Support Vector Machines

David Quigley
CSCI 5622
2021 Fall

Course Logistics

- Project Pitch Feedback – Did you get your Projects to Review?
- Exam 1 – October ~~20th~~ 21st
- Problem Set 3 – Due ~~October 25th~~ Early Nov
  - It still has~~n't yet~~ been released, so the due date ~~will need to be~~ is adjusted
  - This ~~will~~ impacted the project milestone 3 deadline (which is *tiny*, but still don't want a 0 - 3 day gap)
  - This will probably only minorly impact "Problem Set 4"
    - It's going to be a very different kind of activity

Classification – Weeks 1 – 7*

- K Nearest Neighbors
  - Manhattan, Euclidian Distance
- Naïve Bayes
  - Class-Conditional, Prior, Evidence
- Trees & Ensemble Methods
  - Splitting, Pruning
  - Random Forests
  - AdaBoost
- Logistic Regression
  - Idea, Proof, Implementation
    - Gradient Descent
    - Stochastic & Mini-Batch Gradient Descent
- SVM
  - Idea, Proof, Implementation
    - Sequential Minimal Optimizer
    - Kernel Trick

Regression

- Linear Regression
  - Idea, Proof, Implementation
    - Minimizing MSE / LSS

Dimensionality Reduction

- Select-K-Best / Best Subset
- Stepwise selection
- Stagewise Selection

- Lasso & Ridge Regression
    - Idea, Proof, Implementation
        - Within Linear Regression

Skills

- Evaluation
  - Accuracy
  - Types of Errors & Confusion Matrix
  - ROC Curves
  - R^2
- Training
  - Train, Test, Validation (Hold-Out)
  - Cross-Fold Validation
- Multi-Class
  - One vs. All
  - All-Pairs (One vs. Another)
- Data Transformation
  - Encoding, Binning, Smoothing, Binarization / One-Hot Encoding, etc.
- Scaling & Normalization
  - Min-Max, Mean & Std. Dev

Concerns

- Impact of Errors
    - False Positive vs. False Negative
- Overfitting
    - High Variance
- Underfitting
    - High Bias
- Curse of Dimensionality

?

- When to use what classifiers?

Exam 1 – Logistics

- Delivered via Canvas
- Open on Oct. 2~~0~~1, Approximately all day
    - (Probably not 12:01AM to 11:59PM, but *well* beyond the class period, including morning through evening times)
- It will be *Timed* to only allow you 75 ~~50~~ minutes to complete
    - Those with 1.5X dispensation, etc. will be accommodated via Canvas.
- Exam *support* will be provided in-lecture and via Zoom *during lecture time*.
    - A slightly longer window of support (i.e. to accommodate extra time students) will be via Zoom, timing TBD.

Exam 1 – Format

- It is considered “open-resources”, but “individual”
  - Previous work, notes, Piazza discussions, textbook, etc. are all fair game
  - (Piazza will be in “read-only” mode for the day, so don’t be scared someone will post solutions there)
  - In general, follow the same plagiarism and honor code policies as any other assignment
- Conceptual Questions
  - Closed-ended (multiple choice, matching, word bank, etc.)
  - Open-ended short answer with clear guidance (name one reason we would choose X over Y)
- Problem Solving
  - Can be done with a 4-function calculator and some paper in a ~~50~~ 75-minute exam
  - No coding or code analysis

Continued Milestone 2 Videos

- 19
- 20
- 21
- 22
- 23
- 24
- 26
- 27
- 0
- 10

Best Decision Boundary

$\text{argmax}_{w,\,b}\ (1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Optimize to find w, b

Best Decision Boundary

$\text{argmax}_{w,\,b}\ (1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Optimize to find w, b

But it's not differentiable! (i.e. we can't find a gradient vector)

Best Decision Boundary

$\mathrm{argmax}_{w,\, b}\ (1\ /\ ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\mathrm{argmin}_{w,\, b}\ (||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,...,m\}$

Best Decision Boundary

$\text{argmax}_{w,\,b}\ (1\ /\ ||w||)$ such that $y_i(w^Tx_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||)$ such that $y_i(w^Tx_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

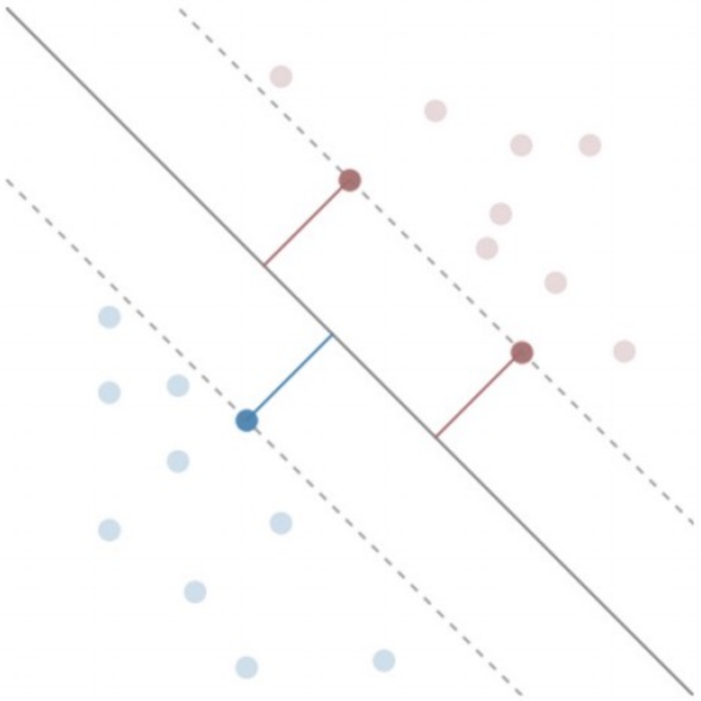
Best Decision Boundary

Support Vectors

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

DB is unaffected by scale

$$(2w)^T x + 2b = 0$$



Best Decision Boundary

END

$\text{argmax}_{w,\,b}\ (1 / ||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,\dots,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,\dots,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||^2)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,\dots,m\}$

Convex + Differentiable

Requires a Convex quadratic program

Underlying Math – Finding our Maximum Margin

Generalizing – adding Lagrange Multipliers

$\min_w f(w)$ such that $g_i(w) <= 0$ for $i = \{1,2,\ldots,m\}$

Generalizing – adding Lagrange Multipliers

$\min_w f(w)$

such that $g_i(w) <= 0$ for $i = \{1,2,\ldots,m\}$

The Lagrangian of this function

$L(w, \alpha) = f(w) + \Sigma_{i=1\ldots m} (\alpha_i g_i(w))$

such that $\alpha_i >= 0$

Generalizing – adding Lagrange Multipliers

$\min_w f(w)$

such that $g_i(w) <= 0$ for $i = \{1,2,\ldots,m\}$

The Lagrangian of this function

$L(w, \alpha) = f(w) + \Sigma_{i=1\ldots m} (\alpha_i g_i(w))$

such that $\alpha_i >= 0$

$\mathrm{Max}_\alpha\, L(w, \alpha) = f(w)^*$

* Holds if f(w) is feasible, otherwise it goes to infinity

Generalizing – adding Lagrange Multipliers

$\min_w \max_\alpha \mathcal{L}(w, \alpha) = \min_w \max_\alpha f(w) + \Sigma_{i=1\ldots m} (\alpha_i g_i(w))$

such that $\alpha_i >= 0$

Problem is convex

Global minimum means

$\nabla_w \mathcal{L}(w, \alpha) = 0$ and $\nabla_\alpha \mathcal{L}(w, \alpha) = 0$

