CS 383/613 – Machine Learning

Support Vector Machines

Slides adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2nd Ed.),
Pattern Recognition and Machine Learning

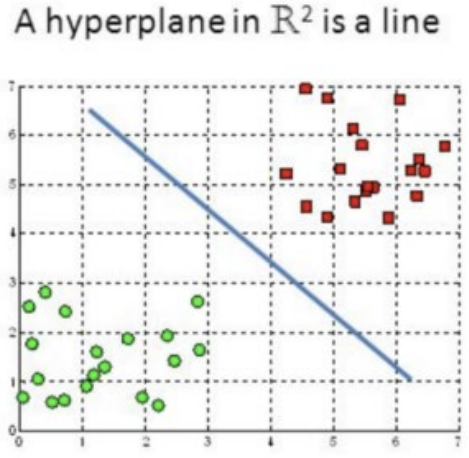
Objectives

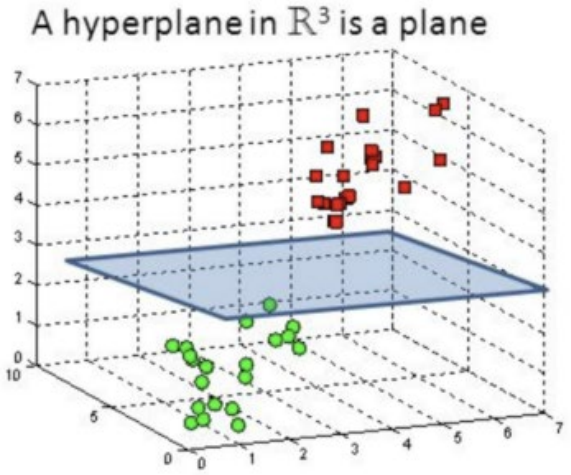
- Support Vector Machines
  - Large Margin Intuition
  - Optimization Objective Function
  - Non-Linear SVMs

SVMs

- Support Vector Machines (SVMs) are a linear classification algorithm that have shown to be quite successful in pattern recognition.

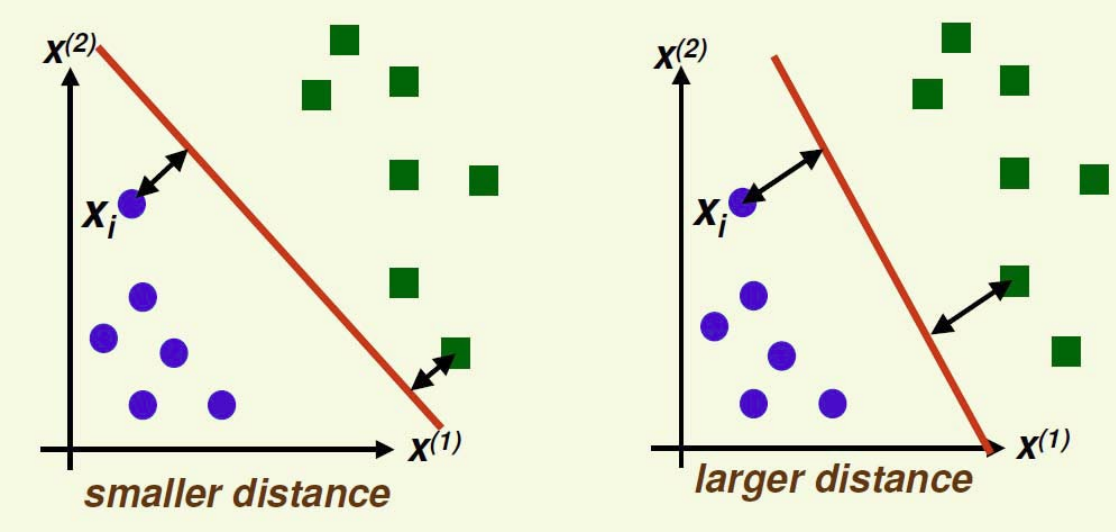
https://towardsdatascience.com





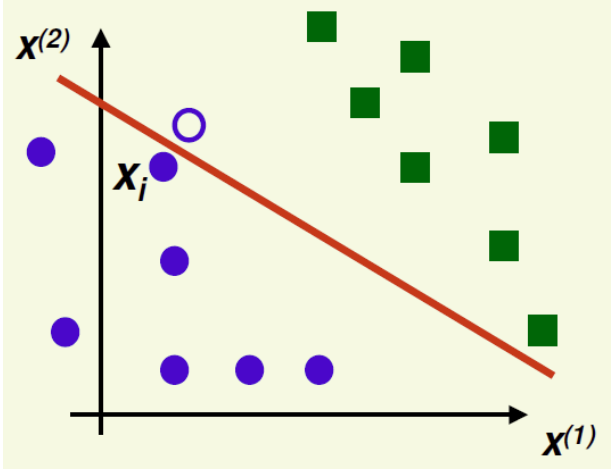
SVM Intuition

- Idea: Maximize distance to closest example (of each type)
  - For now, we'll assume total separability



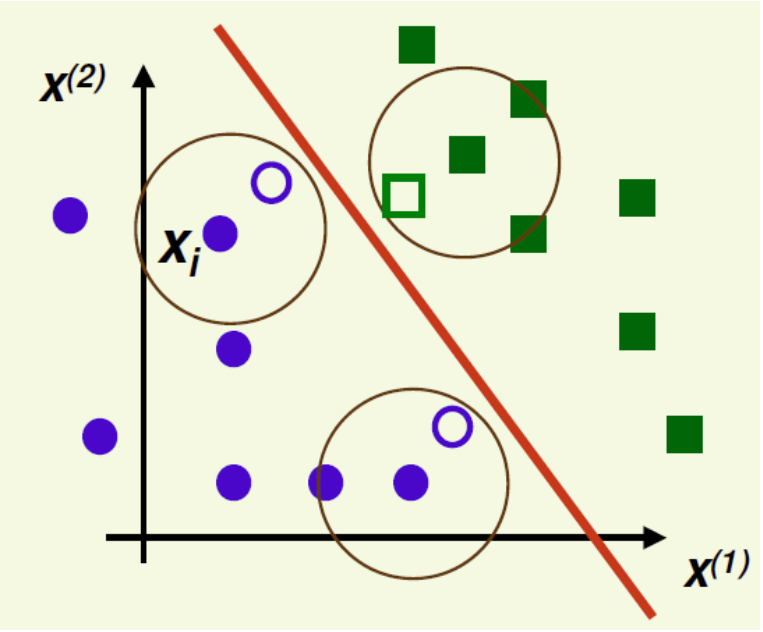
SVM Intuition

- Training data is just a subset of all possible data
  - Suppose hyperplane is close to sample $\boldsymbol{X}_i$ (open circle)
    - The *margin* is small
  - If we see a new sample close to $\boldsymbol{X}_i$ it may be on the wrong side of the hyperplane
- Therefore, poor generalization



