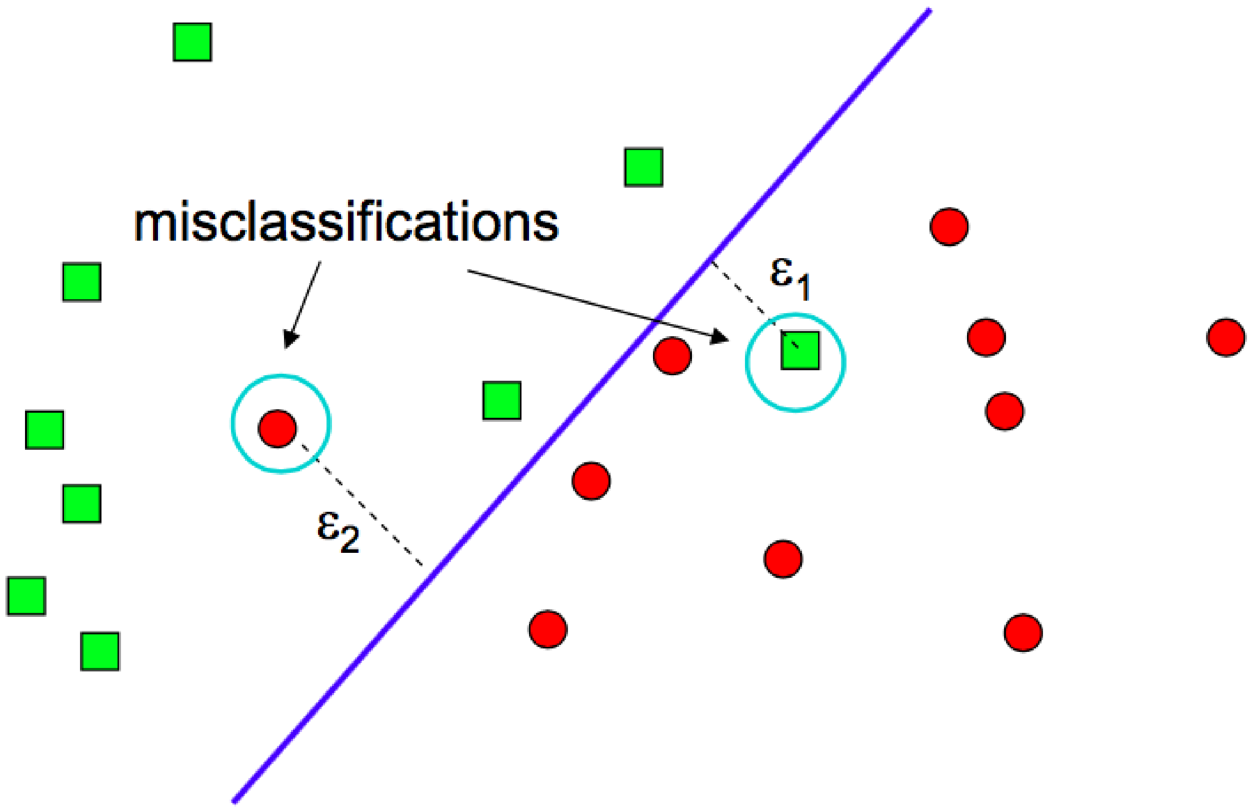
- What happens when you cannot separate the two classes with a linear boundary



GGWP?

Introducing an error term ε

- Aim for a hyperplane that tries to maximize the margin while minimize total error $\Sigma\varepsilon_i$



Slack variables

- We call these error terms “Slack variables”
- Give SVM some slack so that the SVM can do its job.

SVM objective function

- Minimize $\mathbf{w}^T\mathbf{w}$
- Subject to
  - $y_i(<\mathbf{w}, \mathbf{x}_i> + b) = y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- $y_i = \{+1,-1\}$ depending on the binary class
  - Positive class must fall on the positive side of the boundary
  - Negative class must fall on the negative size

SVM objective with slack

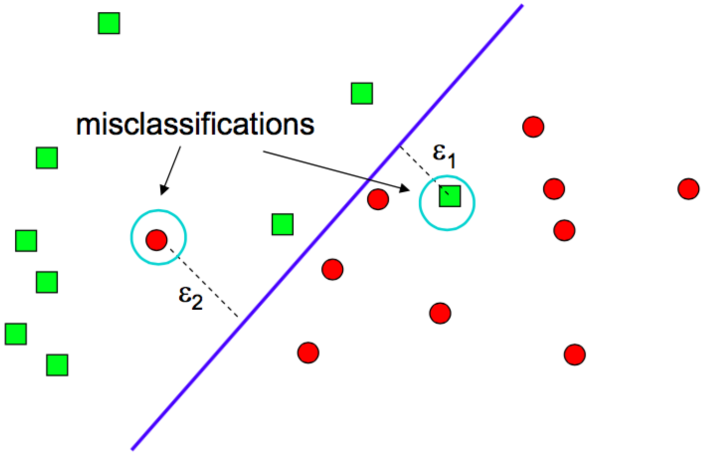
- Minimize $\mathbf{w}^T\mathbf{w} + C\Sigma\varepsilon_i$
- Subject to

C is a weight parameter, how much we care about slack

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq 1 - \varepsilon_i \qquad \textit{for} \quad +ve \qquad \textit{class}$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \leq -1 + \varepsilon_i \qquad \textit{for} \quad -ve \qquad \textit{class}$$

$$\varepsilon_i > 0 \qquad \forall i$$



Notes about slacks

- Even if the problem has linear separability we might want some slack still.
  - Miss labeled points near the boundaries, noise in the data set etc.
  - In this case, we trade-off classifier bias for classifier variance.

  - A form of regularization!
  - What is regularization?