Generalizing – adding Lagrange Multipliers

$\min_w \max_\alpha L(w, \alpha) = \min_w \max_\alpha f(w) + \Sigma_{i=1...m} (\alpha_i g_i(w))$

such that $\alpha_i >= 0$

Problem is convex

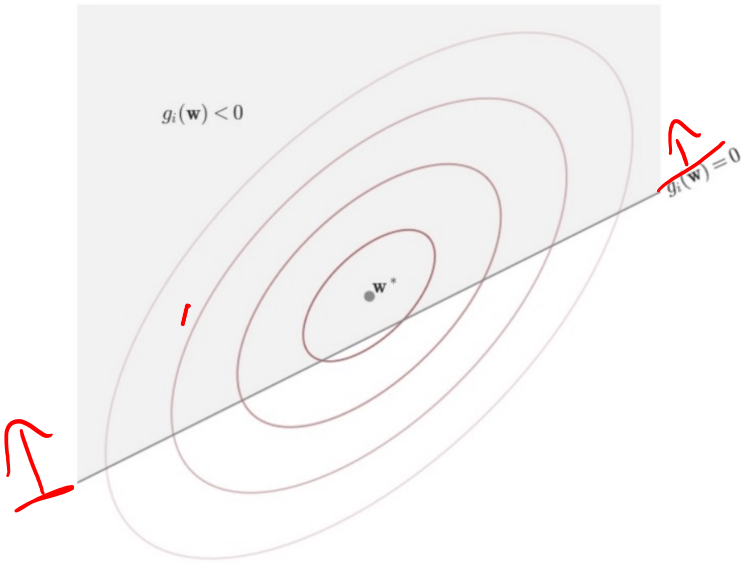
Global minimum means

$\nabla_w L(w, \alpha) = 0$ and $\nabla_\alpha L(w, \alpha) = 0$

$\min_w f(w)$

such that $g_i(w) <= 0$ for $i = \{1,2,...,m\}$

Visualizing f(x) and g(x)

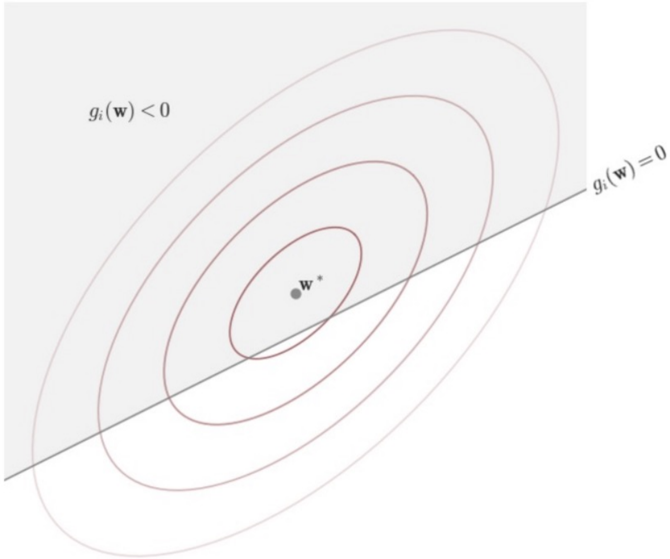


Visualizing f(x) and g(x)

f(x) is minimized within the $g_i(x)$ constraint, $g_i(x)$ is *inactive*

we can "ignore" our α, $\alpha_i = 0$

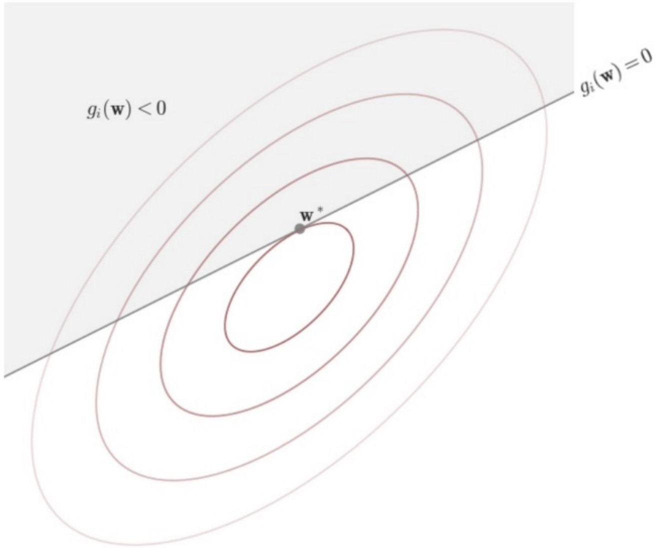
$\nabla_w f(w) = 0$



Visualizing f(x) and g(x)

$f(x)$ is not minimized within the $g_i(x)$ constraint, $g_i(x)$ is *active*

$\alpha_i > 0$



Complementary Slackness

$\alpha_i g_i(w^*) = 0$ for $I = \{1,2,\ldots,m\}$

if $\alpha_i = 0$ if we are not on the boundary of g_i

else $\alpha_i > 0$

The Dual Formulation

$\min_w \max_\alpha L(w, \alpha)$

such that $\alpha_i g_i(w) = 0$

and $\alpha_i >= 0$

$\max_\alpha \min_w L(w, \alpha) <= \min_w \max_\alpha L(w, \alpha)$

DUAL **PRIMAL**

Our New Formulation

$\text{argmin}_{w,\,b}\ (||w||^2)$ such that $y_i(w^T x_i + b) >= 1$ for $i = \{1,2,\ldots,m\}$

$\min_w \max_\alpha \mathcal{L}(w, \alpha) = \min_w \max_\alpha f(w) + \Sigma_{i=1\ldots m}\ (\alpha_i g_i(w))$ - PRIMAL

Becomes

$L_D = \max_\alpha \min_{w,b} (\overbrace{\tfrac{1}{2}||w||^2} + \overbrace{\Sigma_{i=1\ldots m}\ (\alpha_i\,(1 - y_i(w^T x_i + b)))})$

DUAL

Lagragian Form of Maximizing the Margin

$L_D = \max_\alpha \underbrace{\min_{w,b} \left(\tfrac{1}{2}||w||^2 + \Sigma_{i=1\ldots m} (\alpha_i (1 - y_i(w^T x_i + b)))\right)}$

We want the derivative of w & b to be 0

Now that $min_{w,b}$ is the inner optimization, we can do this!