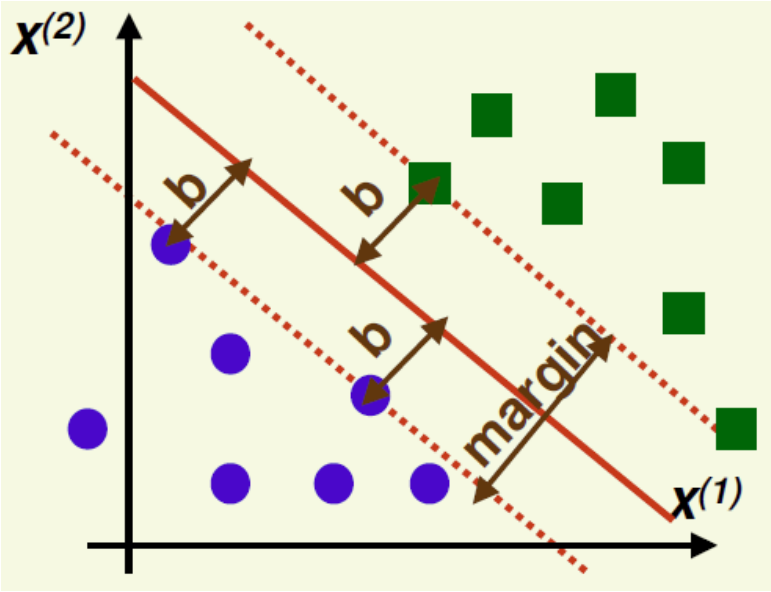
SVM Intuition

- Intuition: We want hyperplane as far as possible from any sample
- New samples close to old samples will then be classified correctly
- Good generalization



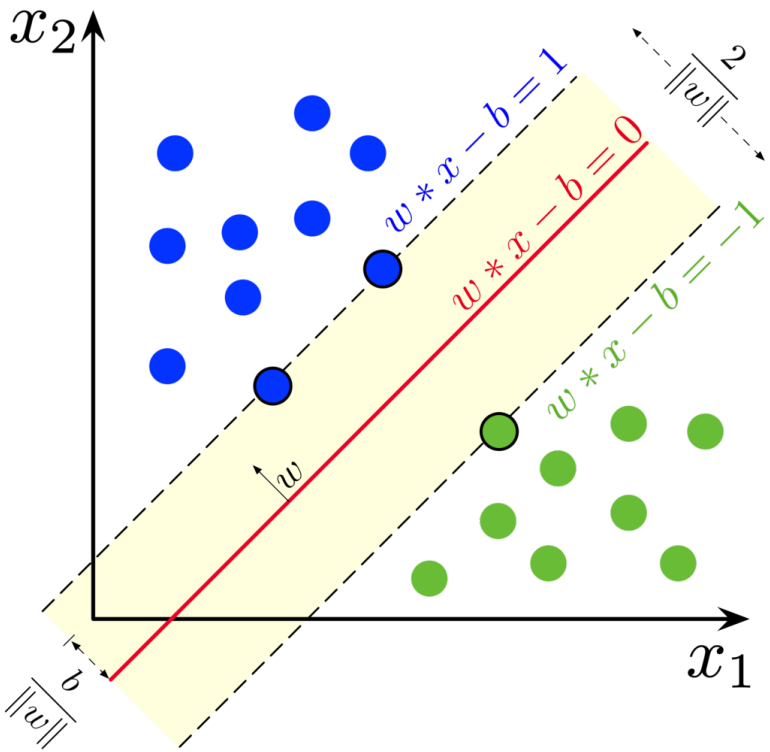
SVM – Linearly Separable Case

- Definition: Margin
  - The margin is twice the absolute value of distance b of the closest example to the hyperplane
- Our goal is to maximize the margin



Linear Discriminant Function

- The general equation for a line is
$$Ax_1 + Bx_2 + C = 0$$
- This equation holds for any point (x_1, x_2) on the line.
- We can write this as a discriminant function:
$$g(\mathbf{x}) = Ax_1 + Bx_2 + C$$
- Then:
  - $g(\boldsymbol{x}) = 0$ for any point $\boldsymbol{x}$ on the line.
  - $g(\boldsymbol{x}) > 0$ for any point above the line
  - $g(\mathbf{x}) < 0$ for any point below the line.



Linear Discriminant Function

$$g(\mathbf{x}) = Ax_1 + Bx_2 + C$$

- A and B are just weights (and C is a bias), so let's write this as:

$$w = \begin{bmatrix} A \\ B \end{bmatrix}, b = C$$

- Now we have

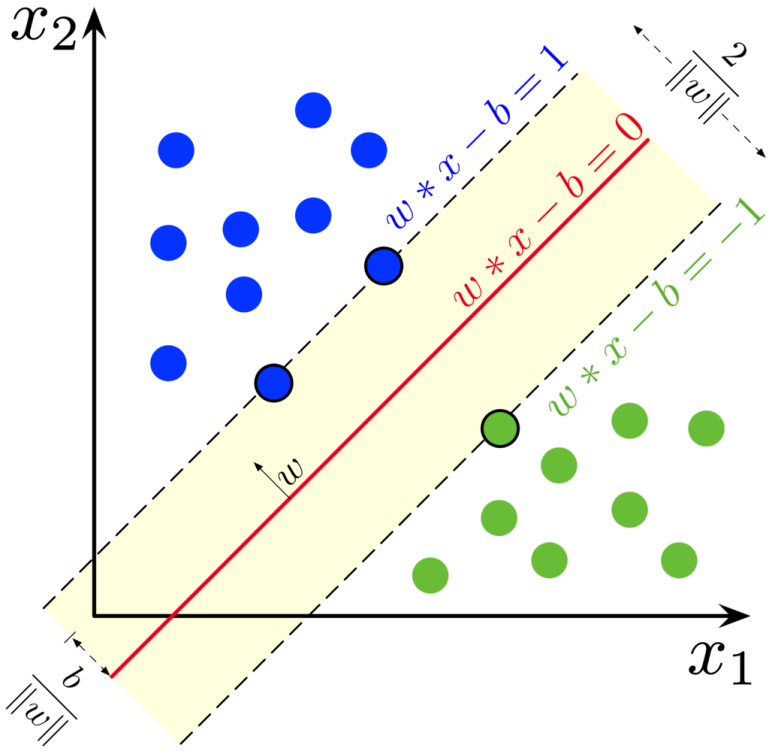
$$g(\boldsymbol{x}) = \boldsymbol{xw} + b$$

- Can we just add a bias feature to our data (like in linear and logistic regression), so the bias is not separate?
- Yes!
- Let

$$\boldsymbol{w} = \begin{bmatrix} A \\ B \\ C \end{bmatrix}, \boldsymbol{x} = [x_1, x_2, 1]$$

