SUPPORT VECTOR MACHINES

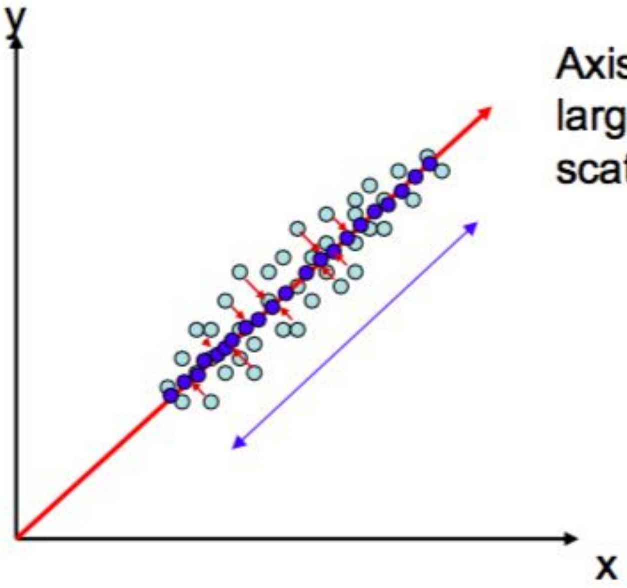
Many slides courtesy of Marios Savvides

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



Axis that describes the largest variation (or scatter)

PCA and LDA Slides from Marios Savvides

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

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

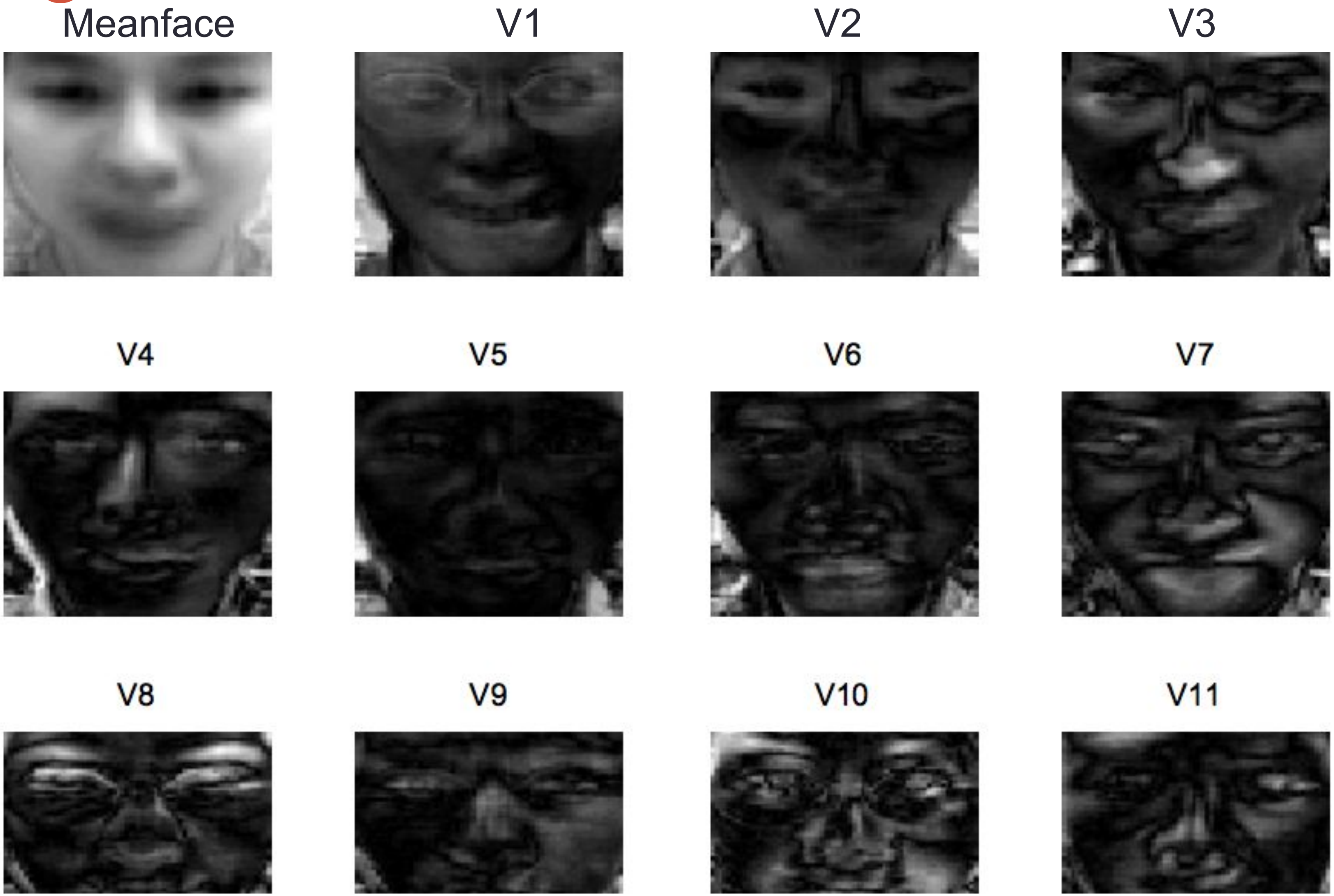
- And our solution yielded

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

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

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

Eigenfaces



Basis decomposition

- Let’s consider our projection w_i which is the eigenvectors to be a basis vector v_i
- We can represent any vector as a sum of basis vectors as follows:

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{Vp}$$

Finding the weights

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i = p_1 \begin{bmatrix} | \\ \mathbf{v}_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ \mathbf{v}_2 \\ | \end{bmatrix} + .. + p_n \begin{bmatrix} | \\ \mathbf{v}_n \\ | \end{bmatrix} = \mathbf{V}\mathbf{p}$$

- If v_i are orthogonal, the projection of x onto v_i gives p_i

$$\mathbf{V}^{\mathrm{T}}\mathbf{x} = \begin{bmatrix} - & \mathbf{v}_1 & - \\ - & \mathbf{v}_2 & - \\ - & \mathbf{v}_3 & - \end{bmatrix} \begin{bmatrix} | \\ \mathbf{x} \\ | \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Means

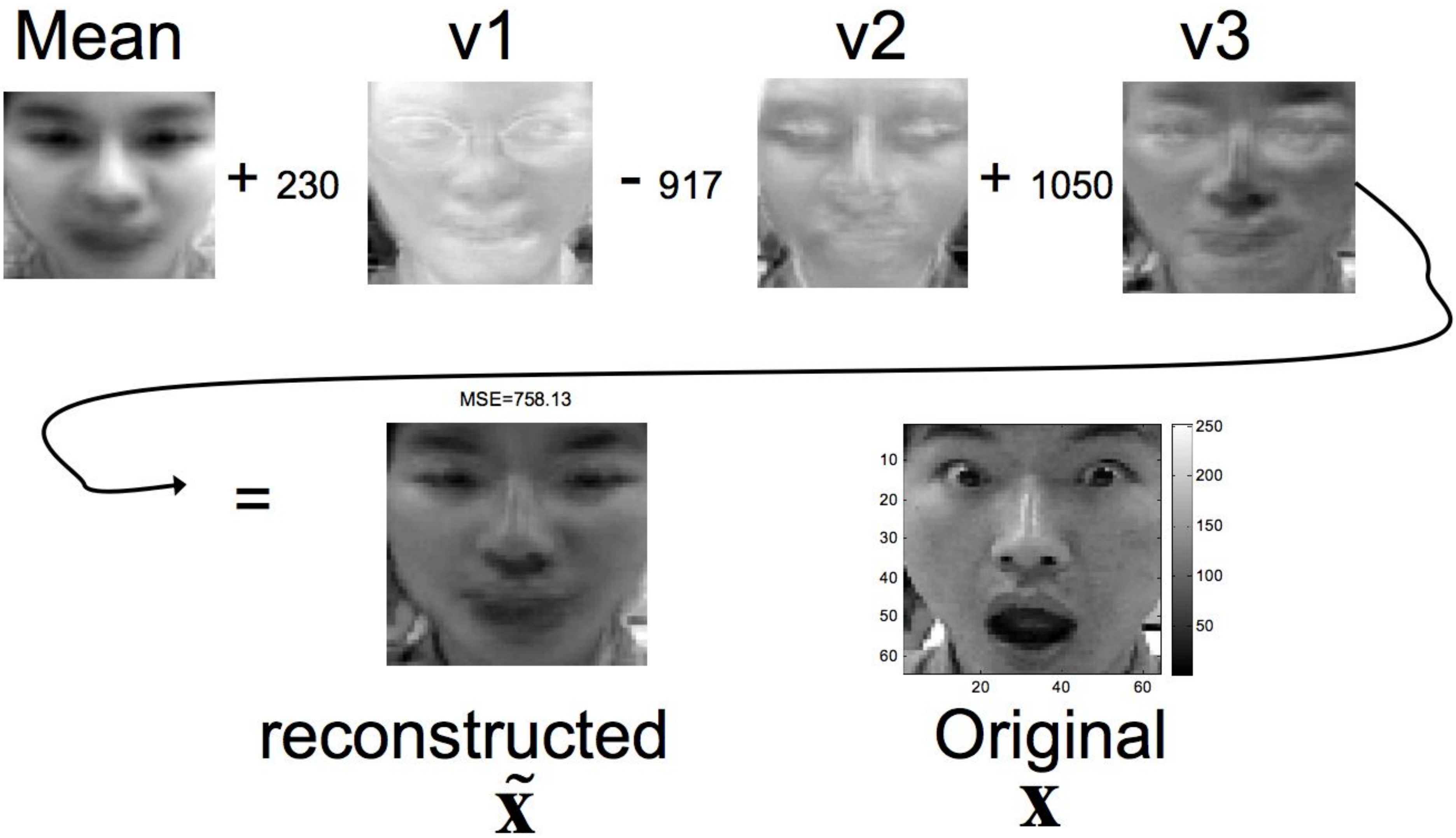
- In PCA, we model variance. (Variation around the mean)
- In our projection we need to remove the mean

$$\mathbf{p} = \mathbf{V}^{\mathbf{T}}(\mathbf{x} - \mathbf{m})$$

- The mean is the mean of all your training data
- If we want to reconstruct the data we need to add back the mean

$$\mathbf{x} = \sum_{i=1}^{N} p_i \mathbf{v}_i + \mathbf{m} = p_1 \begin{bmatrix} \vert \\ \mathbf{v}_1 \\ \vert \end{bmatrix} + p_2 \begin{bmatrix} \vert \\ \mathbf{v}_2 \\ \vert \end{bmatrix} + .. + p_n \begin{bmatrix} \vert \\ \mathbf{v}_n \\ \vert \end{bmatrix} + \mathbf{m} = \mathbf{V}\mathbf{p} + \mathbf{m}$$

Reconstruction with eigenfaces



Mean + 230 v1 - 917 v2 + 1050 v3

= reconstructed $\tilde{\mathbf{x}}$

MSE=758.13

Original $\mathbf{x}$

Practical issues

- If your data has different magnitudes in different dimensions, normalize each dimension before PCA

- If we have 640x640 images = ~400000 dimensions.
- What is the size of the covariance matrix?