$w = \Sigma_i \alpha_i y_i x_i$

$0 = \Sigma_i \alpha_i y_i$

Lagragian Form of Maximizing the Margin

$L_D = \max_\alpha \min_{w,b} (½||w||^2 + \Sigma_{i=1...m} (\alpha_i (1 - y_i(w^T x_i + b))))$

Apply complementary slackness & set derivative w, b to 0

Karush-Kuhn-Tucker Conditions

$\max_\alpha (L_D) = \max_\alpha (\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$

$\alpha_i > 0 \text{ if } y_i(w^T x_i + b) = 1$

$\alpha_i = 0 \text{ if } y_i(w^T x_i + b) < 1$

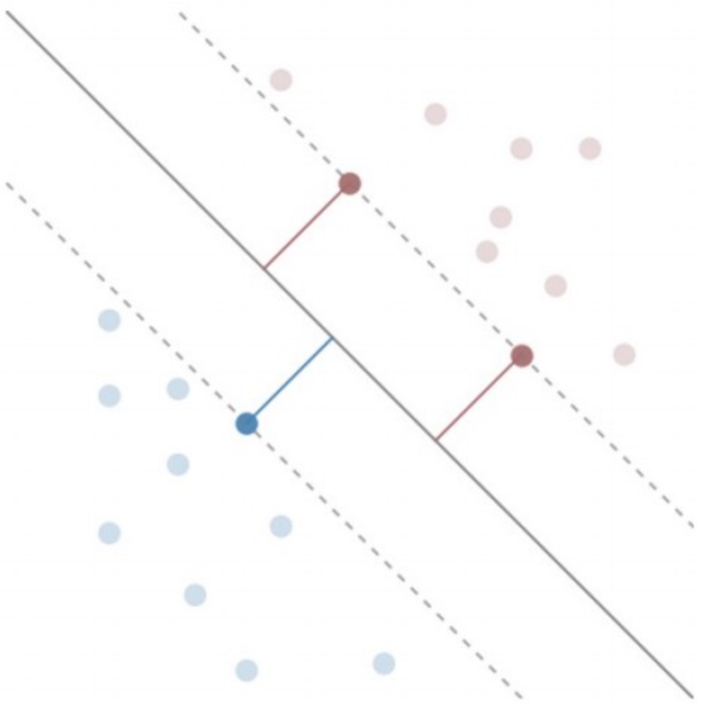
Maximizing our Margin

END

$$\max_\alpha \left(\Sigma_i \alpha_i - \tfrac{1}{2} \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j\right)$$

$$\alpha_i > 0 \text{ if } y_i(w^T x_i + b) = 1$$

$$\alpha_i = 0 \text{ if } y_i(w^T x_i + b) < 1$$



Thursday

Course Logistics

- ~~Project Pitch Feedback – Did you get your Projects to Review?~~
- Exam 1 – October 21st
  - Just released a "Practice Exam" that should show off the formatting, etc.
- Problem Set 3 – Due November 2

Participation: Exam 1 Review Poll

Best Decision Boundary

END

$\text{argmax}_{w,\,b}\ (1\ /\ ||w||)$ such that $y_i(w^Tx_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||)$ such that $y_i(w^Tx_i + b) >= 1$ for $i = \{1,2,...,m\}$

Not Differentiable

$\text{argmin}_{w,\,b}\ (||w||^2)$ such that $y_i(w^Tx_i + b) >= 1$ for $i = \{1,2,...,m\}$

Convex + Differentiable

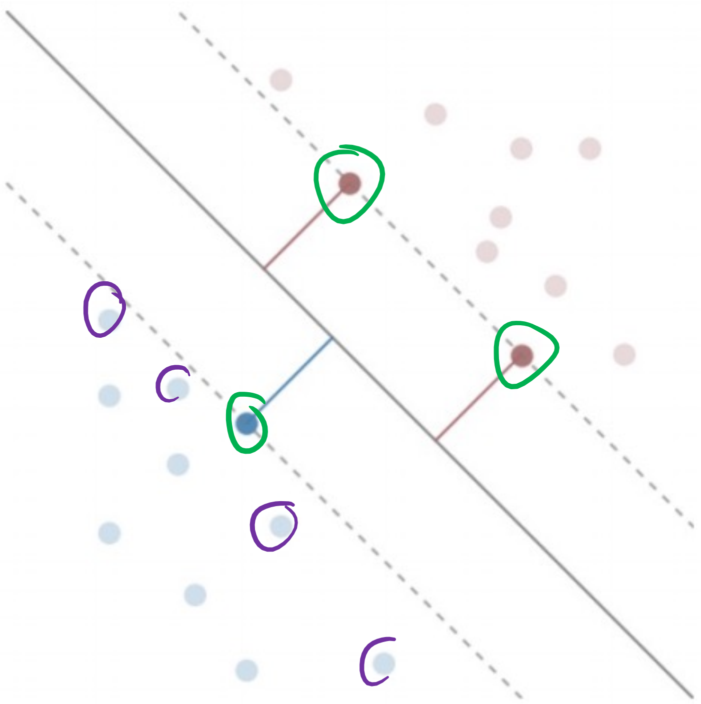
Requires a Convex quadratic program

END

Maximizing our Margin

$$\max_{\alpha}\left(\sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j\right)$$

$$\begin{cases} \alpha_i > 0 \text{ if } y_i(w^T x_i + b) = 1 \\ \alpha_i = 0 \text{ if } y_i(w^T x_i + b) < 1 \end{cases}$$



Maximizing our Margin

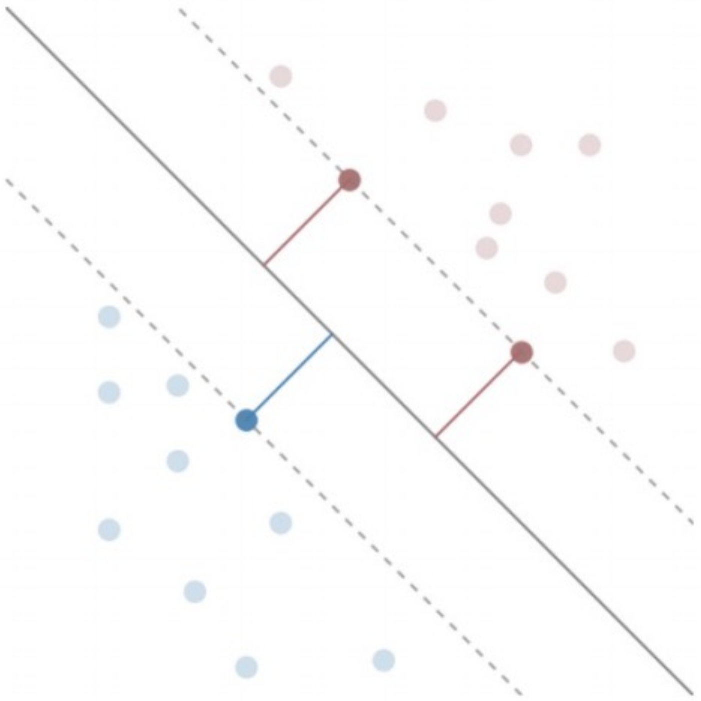
$$\max_\alpha \left(\Sigma_i \alpha_i - \tfrac{1}{2} \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j\right)$$

$$\alpha_i > 0 \text{ if } y_i(w^T x_i + b) = 1$$

$$\alpha_i = 0 \text{ if } y_i(w^T x_i + b) < 1$$

Most of these points (all the greyed out ones) Are 0!