Regularization in one slide

- What?
  - Regularization is a method to lower the model variance (and thereby increasing the model bias)
- Why?
  - Gives more generalizability (lower variance)
  - Better for lower amounts of data (reduce overfitting)
- How?
  - Introducing regularizing terms in the original loss function
    - Can be anything
      - $\mathbf{w}^T\mathbf{w} + C\Sigma\varepsilon_i$
      - MAP estimate is MLE with regularization (the prior term)

Maximum A Posteriori (MAP) Estimate

MLE

- Maximizing the likelihood (probability of data given model parameters)

$$\underset{\theta}{\operatorname{argmax}}\ p(\mathbf{x}|\theta)$$

$p(\mathbf{x}|\theta)$
$= L(\theta)$

- Usually done on log likelihood

- Take the partial derivative wrt to θ and solve for the θ that maximizes the likelihood

MAP

- Maximizing the posterior (model parameters given data)

$$\underset{\theta}{\operatorname{argmax}}\ p(\theta|\mathbf{x})$$

- But we don't know $p(\theta|\mathbf{x})$

- Use Bayes rule

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

- Taking the argmax for θ we can ignore $p(\mathbf{x})$

- $\underset{\theta}{\operatorname{argmax}}\ p(\mathbf{x}|\theta)\ p(\theta)$

Famous types of regularization

- L1 regularization: Regularizing term is a sum

$$\frac{1}{2}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2 \ + \Sigma_j \mid \theta_j \mid$$

- L2 regularization: Regularizing term is a sum of squares

$$\frac{1}{2}\Sigma_{i=1}^{m}(y_i - \theta^T \mathbf{x_i})^2 \ + \Sigma_j \mid \theta_j \mid^2$$

(Not to be confused with L1 and L2 losses)

http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/

Primal form – Dual form

- In optimization, many problems can be framed in two ways
  - Original version: Primal form
  - Transformed version: Dual form
- Both yield the same solution (under some conditions), but sometimes solving one method is a lot easier than the other.

SVM objective function – Primal form

- Minimize $\mathbf{w}^T\mathbf{w}$
- Subject to
  - $y_i(<\mathbf{w}, \mathbf{x}_i> + b) = y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- $y_i = \{+1,-1\}$ depending on the binary class
  - Positive class must fall on the positive side of the boundary
  - Negative class must fall on the negative size

Primal Lagrangian Form

$$L(\mathbf{w}, b, \alpha) = \tfrac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{w} - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) - 1 \right]$$

margin constraints

- Where α_i are the lagrange multipliers $\alpha_i \geq 0$
- We want to optimize this function

Primal form **w**

- Primal form:

$$L(\mathbf{w},b,\alpha)=\tfrac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}-\sum_{i=1}^{N}\alpha_i\left[y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-1\right]$$

- Differentiate with respect w to find

$$\frac{L(\mathbf{w},b,\alpha)}{\partial\mathbf{w}}=\mathbf{w}-\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i=\mathbf{0}$$

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

Primal form b

- Primal form:

$$L(\mathbf{w},b,\alpha)=\tfrac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}-\sum_{i=1}^{N}\alpha_i\left[y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-1\right]$$

- Differentiate with respect b to find

$$\frac{L(\mathbf{w},b,\alpha)}{\partial b}=\sum_{i=1}^{N}\alpha_i y_i=0$$

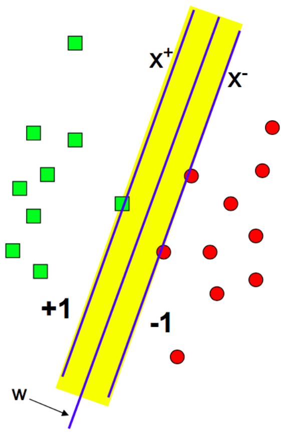
$$\sum_{i=1}^{N}\alpha_i y_i=0$$

Primal form solution

- w is a linear combination of our training data

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \qquad \sum_{i=1}^{N} \alpha_i y_i = 0 \qquad \alpha_i \geq 0$$

- Which training data depends on whether that training data is a support vector (the vector on the boundary) or not



α_i will be 0 for non support vectors

Dual form

- Primal form:

$$L(\mathbf{w},b,\alpha)=\tfrac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}-\sum_{i=1}^{N}\alpha_i\left[y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-1\right]$$

- Substitute $\mathbf{w}=\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i$ back into above Eq.

$$L(\mathbf{w},b,\alpha)=\tfrac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}-\sum_{i=1}^{N}\alpha_i\left[y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-1\right]$$

$$=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_i y_j\alpha_i\alpha_j\left\langle x_i,x_j\right\rangle-\sum_{i=1}^{N}\sum_{j=1}^{N}y_i y_j\alpha_i\alpha_j\left\langle x_i,x_j\right\rangle+\sum_{i=1}^{N}\alpha_i$$

$$\boxed{=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}y_i y_j\alpha_i\alpha_j\left\langle x_i,x_j\right\rangle}$$

Dual form

Dual form optimization

- Dual form:

$$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

- Subject to the constraints $\sum_{i=1}^{N}\alpha_i y_i = \mathbf{0}$ and $\alpha_i \geq 0$

- Again this is solvable with QP.

What does the dual form give us?