Gram Matrix

$$\Sigma=E(x-\mu)(x-\mu)^T=XX^T$$

Covariance matrix is the outer-product of the input matrix

Must solve $\Sigma v = \lambda v$

$XX^T v = \lambda v$ (pre-mult by X^T) (1)

$X^T X X^T v = \lambda X^T v$ ($v' = X^T v$) (2)

Solve eigenvalue problem $X^T X v' = \lambda v'$

- $X^T X$ is a gram of inner-product matrix. Its size is NxN where N is the number of data samples.

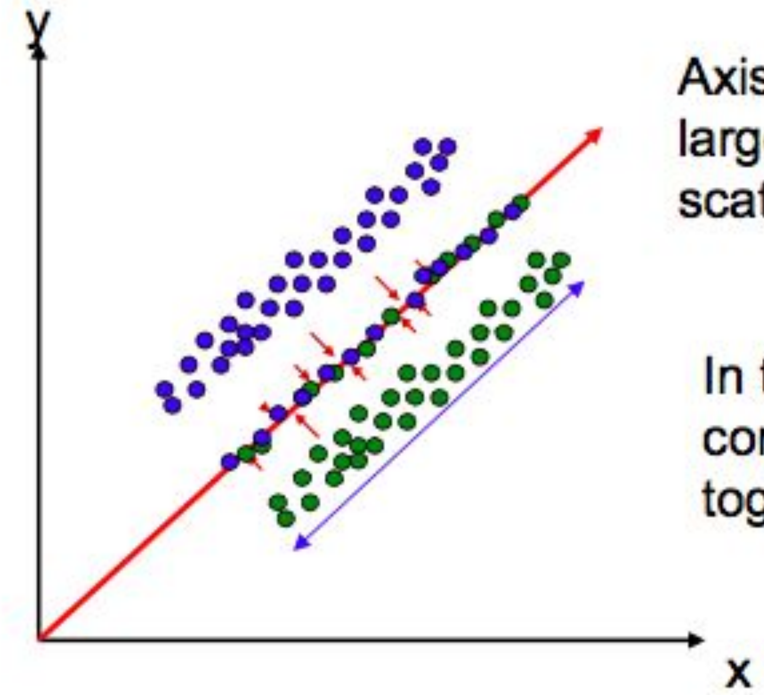
But how to get v from v'?

- From previous slide, equation (1) and (2)
  - $XX^Tv = \lambda v$ (1)
  - $v' = X^Tv$ (2)
- Substitute (2) into (1)
  - $Xv' = \lambda v$

- Thus, $v = Xv'$. We don't care about the scaling term because we will always scale the eigenvector so that it is orthonormal i.e. $||v|| = 1$.

PCA for classification

- PCA does not cares about the class labels

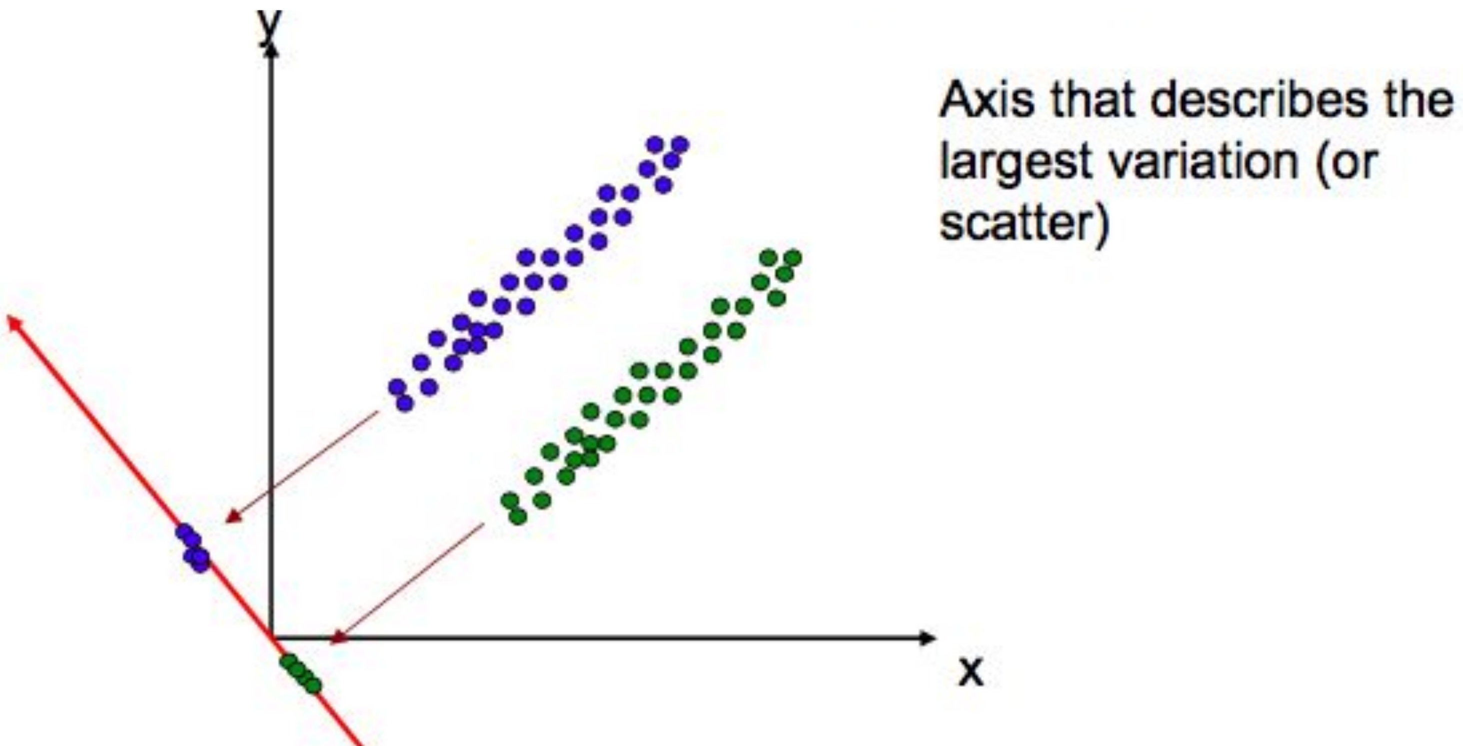


Axis that describes the largest variation (or scatter)…

In this case the projection vector completely smears the two classes together, making them in-separable

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



Simple 2 class case

- We want to maximize the distance between the projected means:

e.g. maximize $|(\tilde{\mu}_1 - \tilde{\mu}_2)|^2$

Where $\tilde{\mu}_1$ is the projected mean $\boldsymbol{\mu}_1$ of class onto LDA direction vector **w** , i.e.

$$\tilde{\mu}_1 = \mathbf{w}^{\mathrm{T}}\boldsymbol{\mu}_1$$

and for class 2: $\tilde{\mu}_2 = \mathbf{w}^{\mathrm{T}}\boldsymbol{\mu}_2$ thus

$$|(\tilde{\mu}_1 - \tilde{\mu}_2)|^2 = |(\mathbf{w}^{\mathrm{T}}\boldsymbol{\mu}_1 - \mathbf{w}^{\mathrm{T}}\boldsymbol{\mu}_2)|^2$$

$$= \mathbf{w}^{\mathrm{T}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\mathrm{T}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\mathrm{T}}\mathbf{w}$$

$$= \mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathbf{B}}\mathbf{w}$$

We also want to minimize within class scatter

- The variance or scatter of each class. We also want to minimize them.

$$\tilde{s}_1^{\,2} = \sum_{i=1}^{N_1} (\tilde{x}_i - \tilde{\mu}_1)^2$$

Minimize the total scatter $\tilde{s}_1^{\,2} + \tilde{s}_2^{\,2}$

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!

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

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S_B} \mathbf{w}}{\mathbf{w}^T \mathbf{S_W} \mathbf{w}}$$