- *They have no impact on what we are maximizing...*

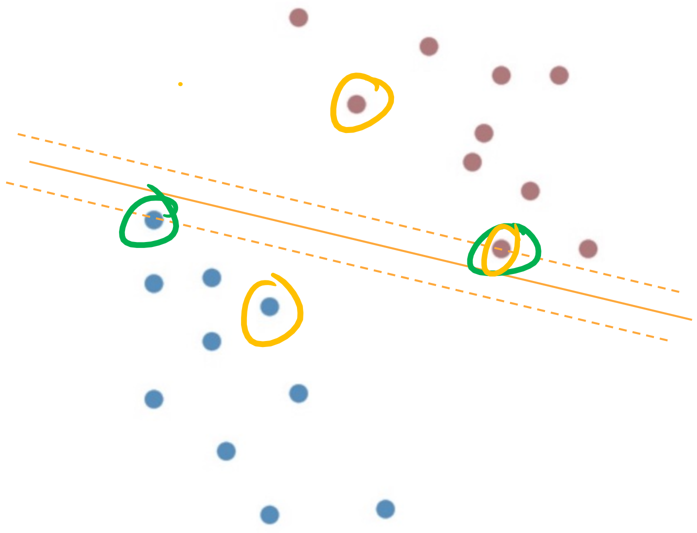


A Less Good Margin

$$\max_\alpha \left(\Sigma_i\, \alpha_i - \tfrac{1}{2}\, \Sigma_{i,j}\, \alpha_i\, \alpha_j\, y_i\, y_j\, x_i \cdot x_j\right)$$

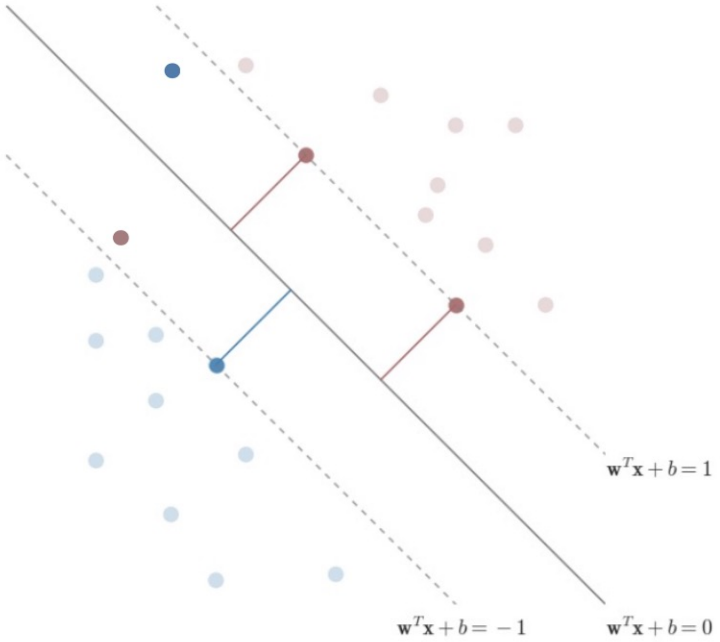
$$\alpha_i > 0 \text{ if } y_i(w^T x_i + b) = 1$$

$$\alpha_i = 0 \text{ if } y_i(w^T x_i + b) < 1$$



Shortcomings of Hard-Margin SVM

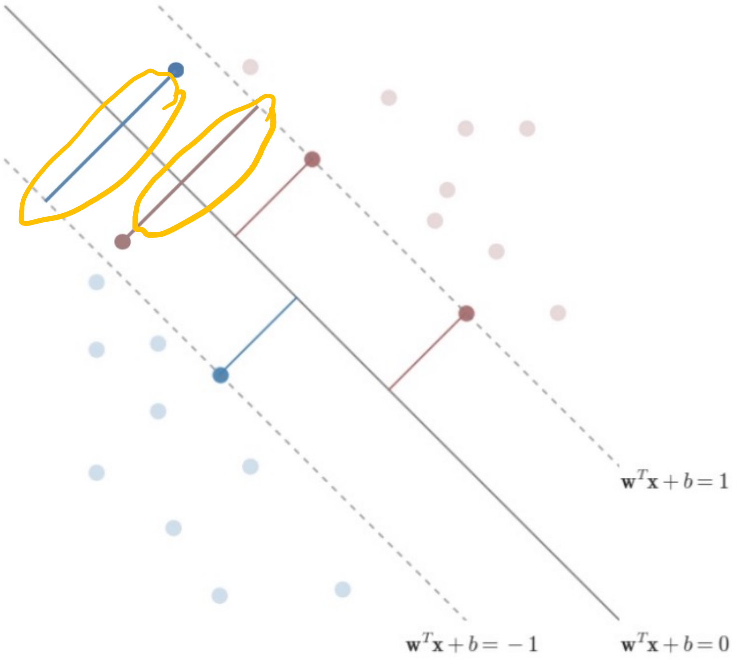
What if Linear Separability does not apply?



Interlude - Soft-Margin SVMs

Overcoming Linear Separability Problem

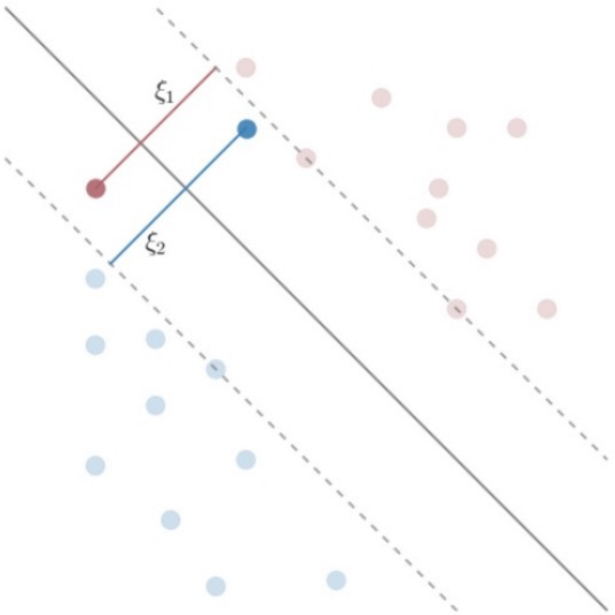
Add some wiggle room to our problem space



Adding a slack term ξ

Add a slack term ξ to every example

(for most examples the ξ is 0)



Adding a slack term ξ – Objective Function

$\text{argmin}_{w,\, b}\,(\tfrac{1}{2}\|w\|^2) \rightarrow \qquad \text{argmin}_{w,\, b}\,(\tfrac{1}{2}\|w\|^2) + C\,\Sigma_{i=1\rightarrow m}\,\xi_i^{\,p}$