- Now

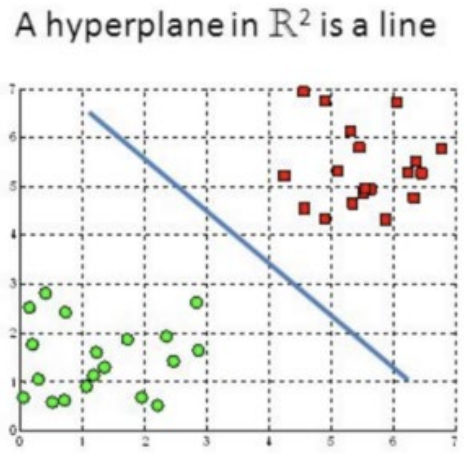
$$g(\boldsymbol{x}) = \boldsymbol{xw}$$

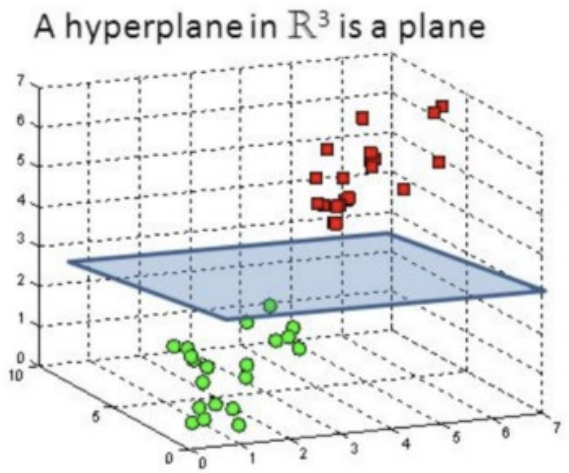


Linear Discriminant Function

$$\boxed{g(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w}}$$

- If $\boldsymbol{x}$ and $\boldsymbol{w}$ have more than three elements, then this is the discriminant function for a hyperplane.
  - If $g(\mathbf{x}) = 0$ then we're on the hyperplane.
  - If $g(\mathbf{x}) > 0$ then we're on one side of the hyperplane
  - If $g(\mathbf{x}) < 0$ then we're on the other side of it.



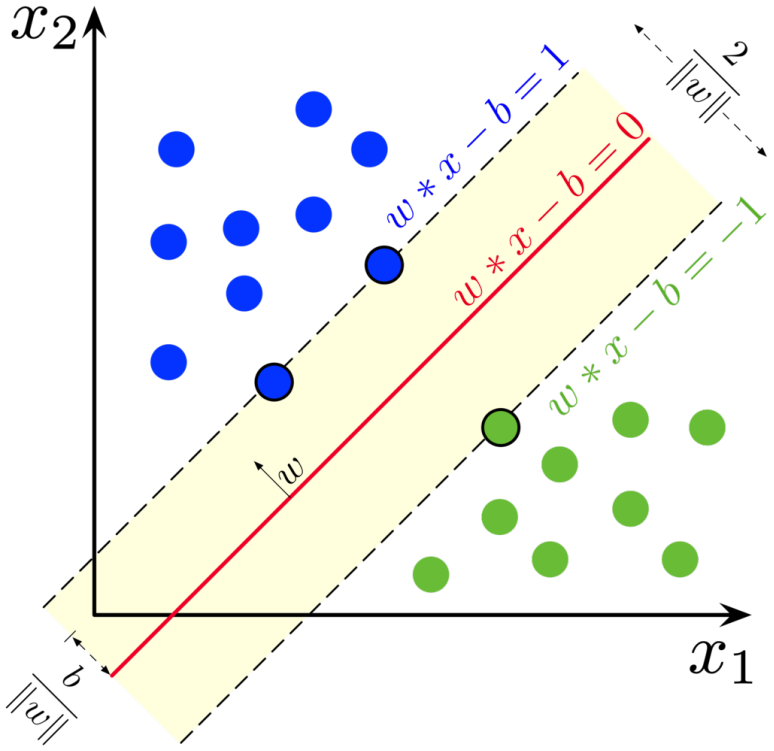


Hyperplanes and Support Vectors

$$g(\boldsymbol{x}) = \boldsymbol{xw}$$

- We want the separating hyperplane to be equidistant from the toughest to classify samples.
- These toughest samples are called the **support vectors,** and we'll designate them as $\boldsymbol{x}_+^*$ and $\boldsymbol{x}_-^*$
- To provide a standard/canonical model we will want our feature vectors to project to be 1 one unit from the hyperplane.

$$\boldsymbol{x}_+^*\boldsymbol{w} = 1$$
$$\boldsymbol{x}_-^*\boldsymbol{w} = -1$$



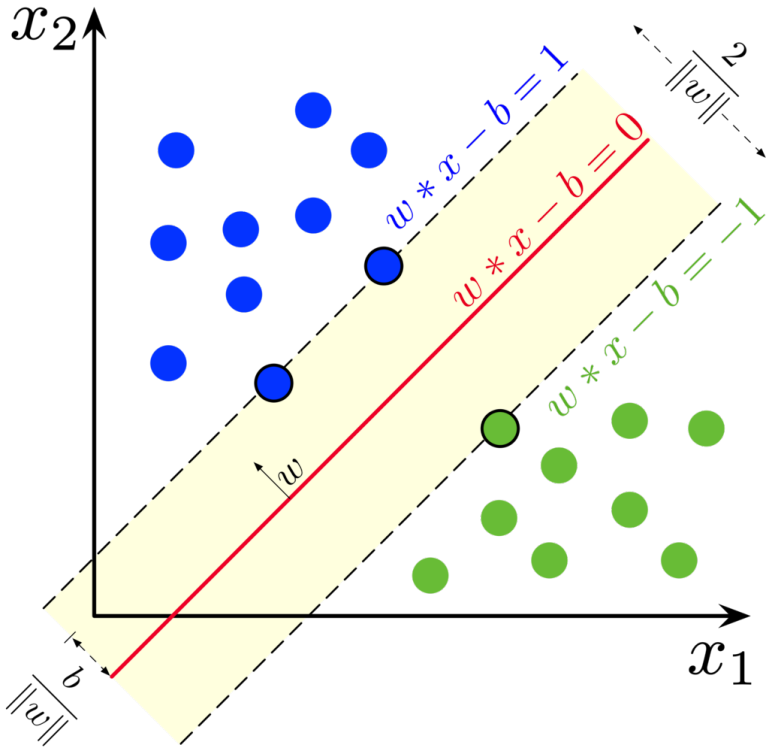
Hyperplanes and Margin

$$g(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w} + b$$

- Geometrically, the distance between the planes containing the support vectors is:

$$\frac{2}{|\boldsymbol{w}|}$$

- We call this the *margin*
  - And it's what we want to maximize!



Margin

$$g(\boldsymbol{x}) = \boldsymbol{xw}$$

$$margin = \frac{2}{|\boldsymbol{w}|}$$

- $|\boldsymbol{w}|$ is just the magnitude of $\boldsymbol{w}$ and thus be computed as:

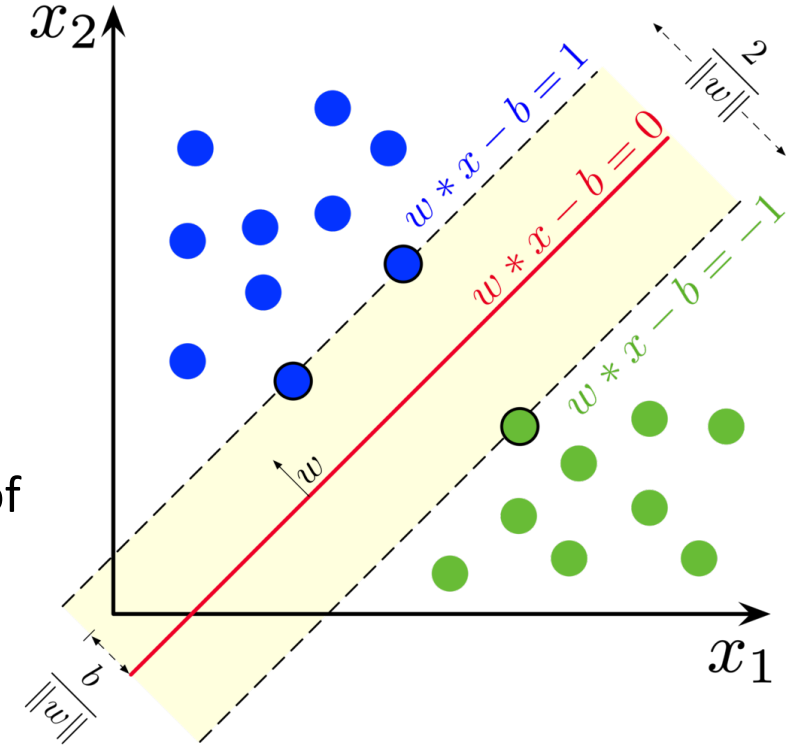
$$margin = \frac{2}{\sqrt{\boldsymbol{w^T w}}}$$

- However, maximizing the square of this is suffice:

$$margin = \frac{2}{\boldsymbol{w^T w}}$$

- And finally, people typically attempt to *minimize* the reciprocal of this:

$$J = \frac{1}{2}\boldsymbol{w^T w}$$

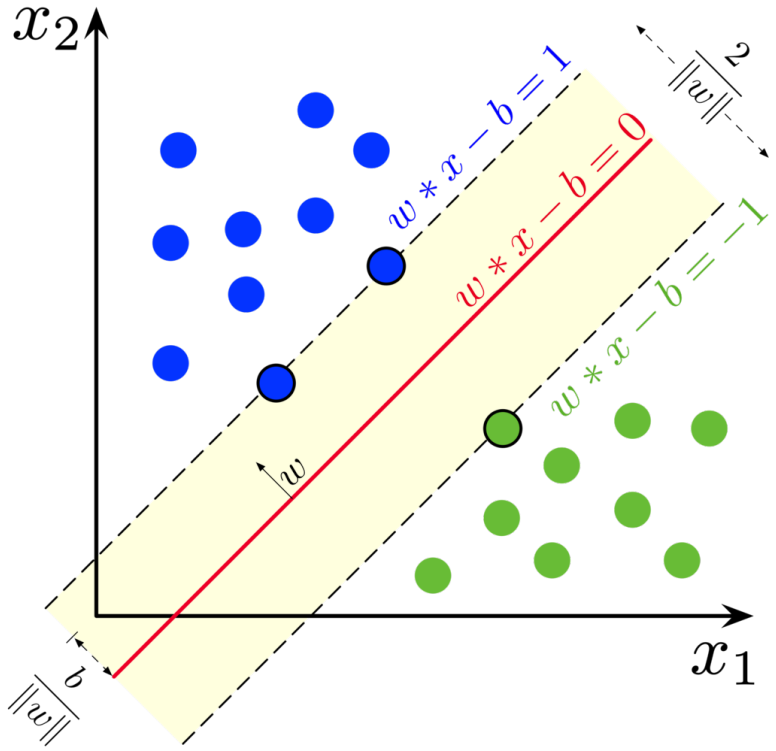


Margin Formula

$$J = \frac{1}{2}\boldsymbol{w^T w}$$

- But what about wanting all the observations ending up on the correct side and outside the margin? i.e

$$\boldsymbol{x_+ w} \geq 1, \qquad \boldsymbol{x\_w} \leq -1$$

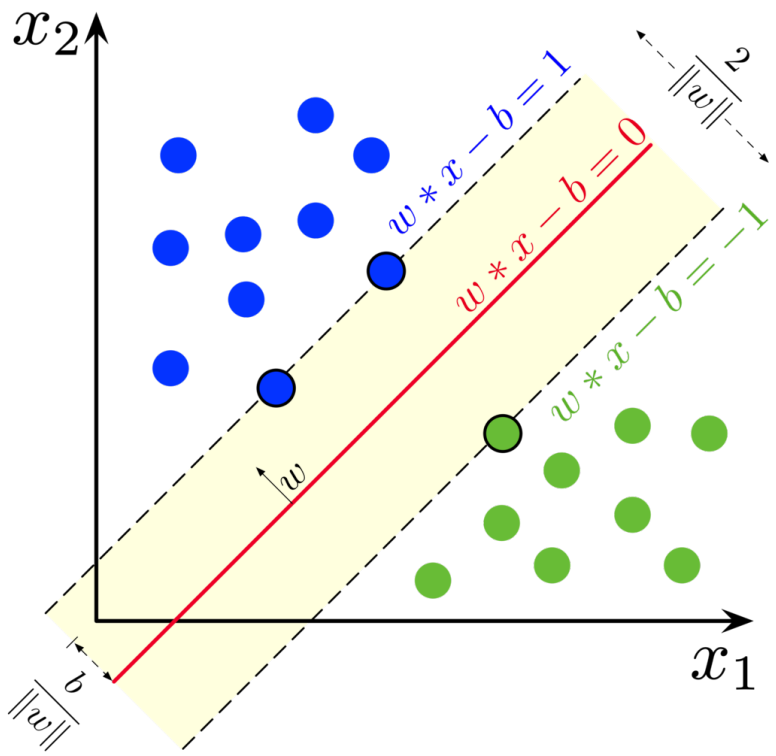


Primal vs Lagrangian Dual Problem

$$J = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t \quad \boldsymbol{x}_+\boldsymbol{w} \geq 1, \qquad \boldsymbol{x}_-\boldsymbol{w} \leq -1$$

- There's two common ways to include these conditions:
  - Primal Form
  - Lagrangian Dual Form.
- We're going to look at the Lagrangian Dual Problem
  - It is more flexible/general and typically provides better results.



Lagrangian Dual Problem

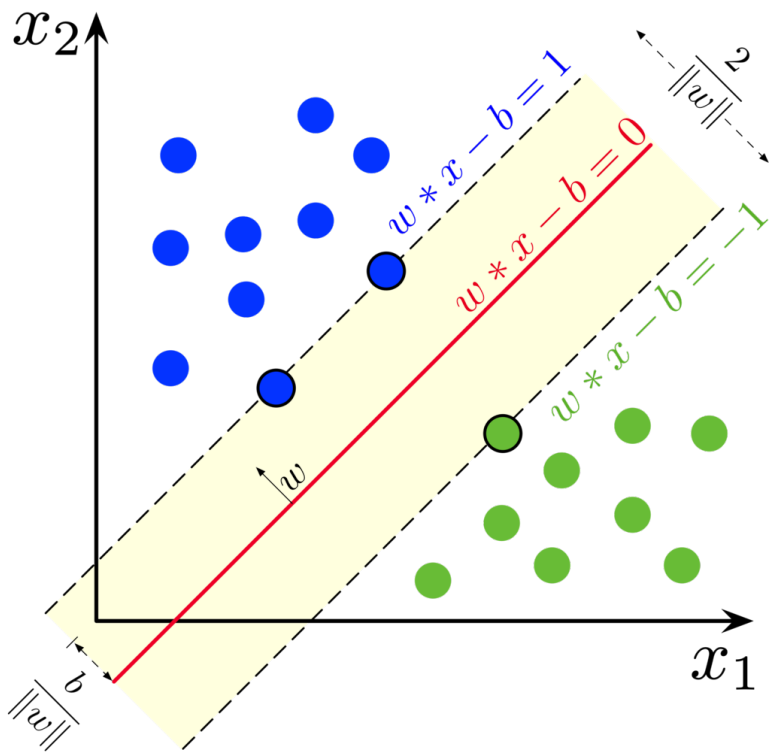
$$J = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$s.t \quad \boldsymbol{x}_+\boldsymbol{w} \geq 1, \qquad \boldsymbol{x}_-\boldsymbol{w} \leq -1$$