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

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

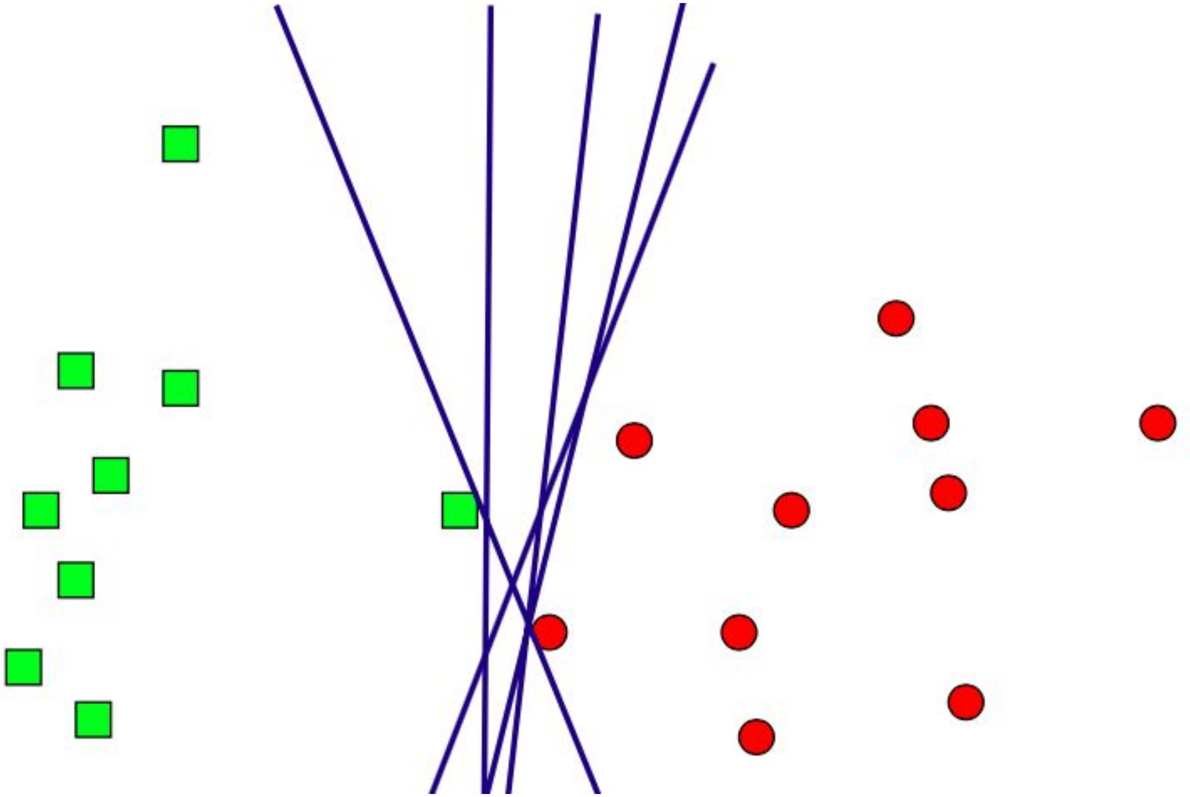
LDA+PCA

- 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

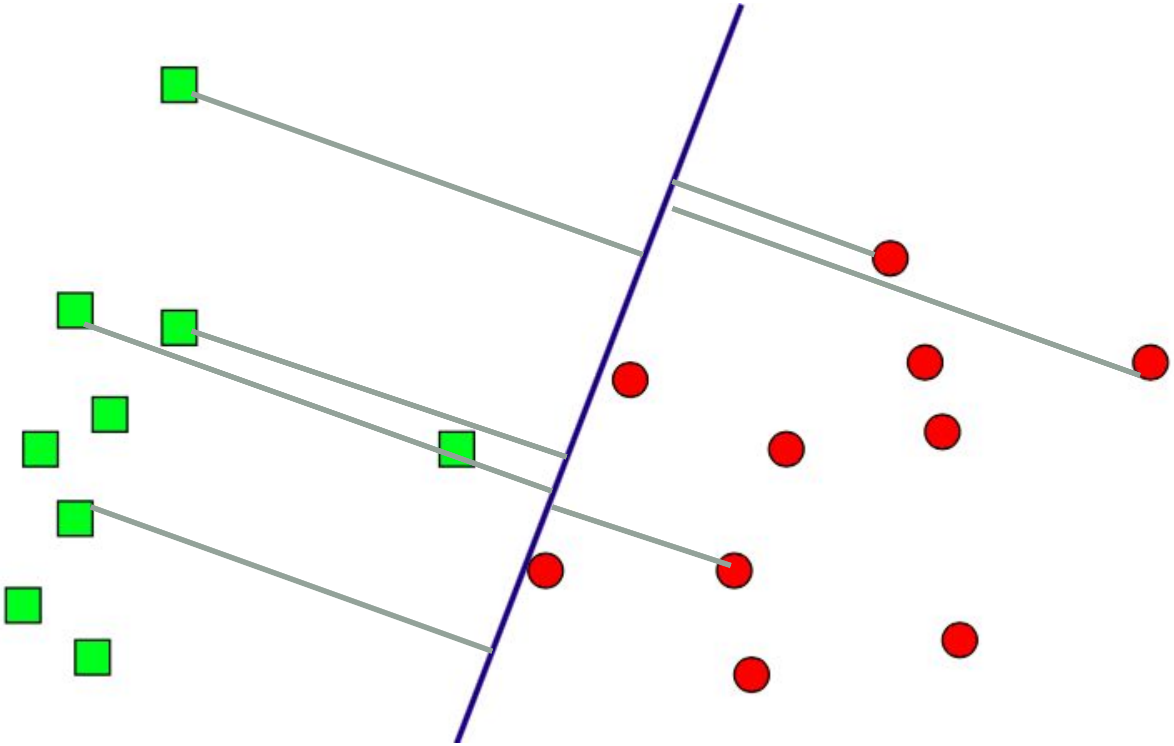
Linear classification problem

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

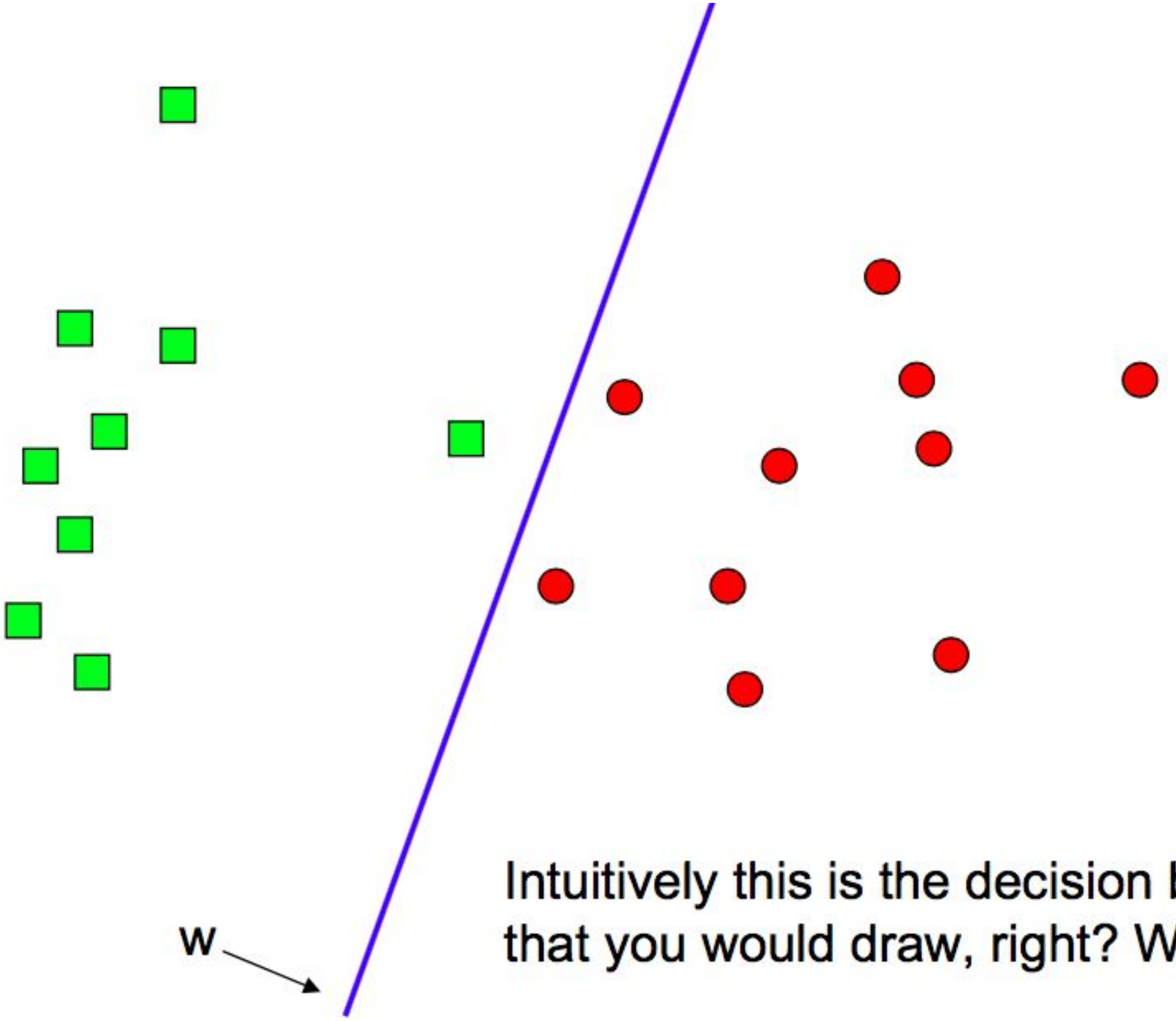


Logistic Regression

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



Support vectors

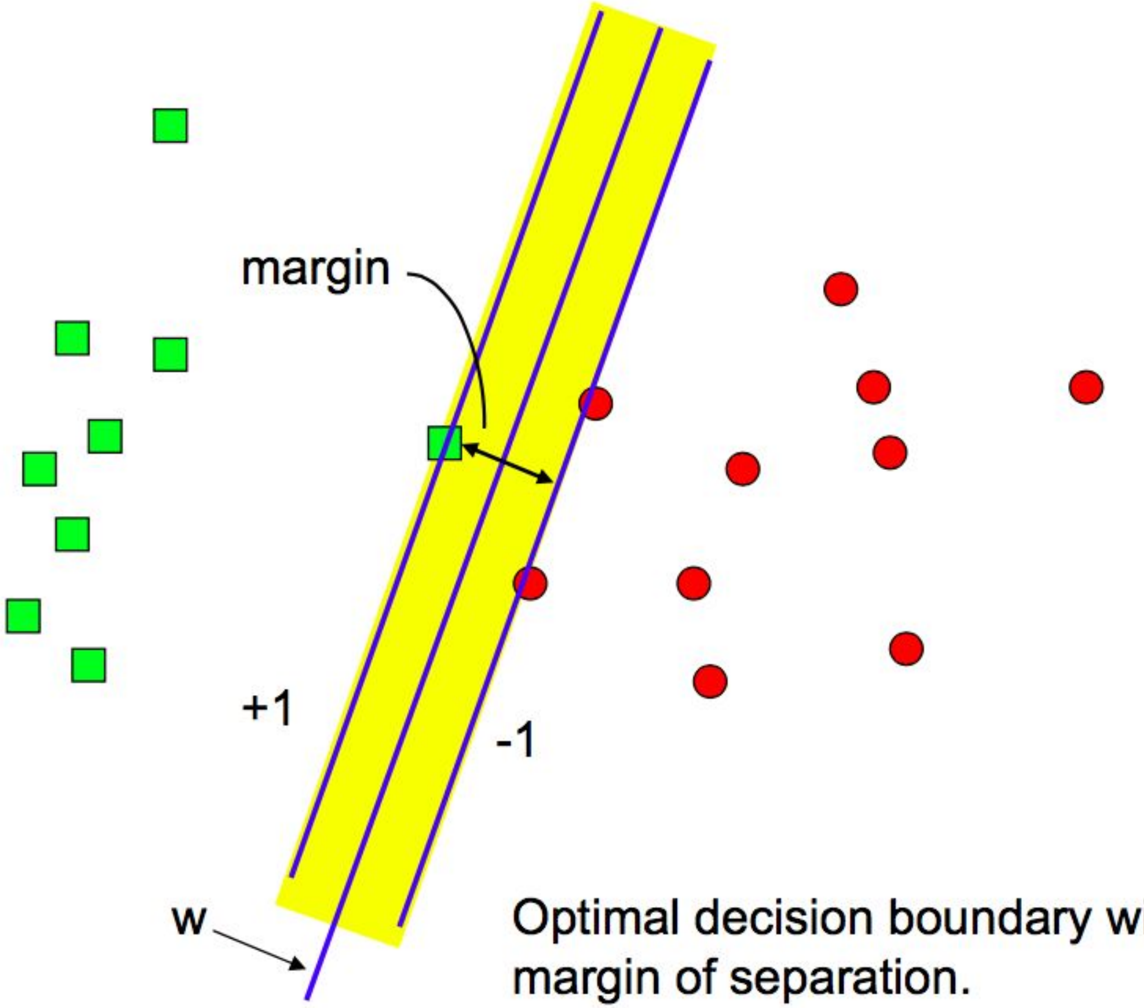


Intuitively this is the decision boundary that you would draw, right? Why?