C = Tuning Parameter

p = scaling of wrongness

p = 1 is often used

Adding a slack term ξ – Constraints

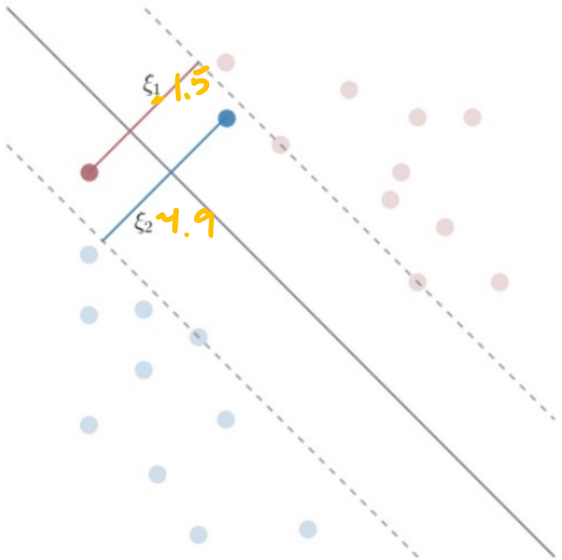
$y_i(w^T x_i + b) >= 1$ → $y_i(w^T x_i + b) >= 1 - \xi_i$

ξ_i relates to distance from the correct margin

ξ_i = 0 when we are on the correct side (or on the boundary) of our *margin*

ξ_i = ½ when we are halfway between margin and decision boundary

ξ_i = 2 when we are 1 margin on the wrong side of the decision boundary



Adding a slack term ξ – Lagrangian

$L(w,b,\alpha) = (½||w||^2 + \Sigma_{i=1...m} (\alpha_i (1 - y_i(w^T x_i + b))))$ →

$L(w,b,\alpha,\xi) = (½||w||^2 - \Sigma_{i=1...m} (\alpha_i (y_i(w^T x_i + b) - 1 + \xi_i))) - \Sigma_{i=1...m} \beta_i \xi_i$

Add ξ as a term to minimize...

Adding a slack term ξ – Lagrangian

$L(w,b,\alpha) = (\frac{1}{2}||w||^2 + \Sigma_{i=1\ldots m} (\alpha_i (1 - y_i(w^T x_i + b))))$ →

$L(w,b,\alpha,\xi)=(\frac{1}{2}||w||^2 - \Sigma_{i=1\ldots m} (\alpha_i (y_i(w^T x_i + b) - 1 + \xi_i))) - \Sigma_{i=1\ldots m} \beta_i \xi_i$

Add ξ as a term to minimize...

------------------------------------ *Magic Happens* --

$\max_\alpha (\Sigma_i \alpha_i - \frac{1}{2} \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$ – *unchanged*

New constraint: $0 <= \alpha <= C$

Adding a slack term ξ – Final Form

$\max_\alpha (L_D) = \max_\alpha \left(\Sigma_i \alpha_i - \frac{1}{2} \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j\right)$

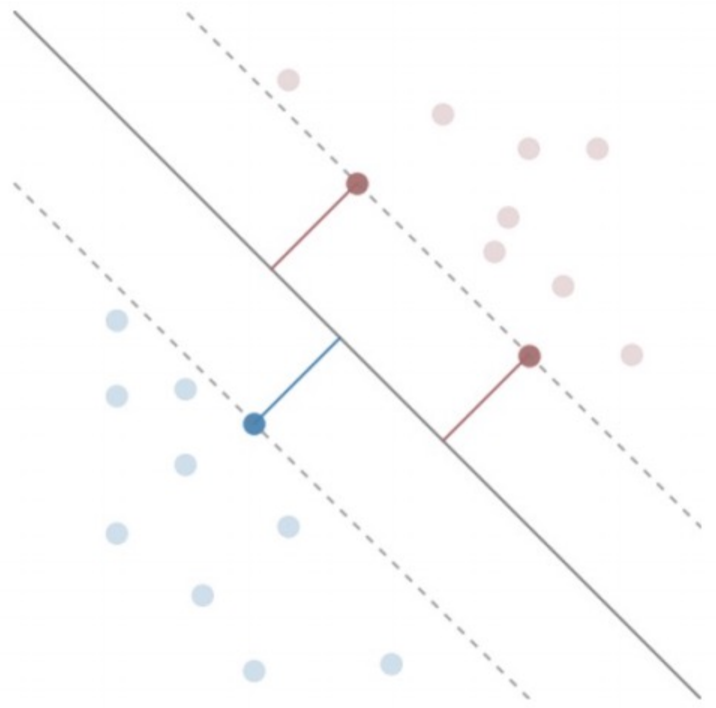
$\alpha_i = C$	if	$y_i(w^T x_i + b) < 1$
$C > \alpha_i > 0$	if	$y_i(w^T x_i + b) = 1$
$\alpha_i = 0$	if	$y_i(w^T x_i + b) > 1$

Maximizing our Margin

$$\max_\alpha \left(\Sigma_i\, \alpha_i - \tfrac{1}{2}\, \Sigma_{i,j}\, \alpha_i\, \alpha_j\, y_i\, y_j\, x_i \cdot x_j\right)$$

$$\alpha_i > 0 \text{ if } y_i(w^T x_i + b) = 1$$

$$\alpha_i = 0 \text{ if } y_i(w^T x_i + b) < 1$$



(Finally) Calculating our Maximum

Coordinate Ascent

$L(\alpha_{i=1\rightarrow m}) = (\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$

Change α_i to make $L(\alpha)$ as big as possible

```
In [ ]: while not converged:
            for ii in [1,...,m]:
                alpha[ii] = argmax_ahat L(alpha[1], ..., ahat, ..., alpha[m])
```

Coordinate Ascent

$L(\alpha_{i=1\rightarrow m}) = (\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$

Change α_i to make $L(\alpha)$ as big as possible

In [ ]:
```
while not converged:
    for ii in [1,...,m]:
        alpha[ii] = argmax_ahat L(alpha[1], ..., ahat, ..., alpha[m])
```

$0 = \Sigma_{i=1\rightarrow m} \alpha_i y_i$

We can't change α_i one at a time!