- Given that $y \in \{-1, +1\}$, we want $yxw - 1 \geq 0$
  - Support vectors will have $yxw - 1 = 0$
  - Things not at ideal locations will have $yxw - 1 < 0$
- Lagrange multipliers are scalers, one per term, that establish costs (or importance) for different terms.
- Let α_i be the Lagrange multiplier for observation i.
- We then want to maximize

$$J_L = \sum_{i=1}^{N} \left(\alpha_i(\mathrm{Y}_i X_{\boldsymbol{i}}\mathbf{w} - 1)\right)$$

- Note, that we'll also want to ensure that $\alpha_i > 0$



Lagrangian Dual Problem

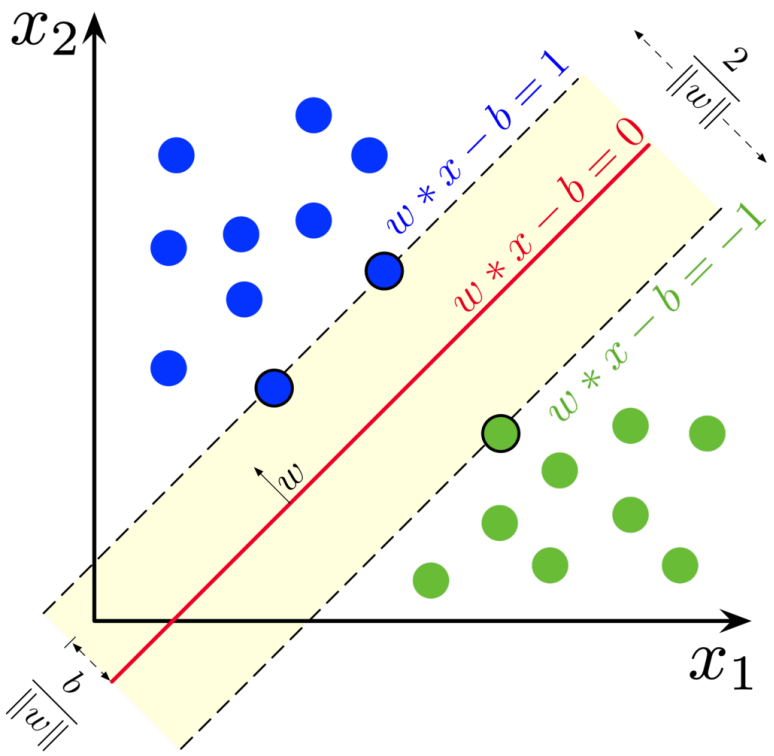
Minimize: $J = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$

Maximize: $J_L = \sum_{i=1}^{N}\left(\alpha_i(\mathrm{Y}_i X_i \mathbf{w} - 1)\right)$

- One thing to minimize (reciprocal of margin), one thing to maximize (things being on the correct side).
  - Hence, the *dual problem!*
- We could write this as a single objective by turning the maximization part into minimization (by negating it):

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \sum_{i=1}^{N}\left(\alpha_i(\mathrm{Y}_i X_i \mathbf{w} - 1)\right)$$

$$s.t\ \alpha_i \geq 0\ \forall i = 1, \ldots, N$$

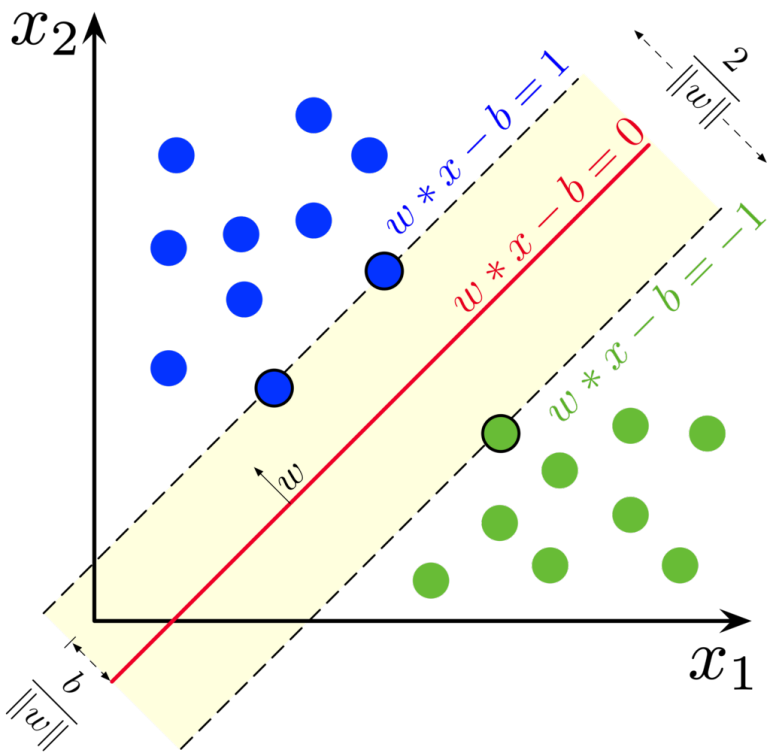


Lagrangian Dual Problem

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \sum_{i=1}^{N}\left(\alpha_i(\mathrm{Y}_i X_i \mathbf{w} - 1)\right)$$

$$s.t\ \alpha_i \geq 0\ \forall i = 1, \ldots, N$$

- This minimization will also want non-support (but correctly located) vectors to have $\alpha_i = 0$.
- Thus:
  - Thing in the wrong location will have $\alpha_i < 0$
    - Which we'll force to $\alpha_i = 0$
  - Things passed the margin (but on the correct side) will get minimized to $\alpha_i = 0$
- Therefore, only our support vectors will have $\alpha_i > 0$!



Lagrangian Dual Problem

$$g(\boldsymbol{x}) = \boldsymbol{xw}$$

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \sum_{i=1}^{N}\left(\alpha_i(\mathrm{Y}_i X_i \mathbf{w} - 1)\right)$$

$$s.t\ \alpha_i \geq 0\ \forall i = 1, \ldots, N$$

- We're going to use calculus to find the values of $\boldsymbol{w}$ and $\boldsymbol{\alpha}$.
- Doing so will be easier if we can write this second term more concisely using linear algebra.
- Let's start by expending the equation out a bit….