Support Vector Machines (SVM)

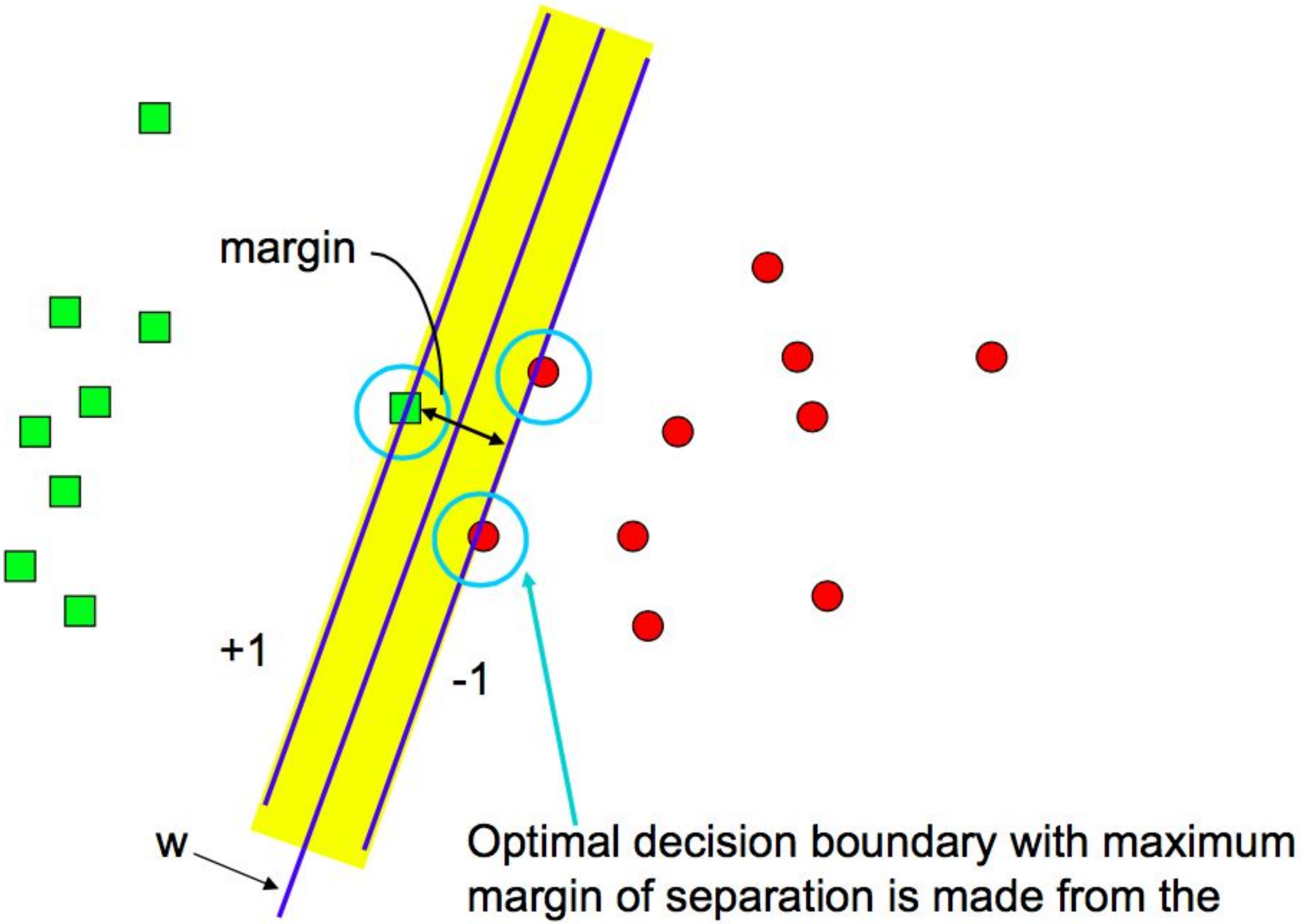
- Goal: improve generalization!
  - Care more about reducing classifier variance than reducing classifier bias
- How?
- Find the decision boundary that gives the most “slack” in classification
  - Don’t care about easy cases, care about borderline cases!
    - Focus on the margin
  - Maximize the “margin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

Support Vectors



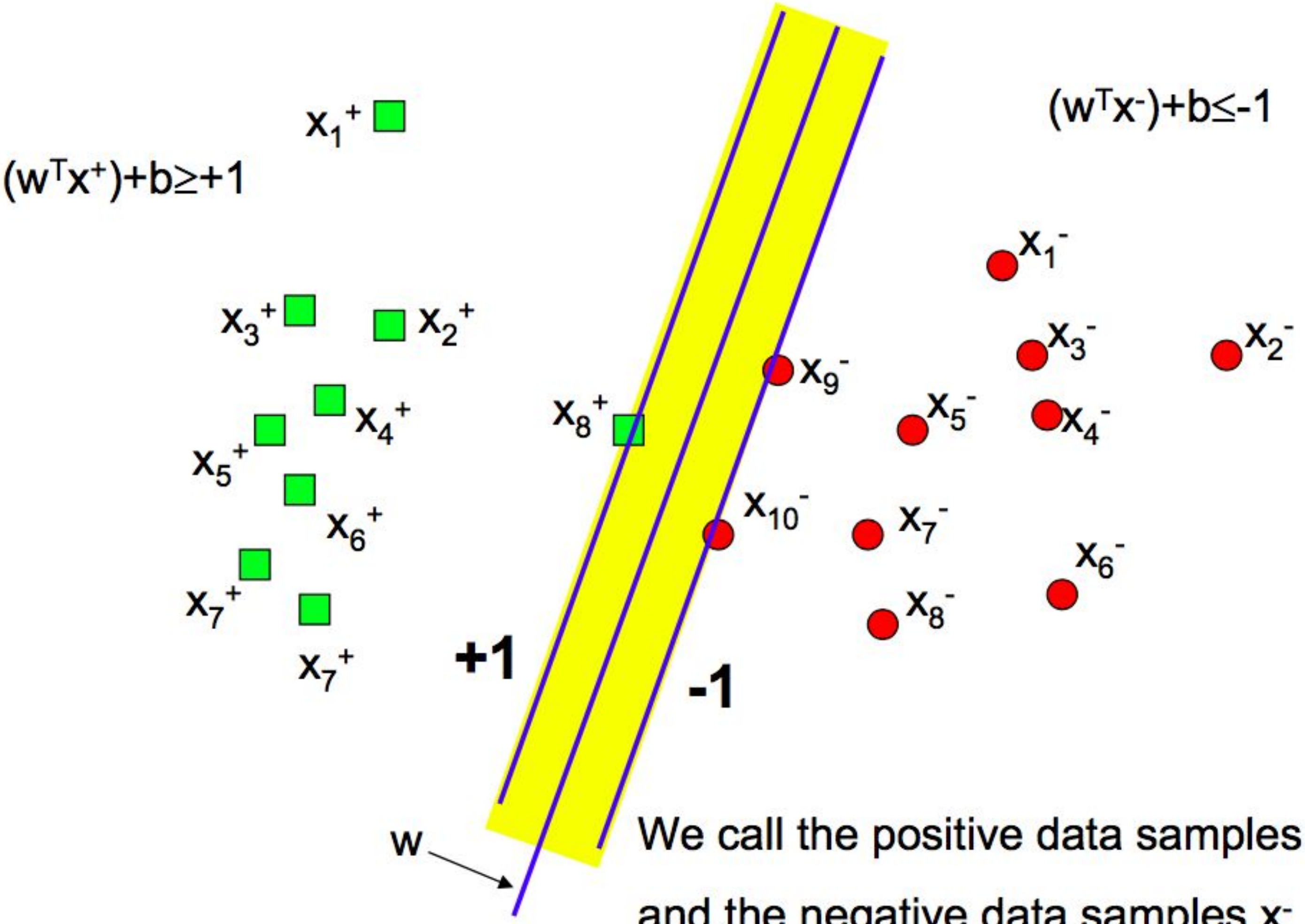
Optimal decision boundary with maximum margin of separation.

Support Vectors



Optimal decision boundary with maximum margin of separation is made from the support vectors (that *support* this margin)

Support Vectors

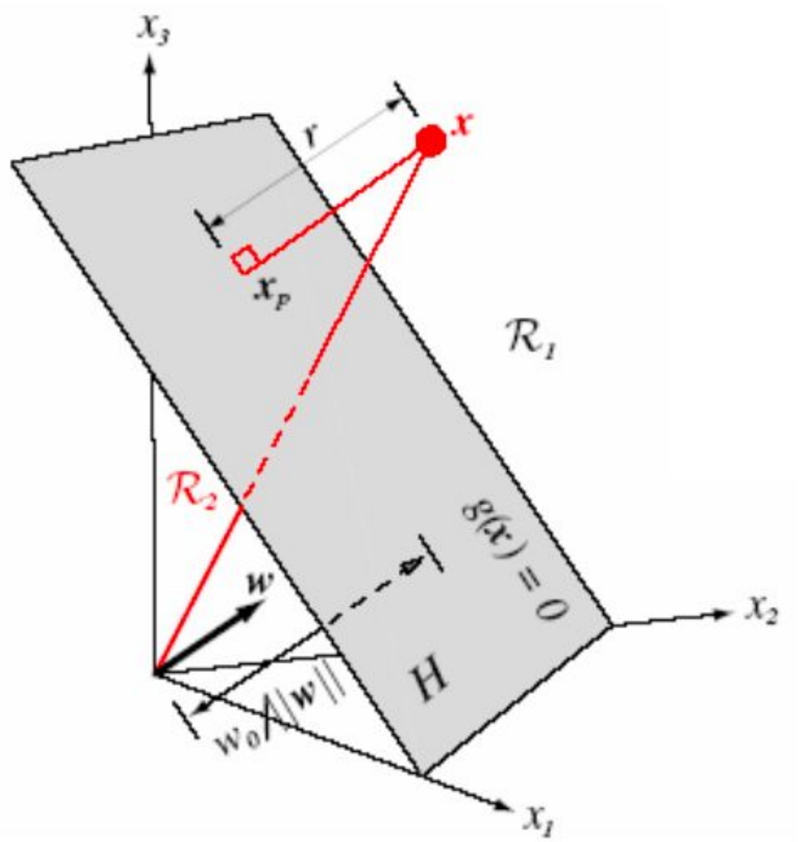


We call the positive data samples x^+ i.e. $(w^Tx^+)+b\geq+1$

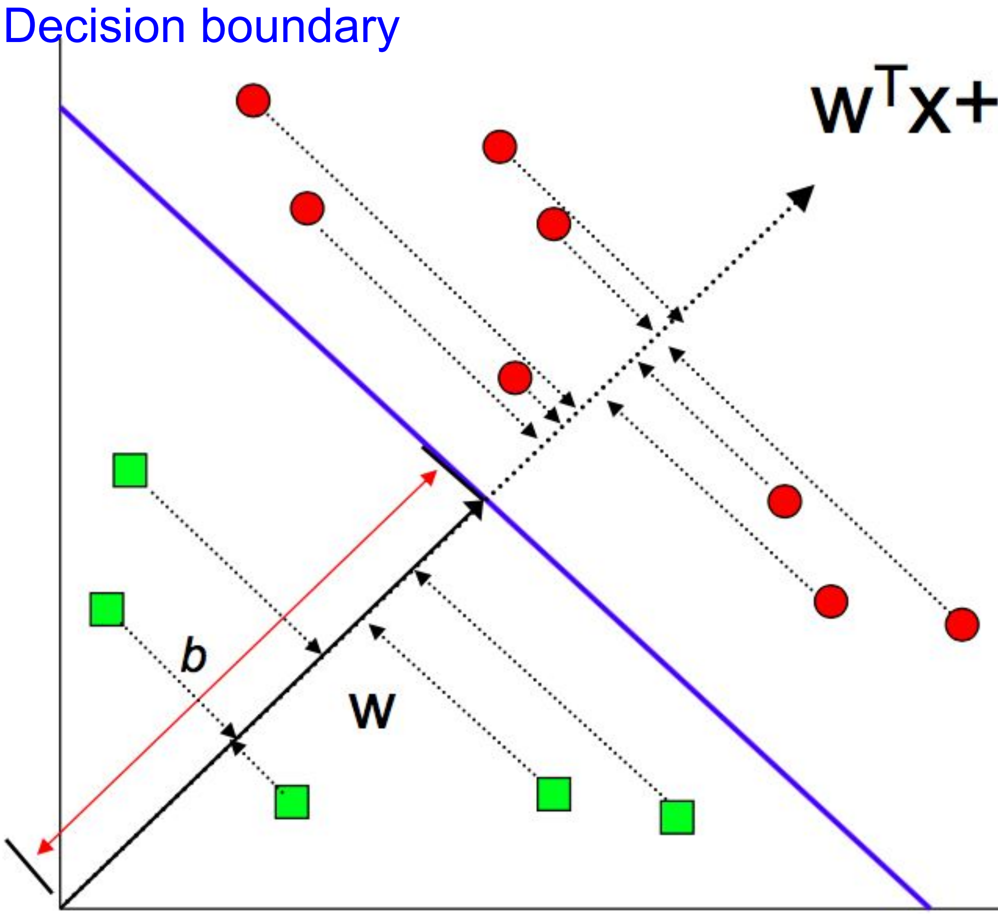
and the negative data samples x^- i.e. $(w^Tx^-)+b\leq-1$