- Dual form

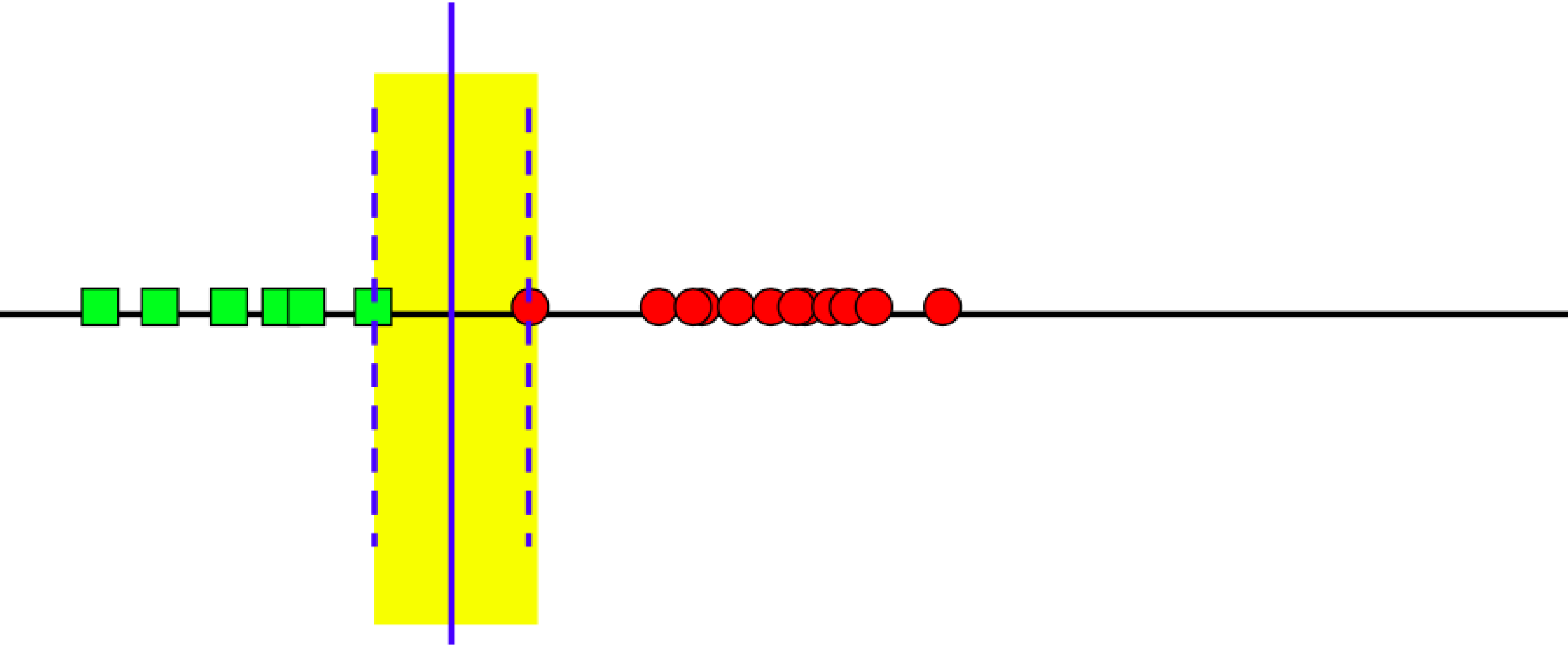
$$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

- Optimize using pairwise inner product of inputs instead of inputs
- Gram matrix (matrix of inner product between inputs)

- How is this useful?

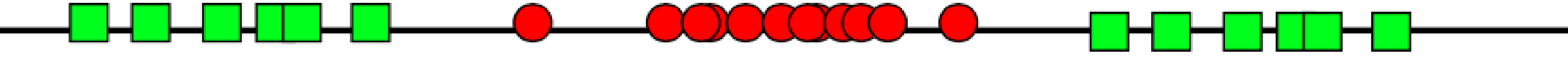
Example SVMs

- Easy



Example SVMs

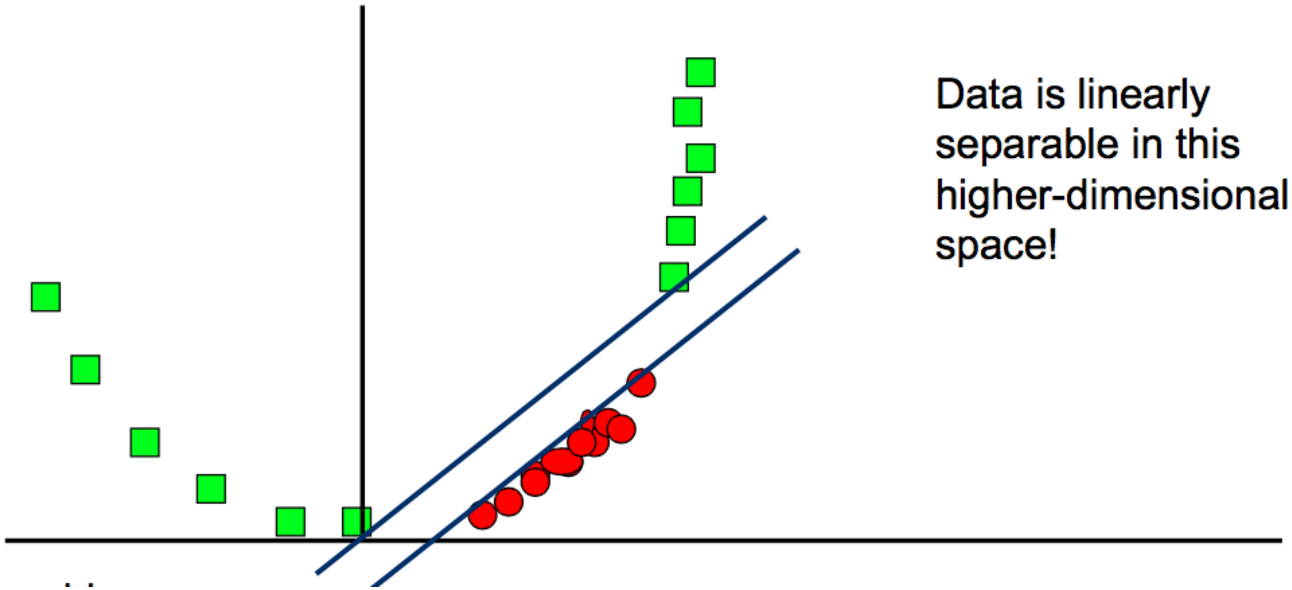
- ??????



Adding features (non-linear transformation)

- Remember we add non-linear features to linear regression to do non-linear fitting
- Consider as a non-linear transformation to higher dimensional space

$F(x) \to (x, x^2)$



Data is linearly separable in this higher-dimensional space!

What about curse of dimensionality?