Sequential Minimal Optimizer

$L(\alpha_{i=1\rightarrow m}) = (\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$

Change α_i, α_j to make $L(\alpha)$ as big as possible

In [ ]:
```
while not converged:
    for ii in [1,...,m]:
        select jj != ii at random (or via heuristic)
        update alpha[ii] and alpha[jj] to maximize L(alpha)
    if KKT conditions satisfied:
        exit
```

$0 = \Sigma_{i=1\rightarrow m} \alpha_i y_i$

Sequential Minimal Optimizer

Change α_i, α_j to make $L(\alpha)$ as big as possible

$0 = \Sigma_{i=1 \rightarrow m} \alpha_i y_i$

$0 = \alpha_1 y_1 + \alpha_2 y_2 + \Sigma_{k=3 \rightarrow m} \alpha_k y_k$

Sequential Minimal Optimizer

Change α_i, α_j to make $L(\alpha)$ as big as possible

$0 = \Sigma_{i=1 \rightarrow m} \alpha_i y_i$

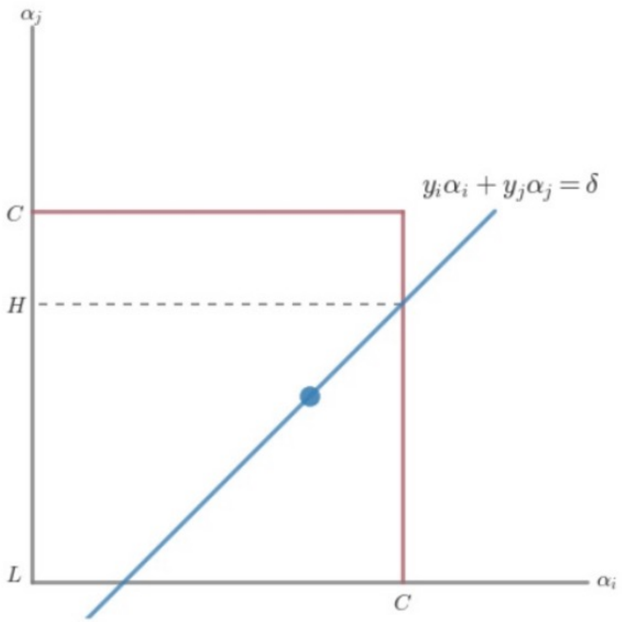
$0 = \alpha_1 y_1 + \alpha_2 y_2 + \Sigma_{k=3 \rightarrow m} \alpha_k y_k$

$\alpha_i y_i + \alpha_j y_j = - \Sigma_{k!=i,j} \alpha_k y_k = \delta$

Sequential Minimal Optimizer

$\alpha_i\, y_i + \alpha_j\, y_j = - \Sigma_{k!=i,j}\, \alpha_k\, y_k = \delta$

$0 <= \alpha_i <= C$



Sequential Minimal Optimizer

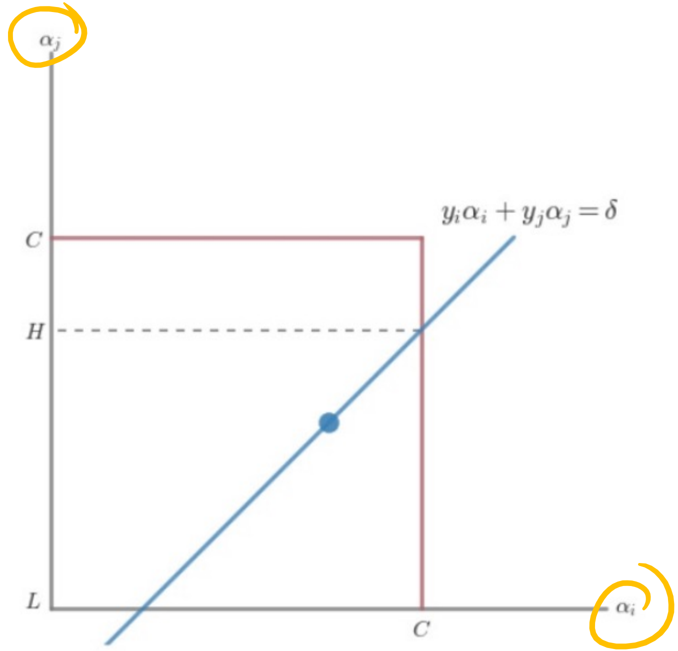
$\alpha_i y_i + \alpha_j y_j = - \Sigma_{k!=i,j} \alpha_k y_k = \delta$

$0 <= \alpha_i <= C$

$y_i = -1$

$y_j = +1$

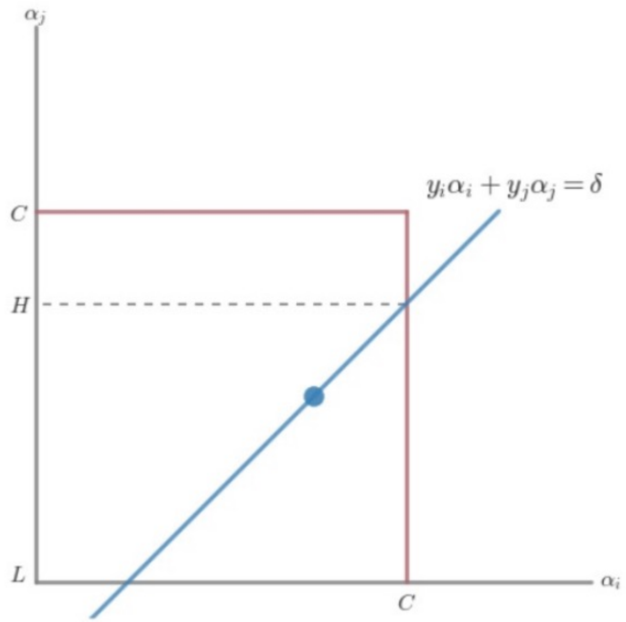
	$y_i != y_j$		$y_i = y_j$
$L = \max(0, \alpha_j - \alpha_i)$	or		$\max(0, \alpha_j + \alpha_i - C)$
$H = \min(C, C + \alpha_j - \alpha_i)$	or		$\min(C, \alpha_j + \alpha_i)$



Sequential Minimal Optimizer

$$\alpha_j = \begin{cases} H \text{ if } \alpha_j > H \\ \alpha_j \text{ if } L <= \alpha_j <= H \\ L \text{ if } \alpha_j < L \end{cases}$$

α_i such that $\alpha_i\, y_i + \alpha_j\, y_j = - \Sigma_{k!=i,j}\, \alpha_k\, y_k = \delta$



Sequential Minimal Optimizer

$$\alpha_j = \begin{cases} H \text{ if } \alpha_j > H \\ \alpha_j \text{ if } L <= \alpha_j <= H \\ L \text{ if } \alpha_j < L \end{cases}$$

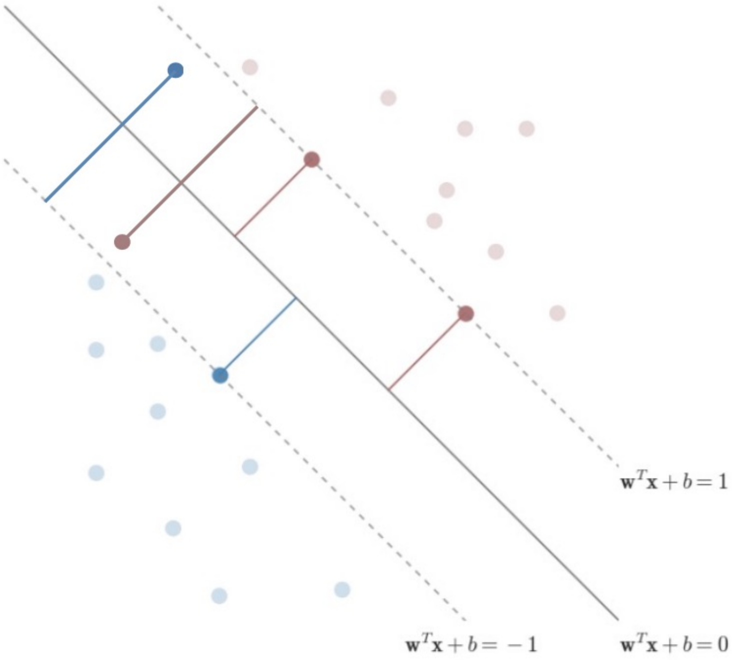
α_i such that $\alpha_i y_i + \alpha_j y_j = - \Sigma_{k!=i,j} \alpha_k y_k = \delta$