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
  - $\mathbf{w}^T\mathbf{x}_1 + w_0 = \mathbf{w}^T\mathbf{x}_2 + w_0 = 0$
  - $w^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$
    - 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



Geometric interpretation

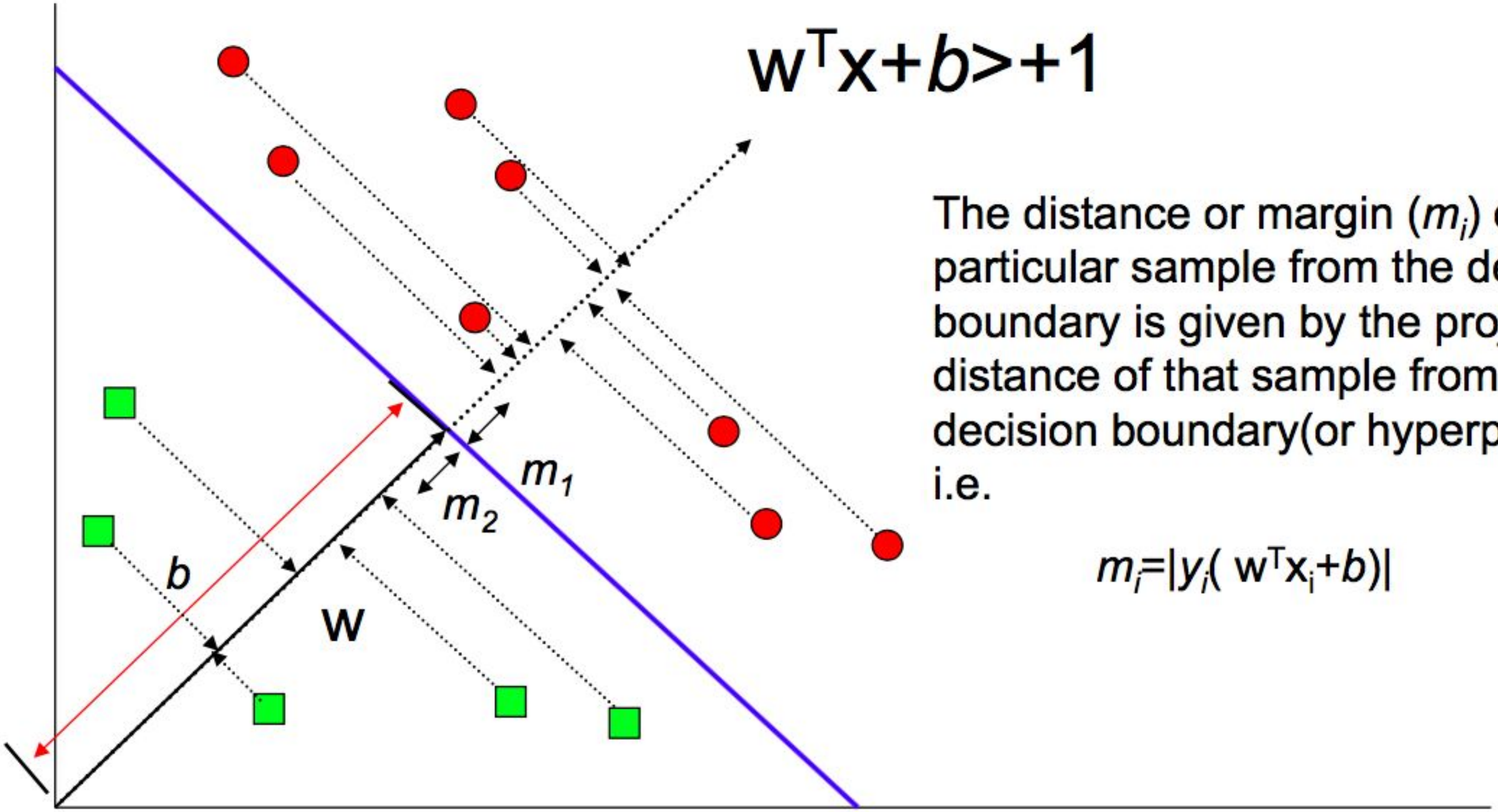


The decision boundary is orthogonal (perpendicular/normal) to the solution vector w.

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.

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

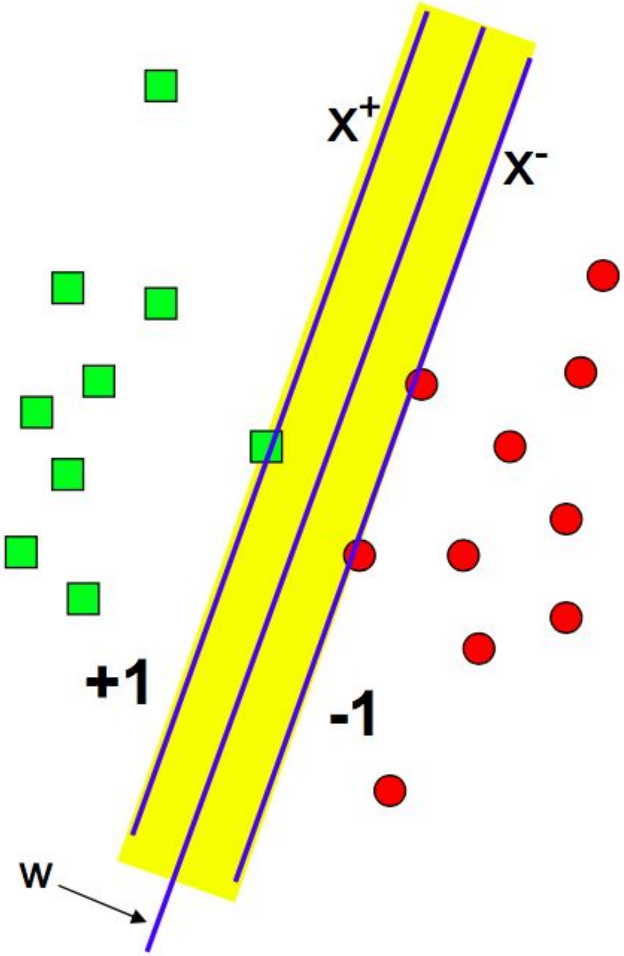
Margins



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.

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

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 functional margin of the resulting classifier m is

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

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

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

< > denotes dot product

Max margin

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

SVM objective function

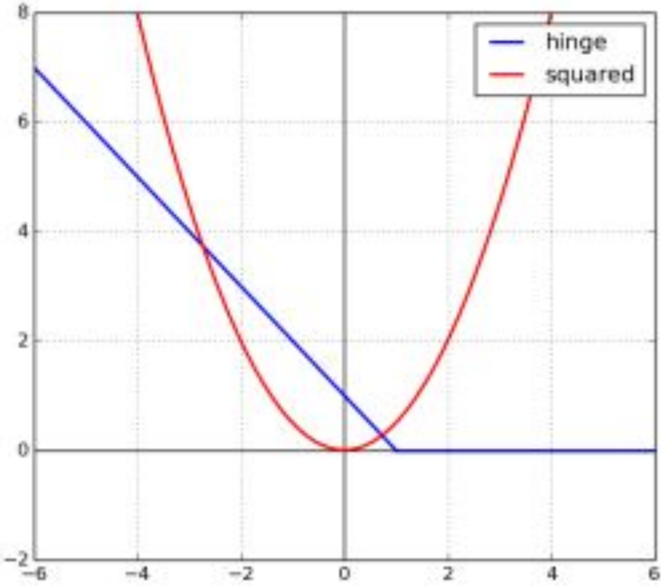
- Minimize $\mathbf{w}^T\mathbf{w}$
- Subject to
  - $y_i(<\mathbf{w}, \mathbf{x}_i> + b) = y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- $y_i = \{+1,-1\}$ 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)

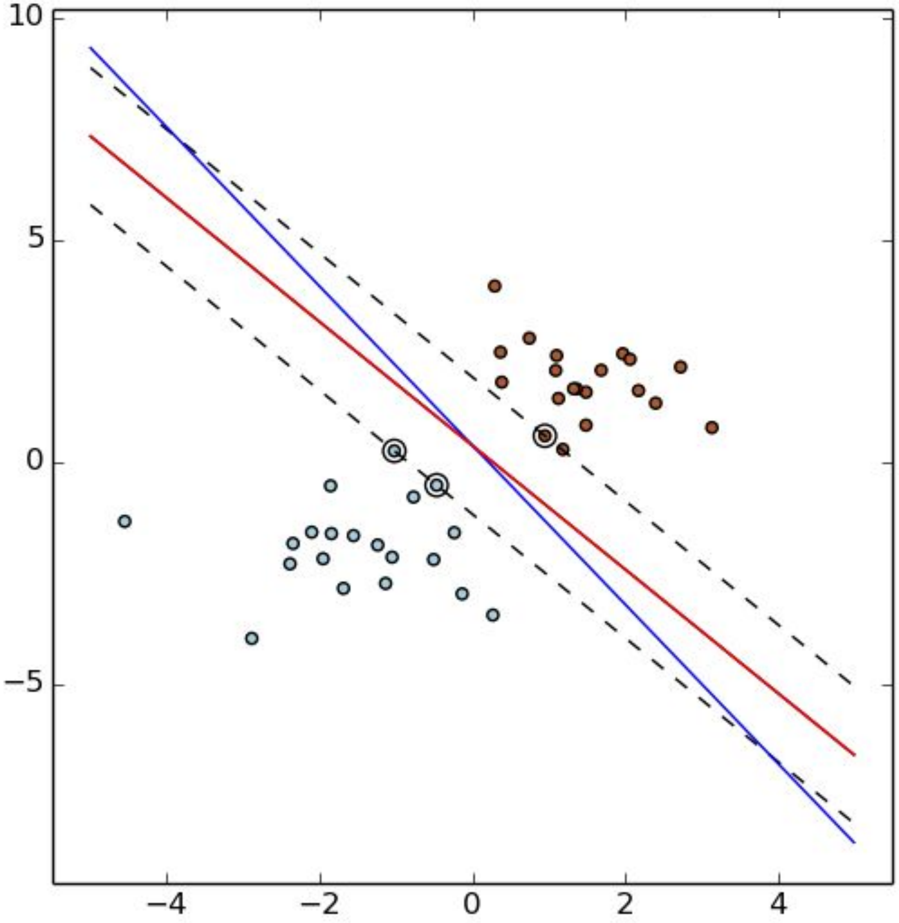
Notes on the Losses

- Linear regression optimizes for the L2 loss (squared loss)
  - Squared distance of data points to boundary $(x- h(x))^2$
- 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://cvstuff.wordpress.com/2014/11/29/latex-l_1-versus-latex-l_2-loss-a-svm-example/