- Didn’t we say higher dimension sucks?

- In this case our data is NOT separable in the original space, so we want to map to higher dimensions
  - Just high enough so the data is separable
- However, this will require higher compute because of dimensionality
  - Dual form will help with this!

Mapping functions

$$\phi : X \to F$$

- A mapping function that maps to higher dimensional space
- Our solutions become

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$$

- And if we want to classify a new sample

$$\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle$$

Mapping function dual form

- In the dual form we solve

$$\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \Phi(x_i), \Phi(x_j) \right\rangle$$

Inner product of the higher space

- Claim: sometimes inner product of the higher space can be solved directly without mapping to the higher space and compute the inner product

Kernel function

- We define the inner product in the mapped space as a kernel function K(x,y)

$$K(\mathbf{x},\mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}\ \ )\ \ \rangle$$

- Kernel of x -> (x, x^2)
  - $xy + x^2y^2$
- Kernel of x -> (x, x^2,x^3)
  - $xy + x^2y^2+x^3y$

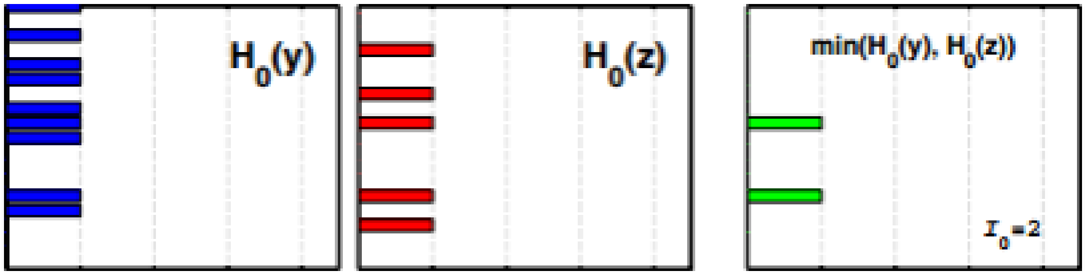
Kernel functions

- Sometimes we don’t even know what the mapping function is, but we “dream up” a kernel
  - A kernel is legitimate if it satisfies “Mercer’s Condition”
  - Mercer’s condition guarantees existence of a higher dimensional space that yields the dot product, but we just don’t know what space

Histogram intersection kernels

- Given input features which are histograms
  - Histogram of first data $H_0(y)$. Histogram of second data $H_1(z)$
- The Kernel that counts the intersection of the histograms is a valid kernel.
  - E.g. Sum of $\min(H_0(y), H_1(z))$ for all histogram bins

- (One of the most used kernels in computer vision)



Radial Basis Kernels

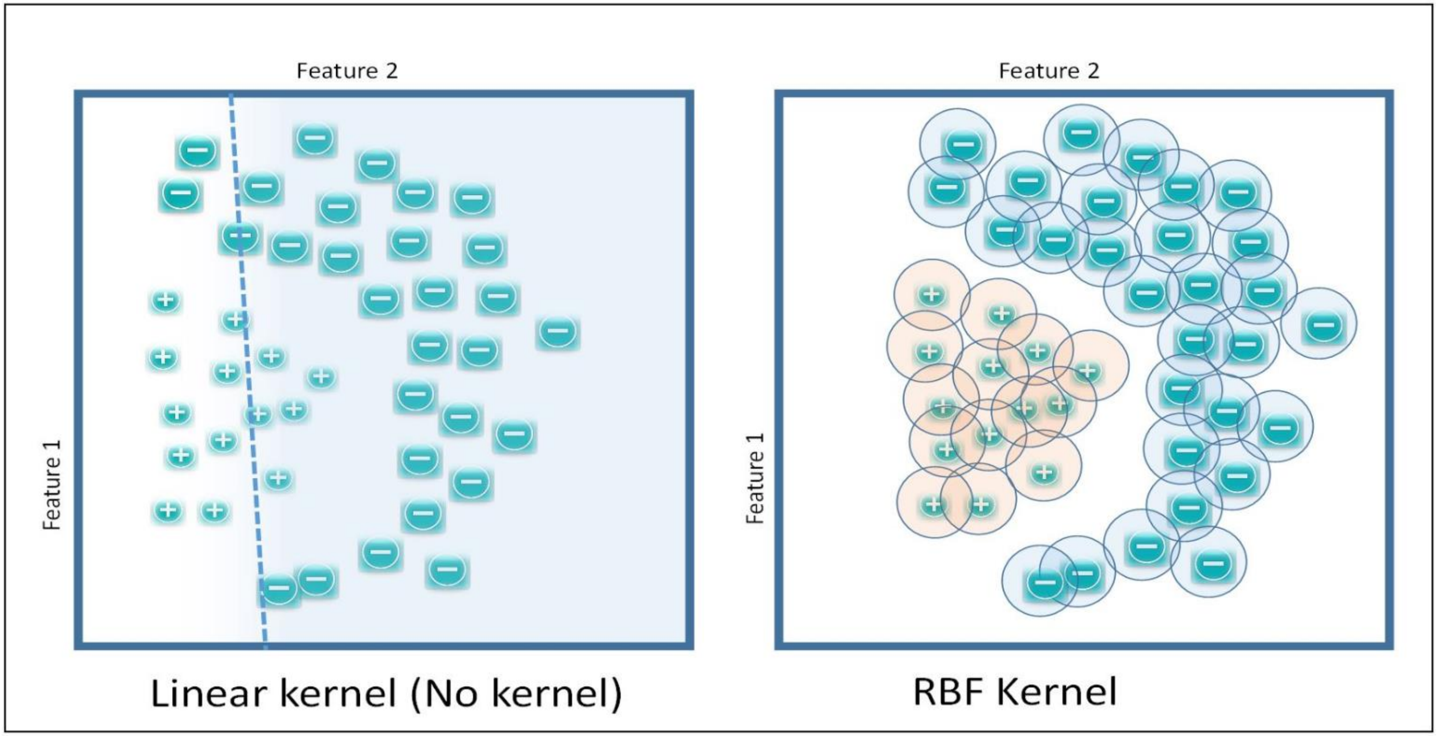
- Most powerful general-purpose kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