In [ ]:

```
while not converged:
    for ii in [1,...,m]:
        select jj != ii at random (or via heuristic)
        update alpha[ii] and alpha[jj] to maximize L(alpha)
    if KKT conditions satisfied:
        exit
```

Shortcomings of Soft-Margin SVM

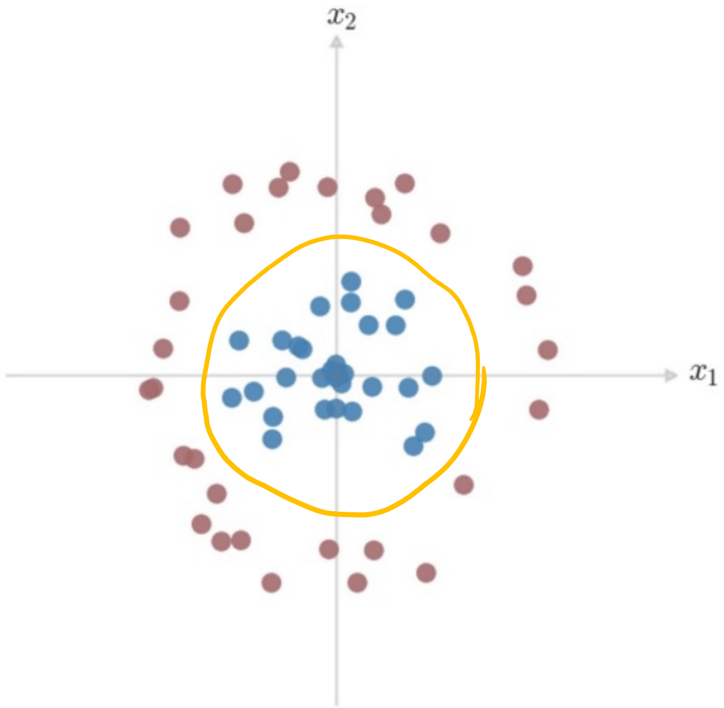
We added some wiggle room



Shortcomings of Soft-Margin SVM

We added some wiggle room

But what if it's not linearly separable?



Linear Inseparability – 1D Case

Linear Inseparability – 1D Case

Your arterial blood's pH should be between 7.35 and 7.45

Otherwise, you're unhealthy (for some reason), we should run further tests.

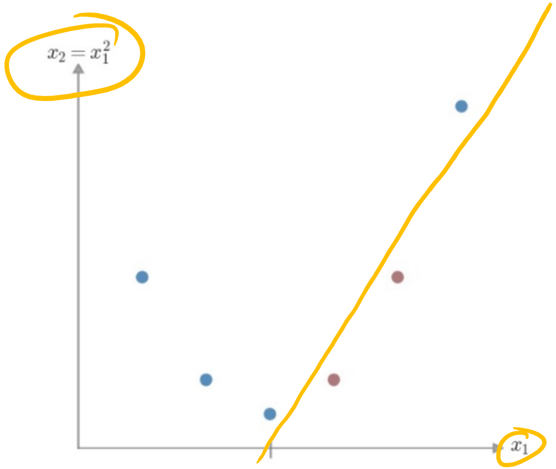
https://www.maximumfun.org/sawbones/sawbones-alkaline-water
https://opentextbc.ca/anatomyandphysiology/chapter/26-5-disorders-of-acid-base-balance/

Linear Inseparability – 1D Case

What can we do to capture this idea?

Linear Inseparability – 1D Case

Project our data into a higher dimensional space



Linear Inseparability – Dimension Projection

Start with our initial feature vector $x = [x_1]$

Create an augmented feature vector $\varphi(x) = [x_1, x_1^2]$

Linear Inseparability – Dimension Projection

Start with our initial feature vector $x = [x_1]$

Create an augmented feature vector $\varphi(x) = [x_1, x_1^2]$

Replace x with $\varphi(x)$ in our optimization problem!

Our Optimization Problem

$(\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$

Our Optimization Problem

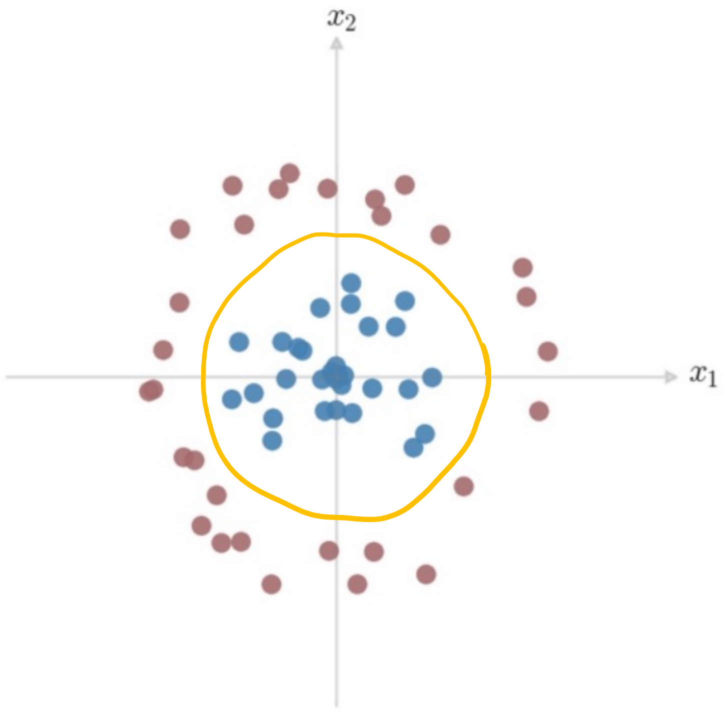
$(\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$

$(\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j \varphi(x)_i \cdot \varphi(x)_j)$

Dimension Projection

How would we separate these classes?