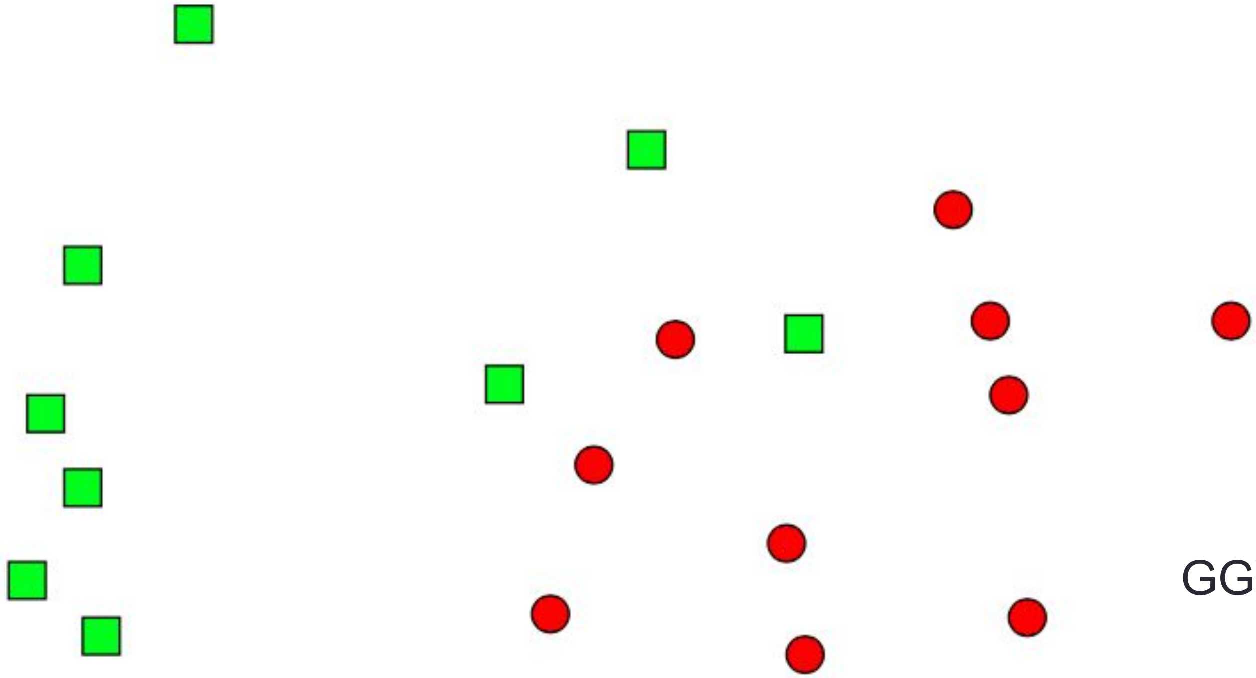
L2 vs hinge loss



https://cvstuff.wordpress.com/2014/11/29/latex-l_1-versus-latex-l_2-loss-a-svm-example/

Linearly non-separable

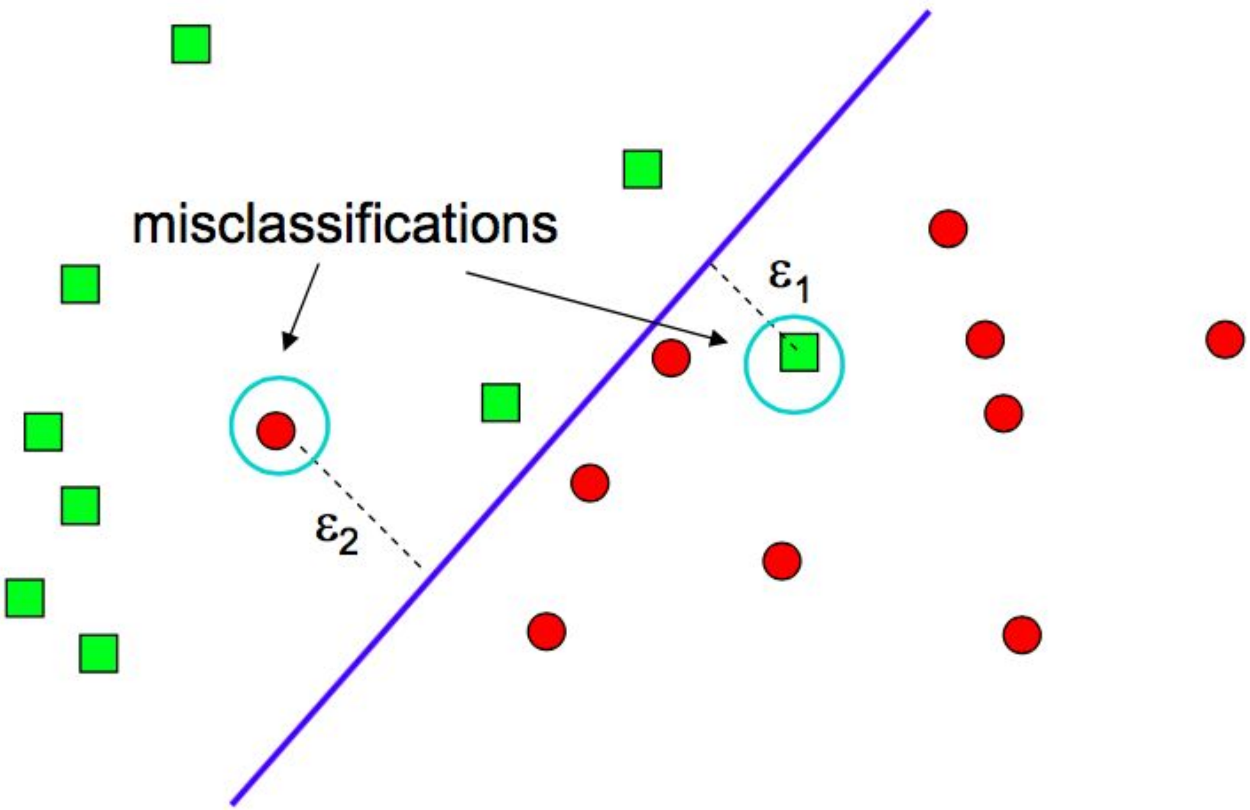
- What happens when you cannot separate the two classes with a linear boundary



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.

<- Not this slack
But this slack also helps get jobs done.

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)

SVM objective with slack

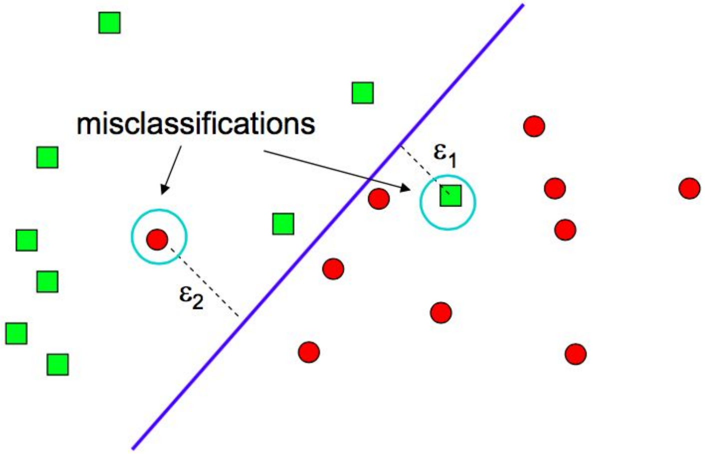
- Minimize **w**^{T}**w** + CΣε$_i$
- Subject to

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

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

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

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



Notes about slacks

- Even if the problem has linear separability we might want some slack still.
  - Missed label 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 that make sense
      - $\mathbf{w}^T\mathbf{w} + C\Sigma\varepsilon_i$
      - 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

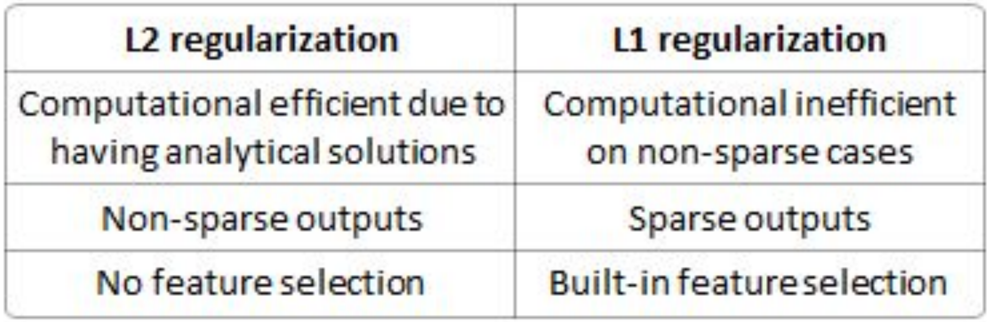
$$p(\theta|\mathbf{x}) = \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})}$$

- 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
  - $\mathbf{w}^T\mathbf{w} + C\Sigma\varepsilon_i$
- L2 regularization: Regularizing term is a sum of squares
  - $\mathbf{w}^T\mathbf{w} + C\Sigma\varepsilon_i^2$



L2 regularization	L1 regularization
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature selection

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:

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

- 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

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

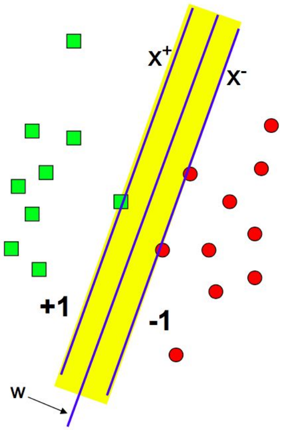
$$\sum_{i=1}^{N}\alpha_i y_i=0$$

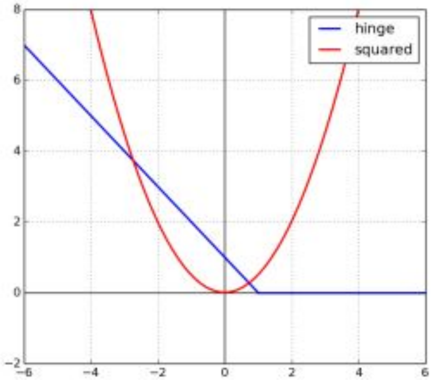
Primal form solution

- 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:

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

- 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 gives us?