- Pretty much a Gaussian with mean x’ and variance σ^2
  - Variance is a parameter to select
- This kernel comes from a space that has infinite dimensions

RBF kernels

- Think of RBF as putting Gaussians onto the support vectors



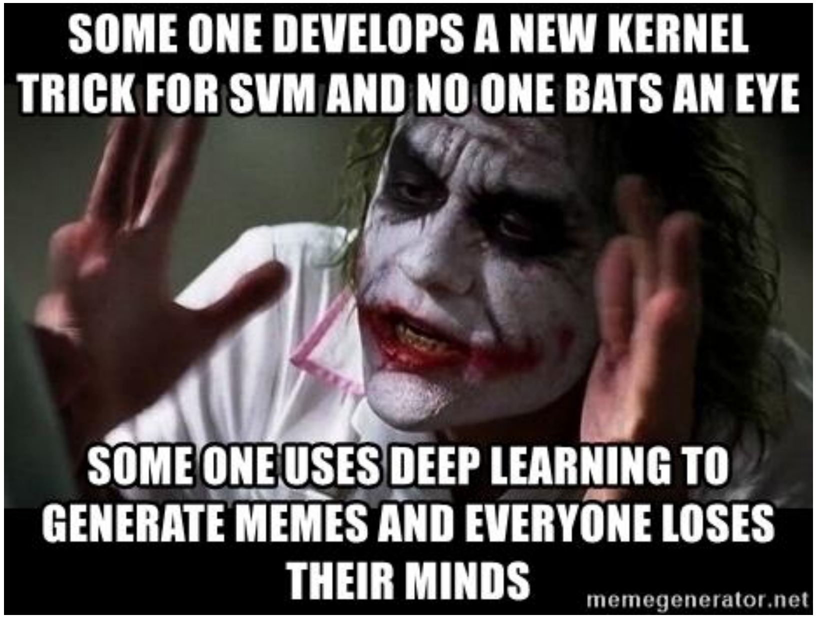
https://www.intechopen.com/books/epilepsy-topics/nonlinear-epilepsy-forewarning-by-support-vector-machines

Design your own kernel

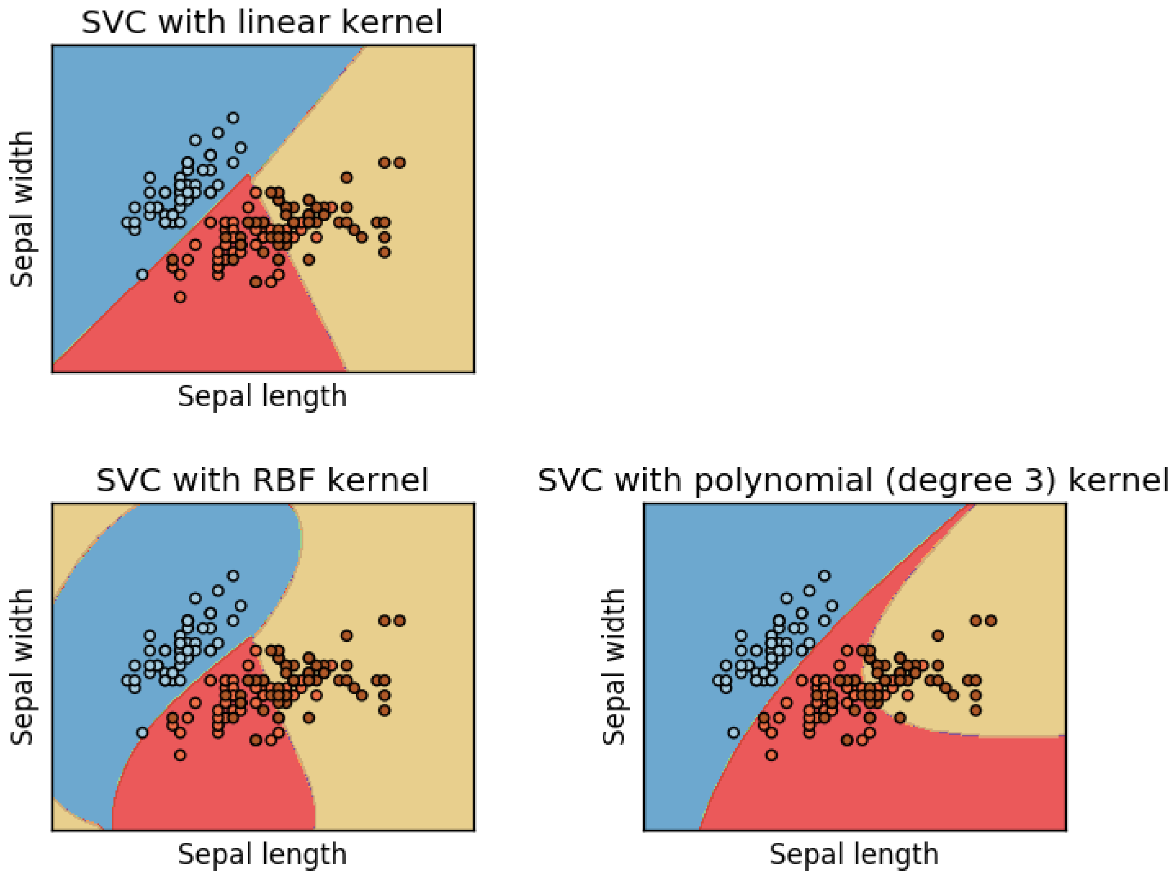
- A kernel is valid if (Mercer's condition)
  - It's symmetric K(x,y) = K(y,x)
  - The matrix of K where $K_{ij} = K(x_i,x_j)$ is positive definite (for any x_i, x_j)