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \left(\sum_{i=1}^{N}\alpha_i Y_i X_i w - \sum_{i=1}^{N}\alpha_i\right)$$

Lagrangian Dual Problem

$$\boxed{g(\boldsymbol{x}) = \boldsymbol{xw}}$$

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \left(\sum_{i=1}^{N} \alpha_i Y_i X_i w - \sum_{i=1}^{N} \alpha_i\right)$$

$$s.t\ \alpha_i \geq 0\ \forall i = 1, \ldots, N$$

- Now to eliminate those summations via matrix multiplications…
- We can do this as:

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \left(\boldsymbol{\alpha}^T diag(Y) X \boldsymbol{w} - trace\left(diag(\boldsymbol{\alpha})\right)\right)$$

  - Recall that the trace operator is sum of elements on the diagonal.

Finding Hyperplane

$$J = \frac{1}{2}\boldsymbol{w^T w} - \left(\boldsymbol{\alpha}^T diag(Y) X \boldsymbol{w} - trace\left(diag(\boldsymbol{\alpha})\right)\right)$$
$$s.t\ \alpha_i \geq 0\ \forall i = 1, \ldots, N$$

- So, how do we use calculus to find $\boldsymbol{w}$ and $\boldsymbol{\alpha}$?
- This is most easily done via a combination of direct and gradient based solutions.
- We'll use a direct solution to find $\boldsymbol{w}$ in terms of $\boldsymbol{\alpha}$, find $\boldsymbol{\alpha}$ via a gradient-based approach
  - The gradient approach for α is needed to impose the $\alpha_i \geq 0$ constraint.

Finding Hyperplane

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \left(\boldsymbol{\alpha}^T diag(Y)X\boldsymbol{w} - trace\left(diag(\boldsymbol{\alpha})\right)\right)$$
$$s.t\ \alpha_i \geq 0\ \forall i = 1, \ldots, N$$

- To find the $\boldsymbol{w}$, we'll take the derivative with respect to it
- Doing this we get:

$$\frac{\partial J}{\partial \boldsymbol{w}} = \boldsymbol{w} - (\boldsymbol{\alpha}^T diag(Y)X)^T = \boldsymbol{w} - X^T diag(Y)\boldsymbol{\alpha}$$

- Set this to zero and solve for $\mathbf{w}$

$$\mathbf{w} = X^T diag(Y)\boldsymbol{\alpha}$$

Finding Hyperplane

$$J = \frac{1}{2}\boldsymbol{w^T}\mathbf{w} - \left(\boldsymbol{\alpha}^T diag(Y)X\boldsymbol{w} + b\boldsymbol{\alpha^T Y} - trace\big(diag(\boldsymbol{\alpha})\big)\right)$$
$$\mathbf{w} = X^T diag(Y)\boldsymbol{\alpha}$$
$$s.t\ \alpha_i \geq 0\ \forall i = 1, \dots, N$$

- To find $\boldsymbol{\alpha}$, let's establish a gradient rule, $\frac{\partial J}{\partial \boldsymbol{\alpha}}$
- First let's do a substitution to get J solely in terms of $\boldsymbol{\alpha}$

$$J = \frac{1}{2}(X^T diag(Y)\boldsymbol{\alpha})^{\boldsymbol{T}}(X^T diag(Y)\boldsymbol{\alpha}) - \left(\boldsymbol{\alpha}^T diag(Y)XX^T diag(Y)\boldsymbol{\alpha} - trace\big(diag(\boldsymbol{\alpha})\big)\right)$$

- Simplifying ….

$$J = trace\big(diag(\boldsymbol{\alpha})\big) - \frac{1}{2}\boldsymbol{\alpha}^T diag(Y)XX^T diag(Y)\boldsymbol{\alpha}$$

- Now $\frac{\partial J}{\partial \boldsymbol{\alpha}}$…

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\big(size(Y)\big) - diag(Y)XX^T diag(Y)\boldsymbol{\alpha}$$

Finding Hyperplane

$$J = \frac{1}{2}\boldsymbol{w^T w} - \left(\boldsymbol{\alpha}^T diag(Y)X\boldsymbol{w} - trace\left(diag(\boldsymbol{\alpha})\right)\right)$$
$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\left(size(Y)\right) - diag(Y)XX^T diag(Y)\boldsymbol{\alpha}$$
$$\mathbf{w} = X^T diag(Y)\boldsymbol{\alpha}$$
$$s.t\ \alpha_i \geq 0\ \forall i = 1, \dots, N$$

- A few caveats….
  1. $\frac{\partial J}{\partial \boldsymbol{\alpha}}$ is actually the partial of the part of J that we want to **maximize.**
  2. During gradient **ascent** we'll need to set any $\alpha_i < 0$ to zero
- Pseudocode!
  - Initialize $\alpha = 0$
  - While minimal change in α
    - $\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\left(size(Y)\right) - diag(Y)XX^T diag(Y)\boldsymbol{\alpha}$
    - $\alpha \mathrel{+}= \eta \frac{\partial J}{\partial \alpha}$
    - $\alpha(\alpha < 0) = 0$
  - $\alpha > 0$ are our support vectors!
  - $w = X^T diag(Y)\alpha$

Simple Example

- Given data

$$X = \begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

- It is typically beneficial to zscore our data in SVMs since we're computing the dot product between observations.
- And we'll also want to add a bias feature.
- So, our new observations are:

$$X = \begin{bmatrix} -0.866 & -0.866 & 1 \\ -0.866 & 0.866 & 1 \\ 0.866 & -0.866 & 1 \\ 0.866 & 0.866 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Simple Example

$$X = \begin{bmatrix} -0.866 & -0.866 & 1 \\ -0.866 & 0.866 & 1 \\ 0.866 & -0.866 & 1 \\ 0.866 & 0.866 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

- Doing gradient ascent to learn α, we arrive at:

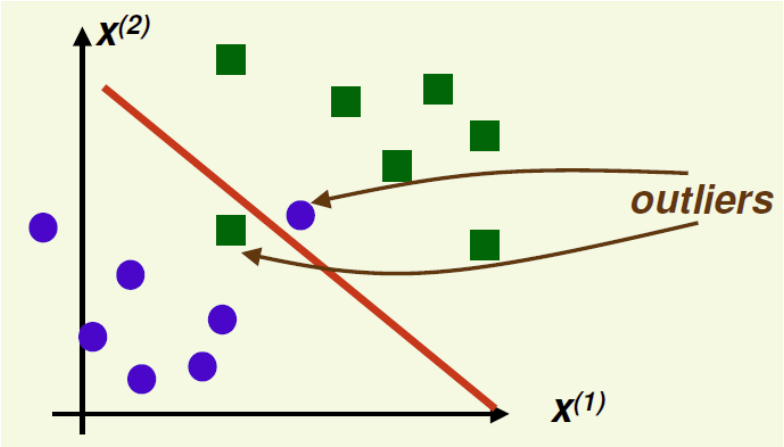
$$\alpha = \begin{bmatrix} 0.325 \\ 0.325 \\ 0.325 \\ 0.325 \end{bmatrix}$$