- Dual form

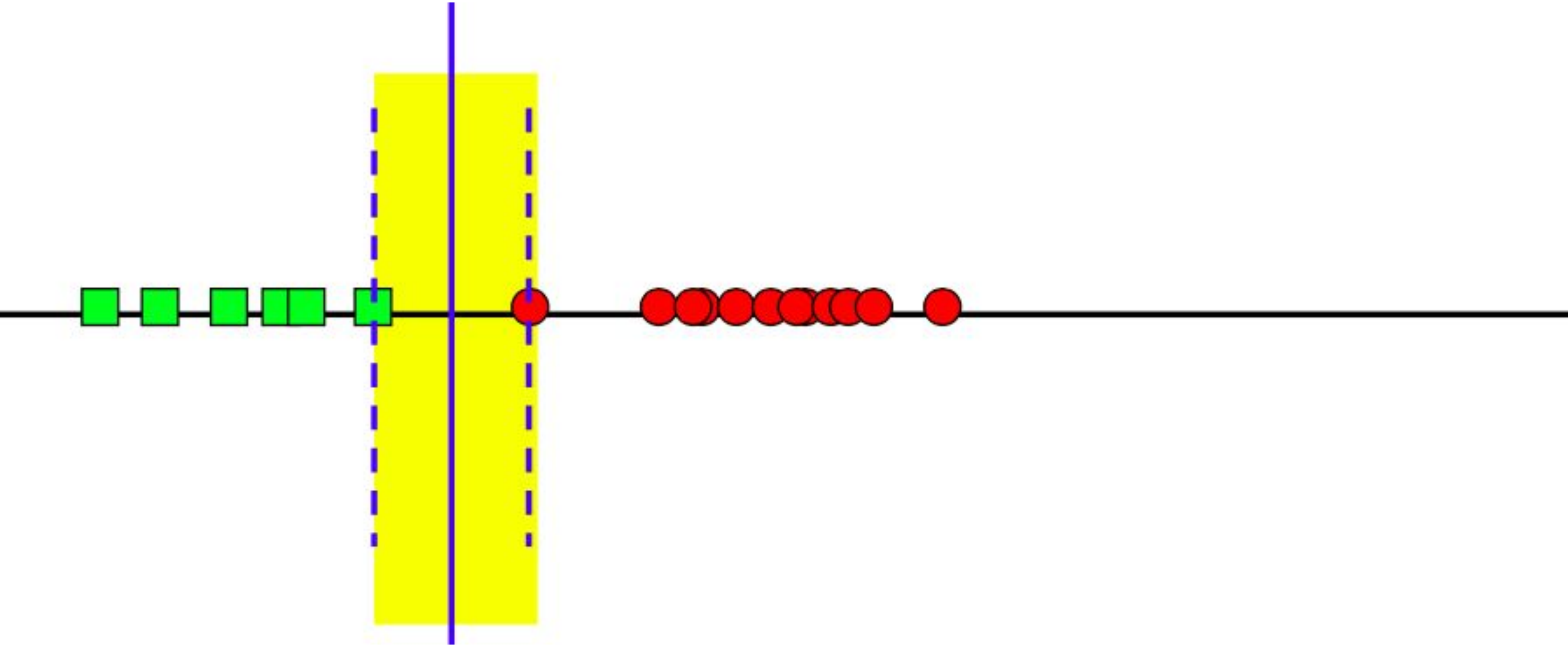
$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

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

- How is this useful?

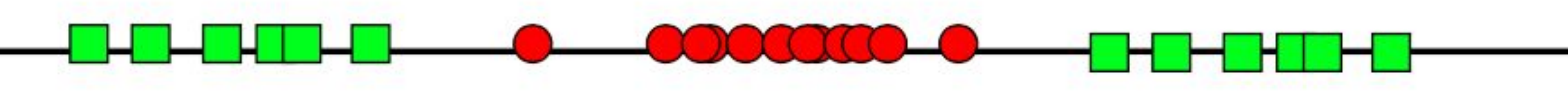
Example SVMs

- Easy



Example SVMs

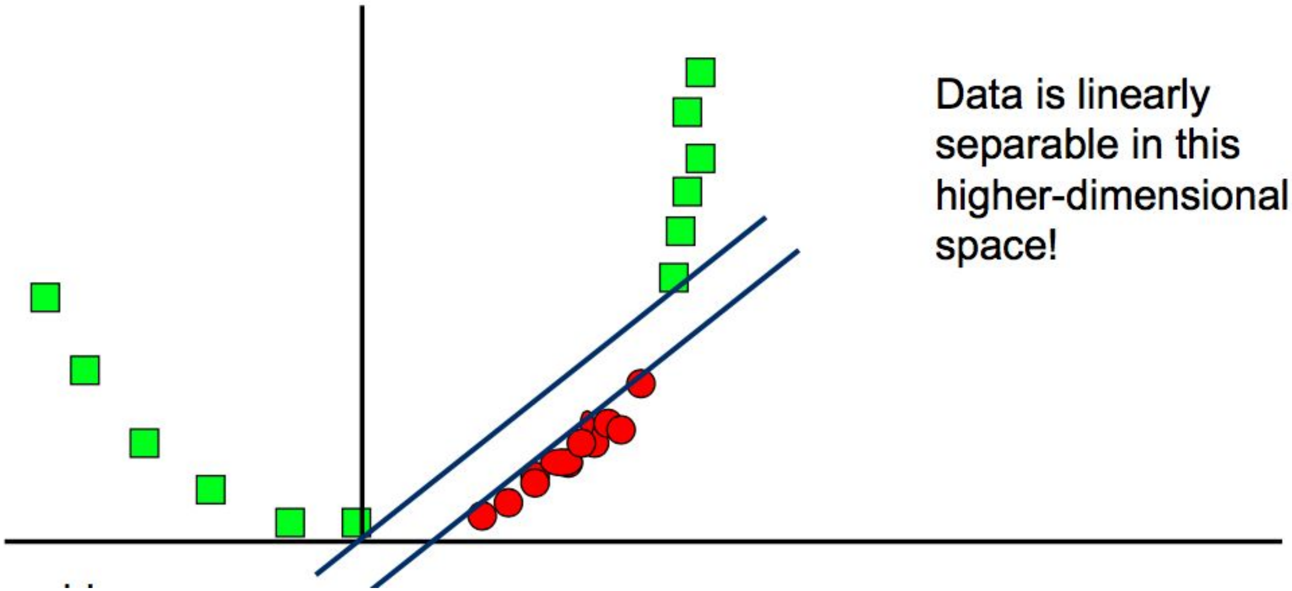
- ??????



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
- One aspect of this means, 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

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

Mapping function dual form

- In the dual form we solve

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

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)

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

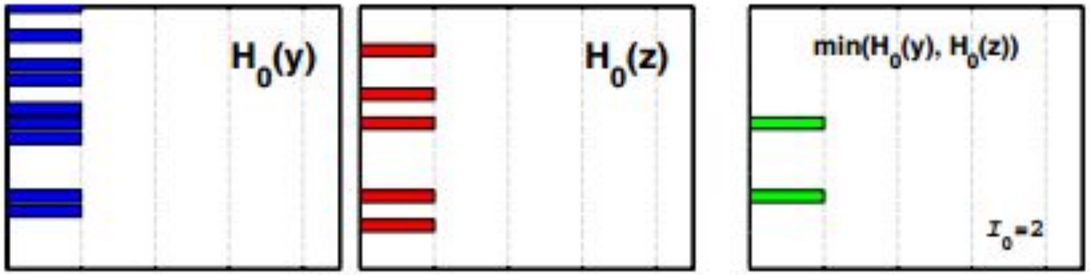
- Kernel of x -> (x, x^2)
  - $xy + x^2y^2$
- Kernel of x -> (x, x^2,x^3)
  - $xy + x^2y^2+x^3y$