- Build from existing kernels
  - If K1 K2 are valid kernels
    - K = aK1 + bK2
    - K = K1*K2
    - K = K1^(K2)

    are valid kernels



SVM examples

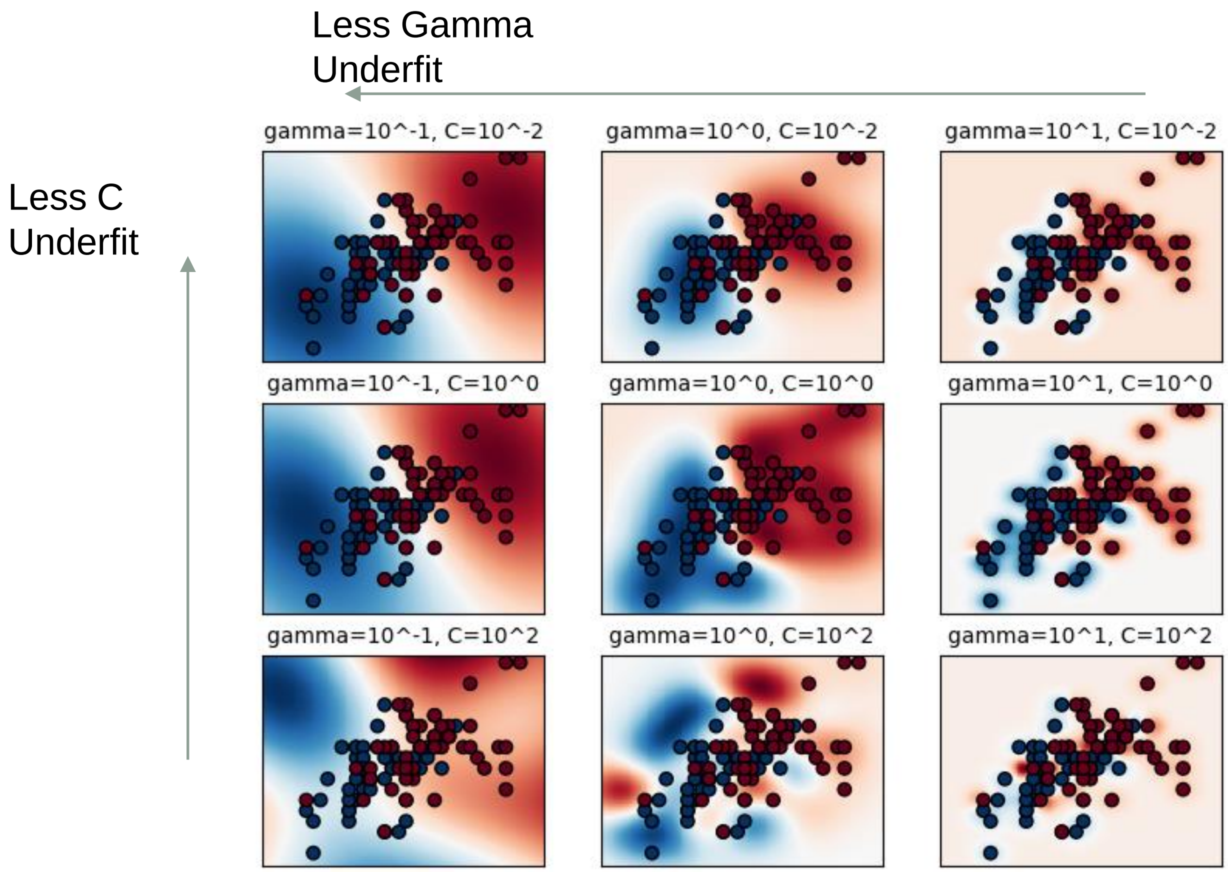


Think of SVM with RBF as K-NN with “support vectors”

http://lijiancheng0614.github.io/scikit-learn/modules/svm.html

RBF SVM and sci-kit learn

- Gamma is the inverse of the variance
- C is the inverse slack variable weight



http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

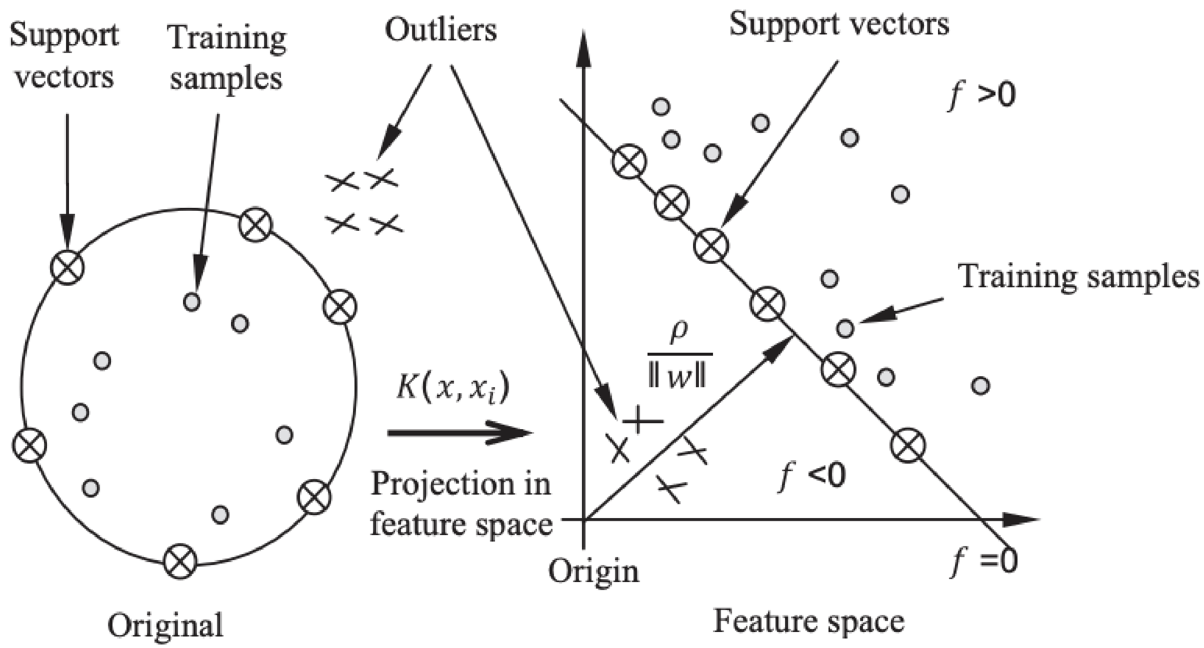
One class SVMs

- Sometimes it is easy to get positive examples but hard to acquire all possible negative examples
  - Email spam filter
    - We kind of know what a good email looks like. And we have lots of examples
    - Hard to model what a spam is. Spammer can change the format and evade detection.