- All support vectors!
- And w as

$$w = X^T diag(Y)\alpha = \begin{bmatrix} 1.1547 \\ 0 \\ 0 \end{bmatrix}$$

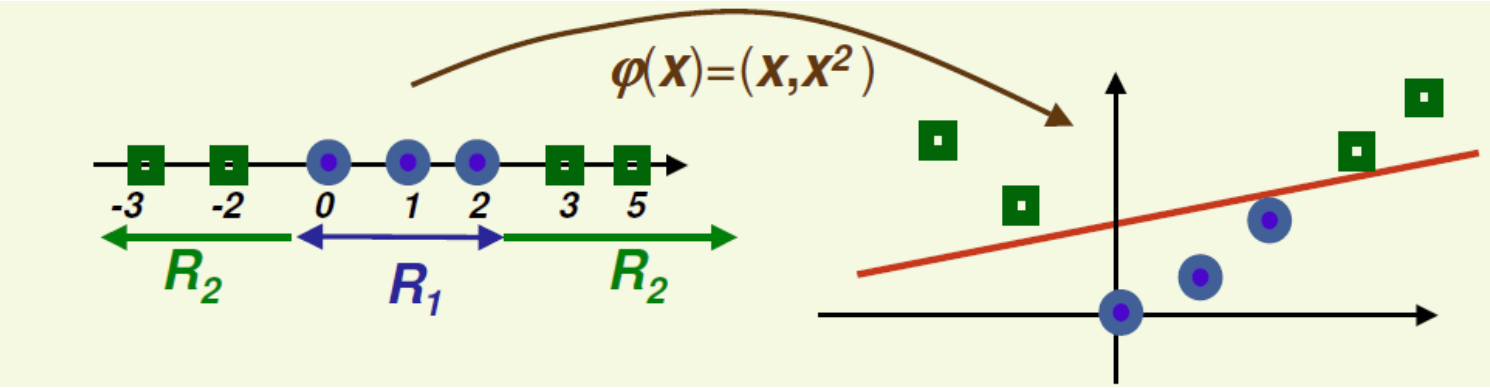
Non-Separable Situations

- Data is most likely to be not linearly separable
  - But linear classifier may still be appropriate
- Idea:
  - Move the data to a *higher* dimensionality where it may be separable



Mapping Functions

- A *mapping function*, $\phi(\boldsymbol{x})$, takes use from the feature space of $\boldsymbol{x}$ to some higher dimensional feature space
- A higher dimensional space is more easily linearly separable.
  - But don't forget about some of the issues with too many features!
- For instance, imagine the mapping function $\phi(\boldsymbol{x}) = [x_1, x_1^2]$ on the data below:



Non-Separable SVM

- So, where in our equations do we need the mapped observations?

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\big(size(Y)\big) - diag(Y)XX^T diag(Y)\boldsymbol{\alpha}$$

$$\mathbf{w} = X^T diag(Y)\boldsymbol{\alpha}$$

Non-Separable SVM

- In our new space this is

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\big(size(Y)\big) - diag(Y)\phi(X)\phi(X)^T diag(Y)\boldsymbol{\alpha}$$

$$\mathbf{w} = \phi(X)^T diag(Y)\boldsymbol{\alpha}$$

- We can make predictions as:

$$g(\boldsymbol{x}) = \phi(\boldsymbol{x})\boldsymbol{w} = \phi(\boldsymbol{x})\phi(X)^T diag(Y)\boldsymbol{\alpha}$$

Cosine Similarty

$$\boldsymbol{\alpha} = \left(diag(Y)XX^T diag(Y)\right)^{-1}\text{ones}(\text{size}(\boldsymbol{\alpha}), 1)$$

- Note that the two places that we need $\phi(\boldsymbol{x})$ involve $\phi(x)\phi(x)^T$:

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\left(size(Y)\right) - diag(Y)\phi(X)\phi(X)^T diag(Y)\boldsymbol{\alpha}$$

$$g(\boldsymbol{x}) = \phi(\boldsymbol{x})\phi(X)^T diag(Y)\boldsymbol{\alpha} + b$$

- The computation $\phi(\boldsymbol{a})\phi(\boldsymbol{b})^T$ is known as the *cosine similarity* between observation $\boldsymbol{a}$ and $\boldsymbol{b}$.
- There is a group of functions, known as *kernel functions*, that can compute $\phi(\boldsymbol{a})\phi(\boldsymbol{b})^T$ without explicitly mapping to the higher dimensional space.

Kernels

- Commonly used kernels include:
  - Linear: $\kappa(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}\boldsymbol{b}^T$
  - Polynomial (degree is p): $\kappa(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}\boldsymbol{b}^T + 1)^p$
  - Radial Basis Function (RBF): $\kappa(\boldsymbol{a}, \boldsymbol{b}) = e^{-\frac{(\boldsymbol{a}-\boldsymbol{b})(\boldsymbol{a}-\boldsymbol{b})^T}{2\sigma^2}}$

Example: The Kernel Trick

- As an example, let's show that using a quadratic kernel, $\kappa(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}\boldsymbol{b}^T + 1)^2$, when a and b are 1-D feature vectors is equivalent to $\kappa(\boldsymbol{a}, \boldsymbol{b}) = \phi(\boldsymbol{a})\phi(\boldsymbol{b})^T$ where $\phi(\boldsymbol{x}) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1 & 1 \end{bmatrix}$?

Kernels

- Quadratic Polynomial Kernel:

$$\kappa(\boldsymbol{a}, \boldsymbol{b}) = \left(\boldsymbol{a}\boldsymbol{b}^{\boldsymbol{T}} + 1\right)^2$$

- Observational data:

$$\boldsymbol{x} = [x_1]$$

- Therefore, $\kappa(\boldsymbol{a}, \boldsymbol{b}) = \left(\boldsymbol{a}\boldsymbol{b}^{\boldsymbol{T}} + 1\right)^2 = (a_1 b_1 + 1)^2$

$$= (a_1 b_1)^2 + 2(a_1 b_1) + 1$$

$$= a_1^2 b_1^2 + 2a_1 b_1 + 1$$

- What is $\phi(\boldsymbol{x})$ in order to write:

$$\kappa(\boldsymbol{a}, \boldsymbol{b}) = a_1^2 b_1^2 + 2a_1 b_1 + 1 = \phi(\boldsymbol{a})\phi(\boldsymbol{b})^T$$

Kernel Trick

$$\kappa(\boldsymbol{a}, \boldsymbol{b}) = \left[a_1^2, \sqrt{2}a_1, 1\right] \begin{bmatrix} b_1^2 \\ \sqrt{2}b_1 \\ 1 \end{bmatrix}$$

- Therefore

$$\phi(\boldsymbol{x}) = \begin{bmatrix} x_1^2 & \sqrt{2}x_1 & 1 \end{bmatrix}$$

- We just showed that using the polynomial kernel of degree two on observations with a single feature is equivalent to compute the cosine similarity on observations in 3D space!

Kernels

- Commonly used kernels include:
  - Linear: $\kappa(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}\boldsymbol{b}^T$
  - Polynomial (degree is p): $\kappa(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}\boldsymbol{b}^T + 1)^p$
  - Radial Basis Function (RBF): $\kappa(\boldsymbol{a}, \boldsymbol{b}) = e^{-\frac{(\boldsymbol{a}-\boldsymbol{b})(\boldsymbol{a}-\boldsymbol{b})^T}{2\sigma^2}}$
- Which should we use?
  - Start with linear, and if underfitting, move up to quadratic, etc…
  - Try to find one that give you the balance between over and underfitting.
  - Note: RBF kernel does the computations in the highest possible dimensionality.