Kernel functions

- Sometime 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

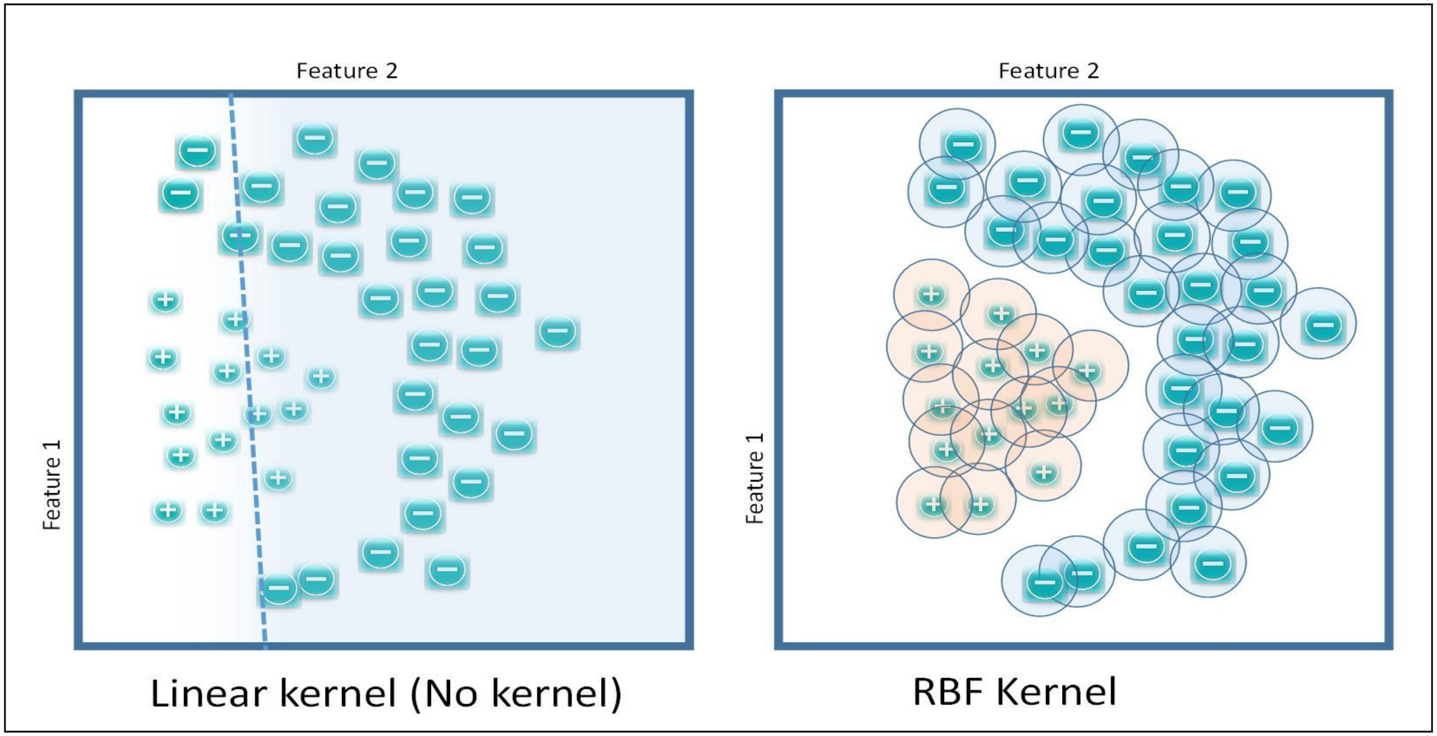
- Most powerful general purpose kernel

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

- 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

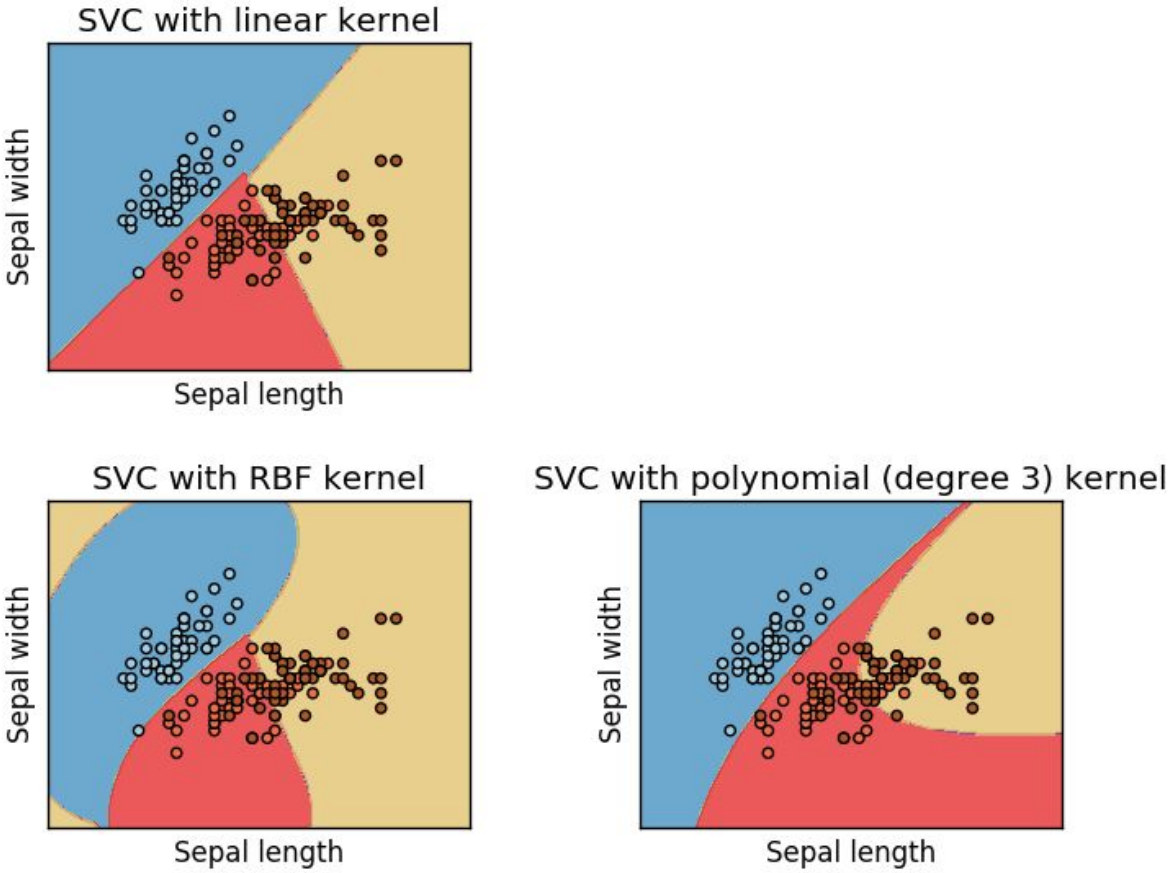
Design your own kernel

- 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



SVC with linear kernel

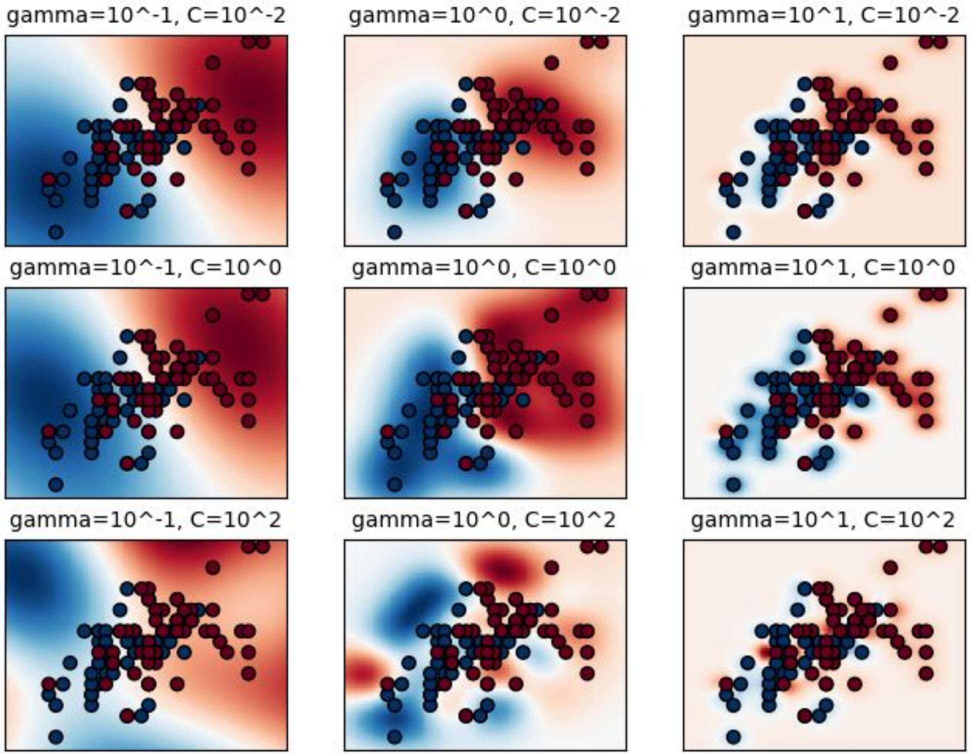
SVC with RBF kernel

SVC with polynomial (degree 3) kernel

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

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 $\mathbf{w}^T\mathbf{w} + 1/(vn)\ \Sigma\varepsilon_i - \rho$
- Subject to

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

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

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

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

Ways to group machine learning models

How do you acquired training data?

Supervised
Unsupervised
Reinforcement

What are you outputting?

Regression
Classification/Clustering

What 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(y,x) 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 performance but can be used for other tasks (simulate data)
- Easy to add a new class y' – train P(x|y = y')

Notes on Generative modeling

- Some people say generative models use 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)

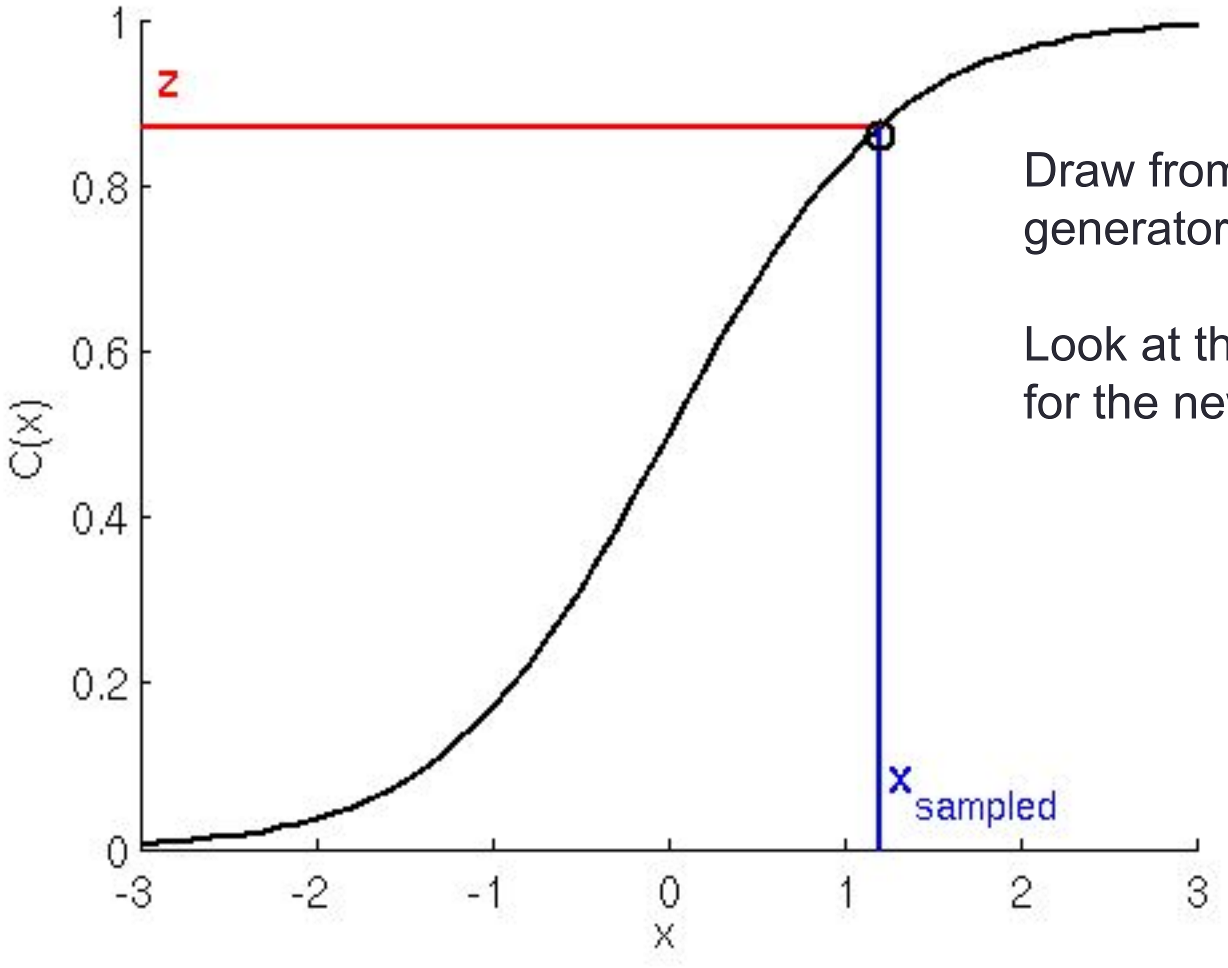
- With this view,
  - Generative models use the joint probability p(x,y)
  - Discriminative models use the conditional probability p(y|x)

Generating data from generative model?

- We have the likelihood p(x|y) so we can sample from the distribution for a new x that comes from that class
- This generates a new data sample 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

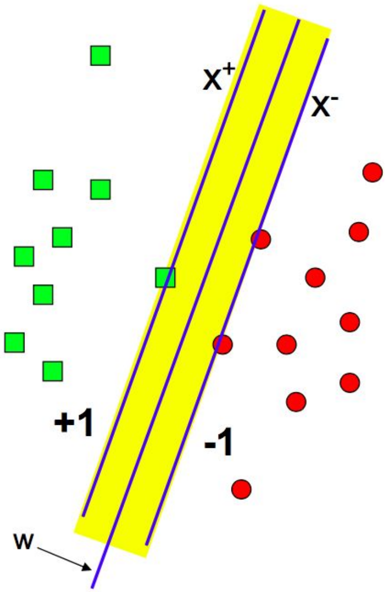


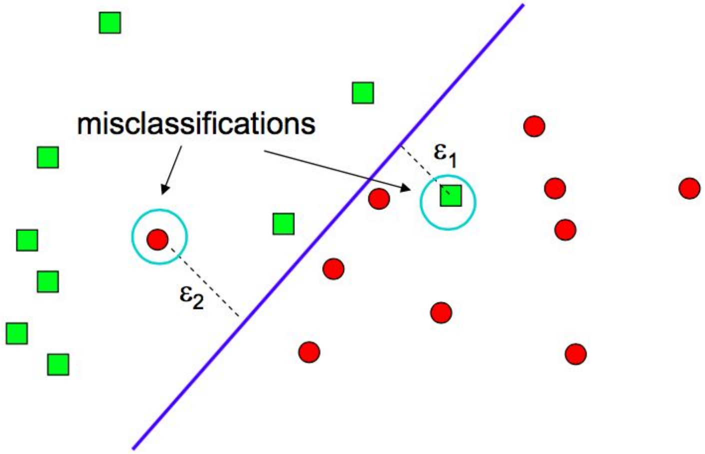
Draw from a uniform random generator

Look at the inverse of the CDF for the new x

Summary

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





misclassifications
ε1
ε2