- Solution: train on just the positive class
  - Model what that class looks like
  - Anything that deviates too much from it is considered negative examples

How?

- Separates the data from the “origin” (in mapped space)
- Maximize the distance between data points and the origin



https://www.sciencedirect.com/science/article/abs/pii/S0031320314002751

http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/

SVM objective with slack

- Minimize $\mathbf{w}^T\mathbf{w} + C\Sigma\varepsilon_i$
- Subject to

C is a weight parameter, how much we care about slack

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq 1 - \varepsilon_i \qquad for \quad +ve \qquad class$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \leq -1 + \varepsilon_i \qquad for \quad -ve \qquad class$$

$$\varepsilon_i > 0 \qquad \forall i$$

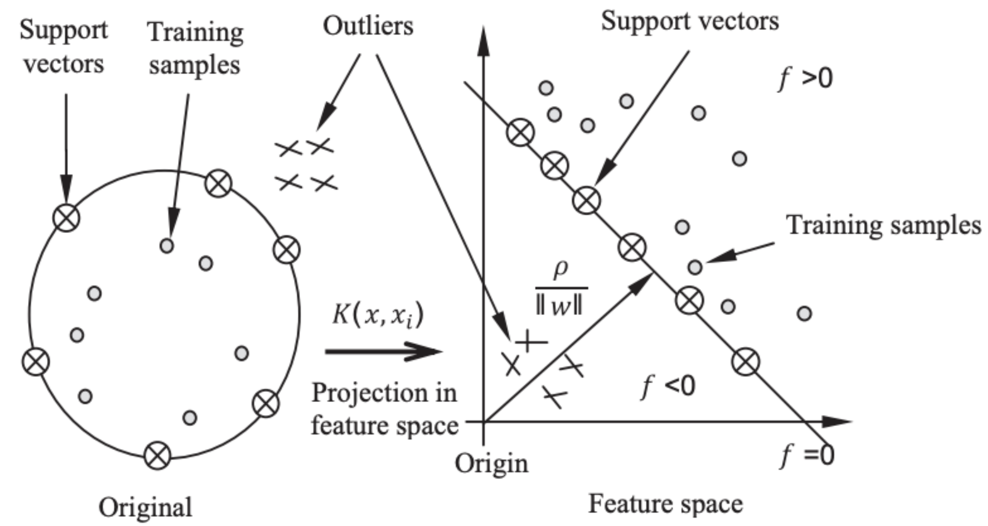
One class SVM with slack

- Minimize $\mathbf{w}^T\mathbf{w}$ + 1/(νn) $\Sigma\varepsilon_i$ - ρ
- Subject to

ρ is the radius of the ball that we also need to optimize

$$\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq \rho - \varepsilon_i$$

$$\varepsilon_i > 0 \qquad \forall i$$



The hyper parameter ν (nu, greek letter n) sets the upper bound of the fraction of training examples to be regarded negative (even though we only put in positive examples). Also called nu-svm

SVM Notes

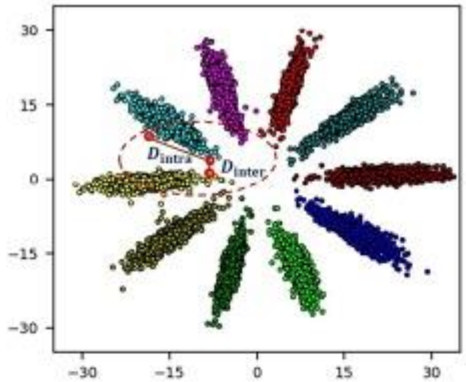
- One of the strongest models and hard to overfit
  - With proper tuning of hyperparameters (kernel, gamma, etc.)
- Convex optimization – no need to try multiple initializations
- Easiest to do one-class setups
- Does not scale well to large datasets
  - Need to compute Gram matrix (sometimes does not fit in memory if too many data points)
  - Inference time and memory requirement during inference depends on the number of support vectors

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$$

Margin-based today

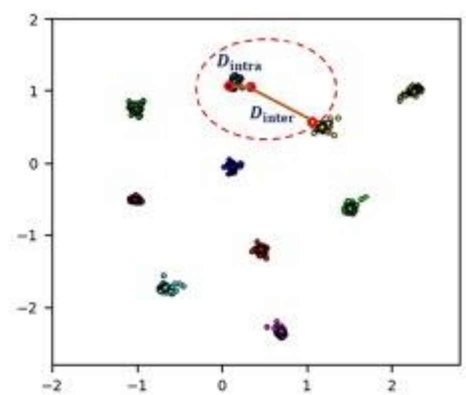
- Many deep learning models now incorporate some kind of margin in the loss function
  - Used in face recognition, embedding-based models

Regular loss without margin



(a) Softmax loss

Loss with margin term



(b) I2CS loss

https://www.sciencedirect.com/science/article/pii/S0925231220318178

Ways to group machine learning models

How do you acquire training data?

Supervised
Unsupervised
Reinforcement

What are you outputting?

Regression
Classification/Clustering

How are you modeling? New!