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\big(size(Y)\big) - diag(Y)XX^T diag(Y)\boldsymbol{\alpha}$$
$$g(\boldsymbol{x}) = xX^T diag(Y)\boldsymbol{\alpha}$$

Non-Separable SVM

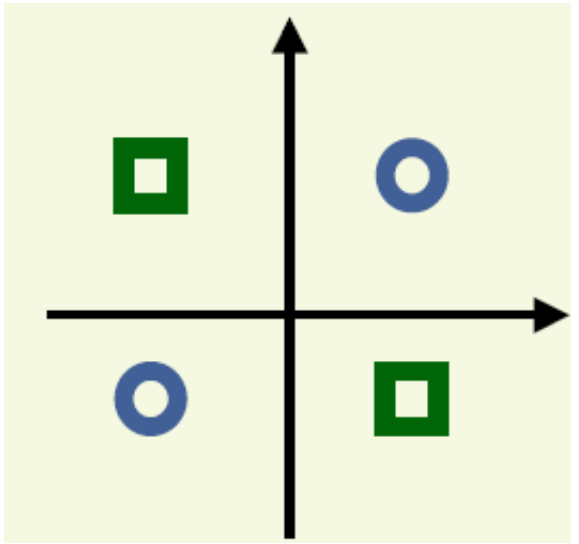
- Let $K(A, B)$ be a pairwise similarity matrix comparing the rows of A and B using a kernel function $\kappa(a, b)$.
  - $K(A, B)$ will be of size of size $M \times N$ where M is the number of rows (observations) in A and B is the number of rows (observations) in B
- We can now compute the gradient of α using the kernel trick as:

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\big(size(Y)\big) - diag(Y)\kappa(X, X)diag(Y)\boldsymbol{\alpha}$$

- And then apply the discriminant to an observation (without actually solving for $\boldsymbol{w}$) as:

$$g(\boldsymbol{x}) = \boldsymbol{K(x, X)}diag(Y)\boldsymbol{\alpha}$$

SVM Example: XOR



- Class 1 (positive): X1=[1,-1], X2=[-1,1]
- Class 2 (negative): X3=[1,1], X4=[-1,-1]
- Therefore, our data set is:

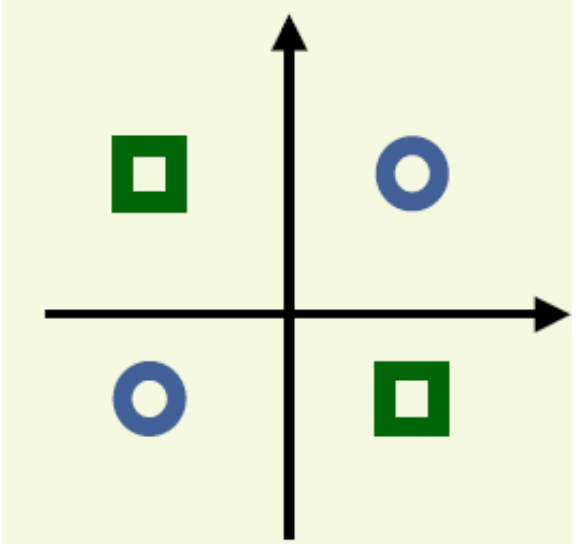
$$X = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & 1 \\ -1 & -1 \end{bmatrix}, Y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

- We will once again prepare our data by zscoring and adding a bias feature:

$$X \rightarrow \begin{bmatrix} 0.866 & -0.866 & 1 \\ -0.866 & 0.866 & 1 \\ 0.866 & 0.866 & 1 \\ -0.866 & -0.866 & 1 \end{bmatrix}$$

SVM Example: XOR

- In this current space our data is not separable!
- Going through the gradient ascent to learn α will never terminate! ☹

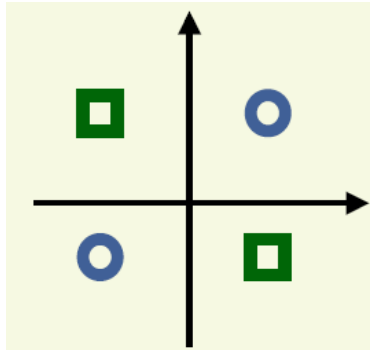


SVM Example: XOR

$$\frac{\partial J}{\partial \boldsymbol{\alpha}} = ones\big(size(Y)\big) - diag(Y)XX^T diag(Y)\boldsymbol{\alpha}$$

- So, we could explicitly do everything in a higher space, if we know of a good mapping function.
- Instead let's use our kernel trick and instead of computing $\phi(\mathbf{X}_i)^T\phi(\boldsymbol{X}_j)$ in our equation for J let's use $\kappa(\boldsymbol{X}_i, \boldsymbol{X}_j)$!
- If we choose to use a quadratic kernel, $\kappa(\mathbf{a}, \mathbf{b}) = (\boldsymbol{a}\boldsymbol{b}^T + 1)^2$ then we'd eventually get to:

$$\boldsymbol{\alpha} = \begin{bmatrix} 0.2222 \\ 0.2222 \\ 0.2222 \\ 0.2222 \end{bmatrix}$$



- Now they're all support vectors!

SVM Example: XOR

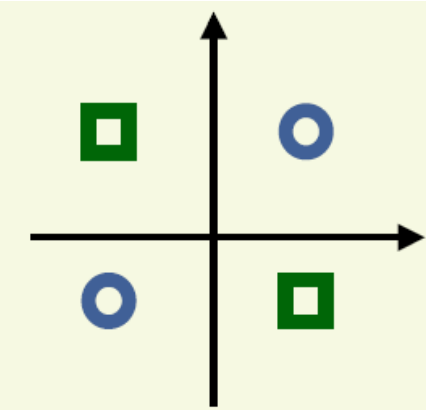
$$X \rightarrow \begin{bmatrix} 0.866 & -0.866 & 1 \\ -0.866 & 0.866 & 1 \\ 0.866 & 0.866 & 1 \\ -0.866 & -0.866 & 1 \end{bmatrix} \quad Y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 0.2222$$

- Let's do a sanity check!
  - Predict the value for $x = [1, -1]$
- Thanks to the kernel trick we just need to compute:

$$g(\mathbf{x}) = \kappa(\boldsymbol{x}, X) diag(Y) \boldsymbol{\alpha}$$

- Recall we chose $\kappa(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}\boldsymbol{b}^T + 1)^2$
- Therefore



$$g([1, -1]) = [12.25, 0.25, 4, 4] \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0.222 \\ 0.222 \\ 0.222 \\ 0.222 \end{bmatrix} \right) = 1$$

Resources

- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://www.saedsayad.com/support_vector_machine.htm
- https://svivek.com/teaching/lectures/slides/svm/svm-sgd.pdf
- https://www.youtube.com/watch?v=_PwhiWxHK8o
- https://www.cs.rpi.edu/~stewart/lec23-post/svms.pdf
- https://kuleshov-group.github.io/aml-book/contents/lecture13-svm-dual.html
- https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf
- https://www.cs.rpi.edu/~stewart/lec23-post/svms.pdf
- https://machinelearningmastery.com/method-of-lagrange-multipliers-the-theory-behind-support-vector-machines-part-1-the-separable-case/