Discriminative
Generative

Generative Models

- Naïve Bayes, Bayes classifiers are generative models
- Learn the model for each class y given input features x p(x|y). (The likelihood probability)

- To do classification we want to solve for the best y given input feature x (the posterior)

$$y^* = argmax_y P(y|x)$$

- We can use Bayes' rule

$$y^* = argmax_y \frac{P(x|y)P(y)}{P(x)}$$

Generative Models

$$y^* = argmax_y \frac{P(x|y)P(y)}{P(x)}$$

- P(y) is called the prior probability
- P(x) is ignored since we only care for argmax wrt. Y

- Can we use P(y|x) instead?

$$y^* = argmax_y P(y|x)$$

Discriminative models

- Discriminative models model P(y|x) directly

$$y^* = argmax_y P(y|x)$$

- P(y|x) is called the <u>posterior probability</u>
- Generally, P(y|x) can be any function h(x,y) that gives a score for each class
  - Logistic regression
  - SVM
  - Neural networks

Discriminative vs Generative

- Model the posterior P(y|x)
- Care about how to *discriminate* between different classes
- Usually outperforms generative models in classification tasks
- Need to retrain the whole model

- Model the likelihood P(x|y) Learns about how x is *generated* from y
- Worse classification performance. Mismatch between training objective
- Easy to add a new class y' – train P(x|y = y') Example: Adding a new class in Naïve Bayes

Notes on Generative modeling

- Some people say generative models model the joint probability, $p(x,y)$.
- This is also true because when we model $p(x|y)$, we also model the prior $p(y)$.
  - $p(x,y) = p(x|y)p(y)$

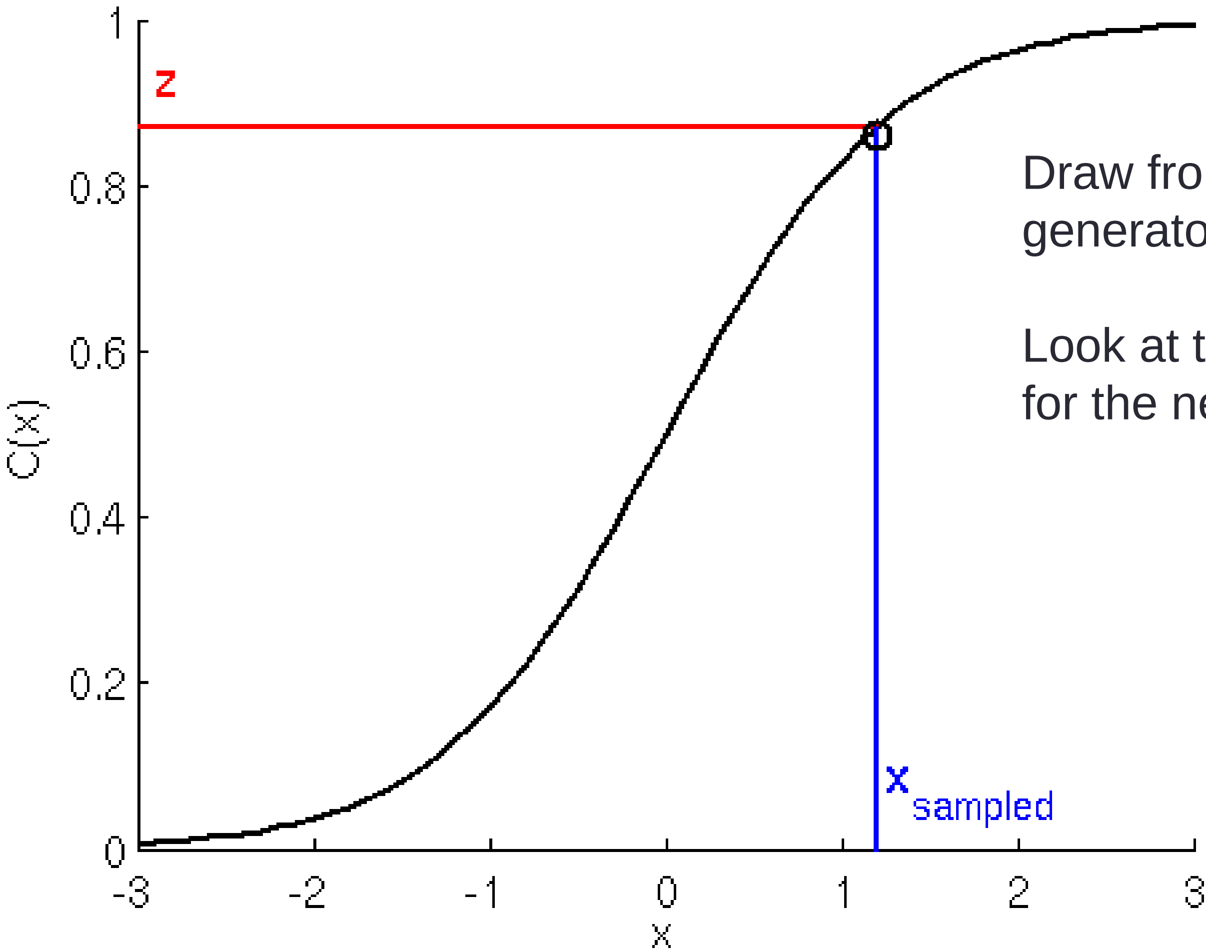
- With this view,
  - Generative models use the joint probability $p(x,y)$
  - Discriminative models use the conditional probability $p(y|x)$

Generating data from generative model?

- We have the joint p(x,y) so we can sample from the distribution for a new (x,y) pair.
- This generates a new data sample x
  - You cannot sample from the posterior p(y|x) because you do not know p(x).

- How to sample from a distribution?
  - Random function usually gives a uniform [0,1]
  - How to sample from arbitrary distribution?

Sampling using the inverse of the CDF



Draw from a uniform random generator

Look at the inverse of the CDF for the new x

Summary

- SVMs
  - Max margin loss
  - Slack
  - Dual-primal
    - Kernel (inner product of higher space)
  - RBF kernels
  - One class SVM