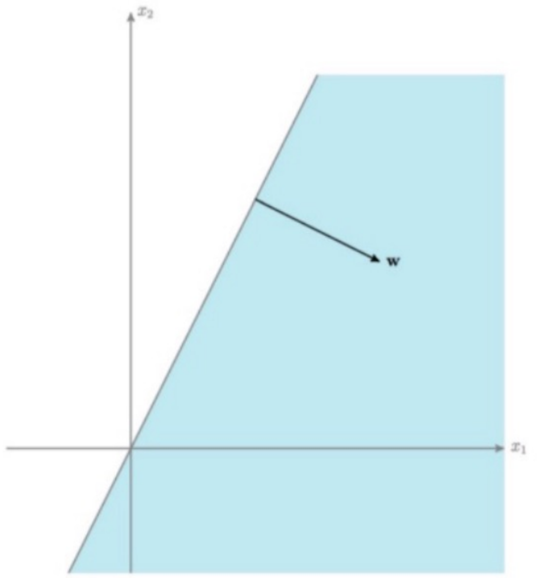
$x_1^2 + x_2^2$



Dimension Projection - Circle

How would we separate these classes?

We think in terms of sides of a boundary...

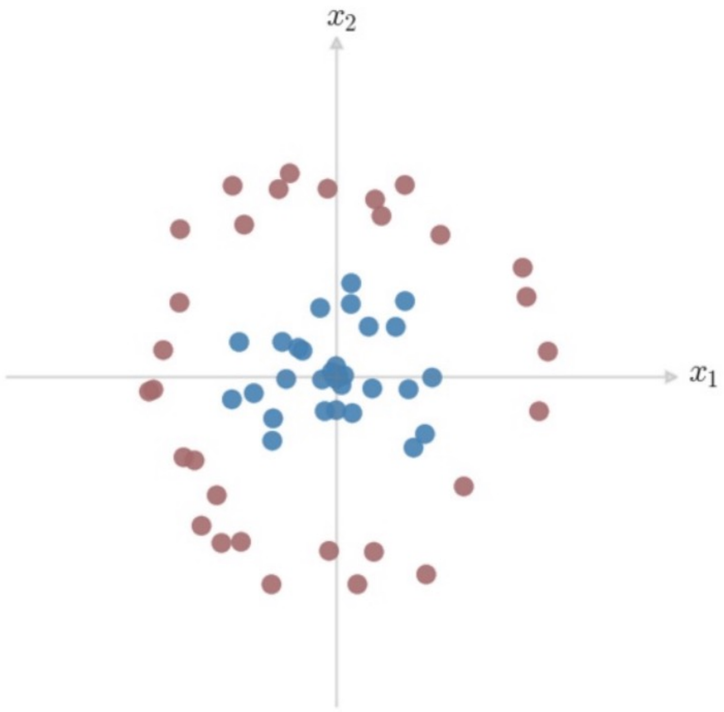


Dimension Projection

How would we separate these classes?

What kind of boundary can we give this space?

Can we express that boundary in terms of our given features?

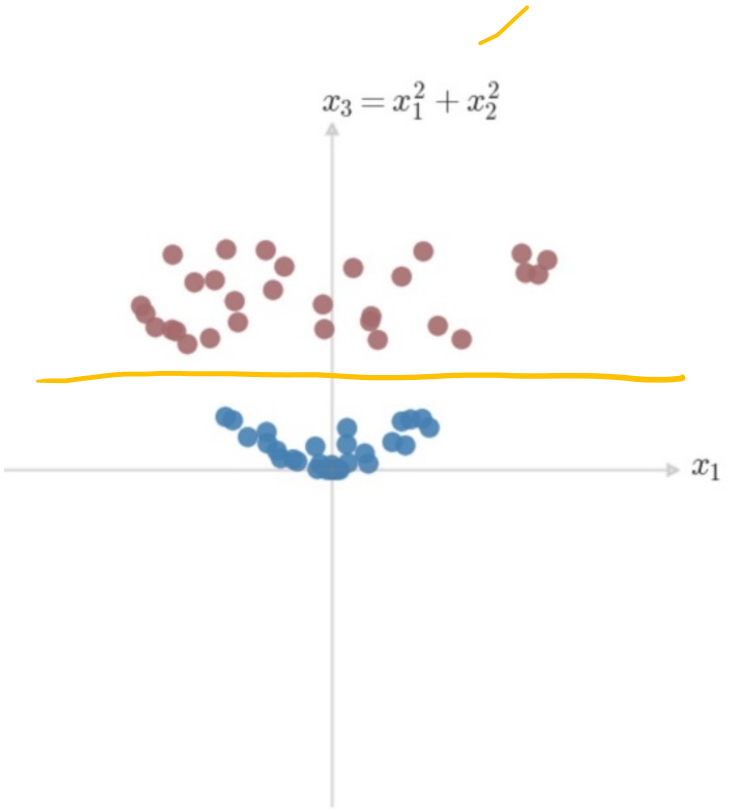


Dimension Projection

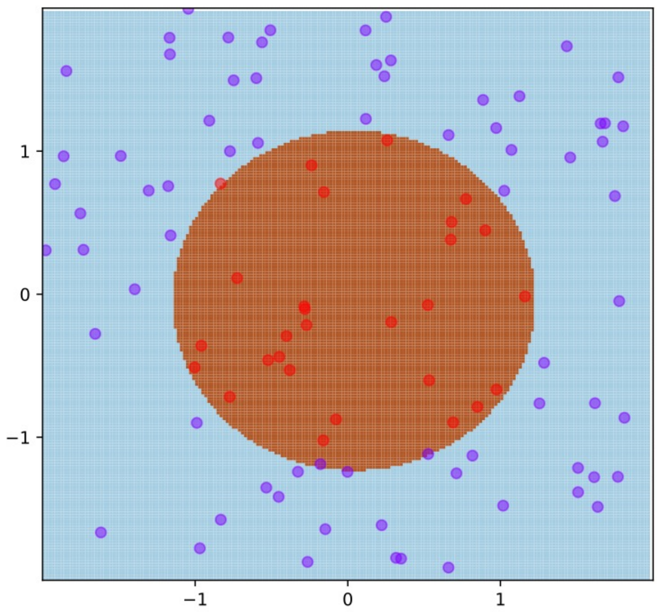
How would we separate these classes?

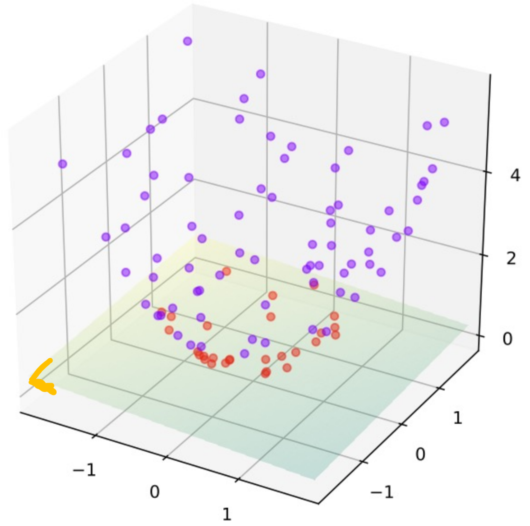
What kind of boundary can we give this space?

Can we express that boundary in terms of our given features?



Dimension Projection - Circle





Linear Inseparability – Dimension Projection

Start with our initial feature vector $x = [x_1, x_2]$

Create an augmented feature vector $\varphi(x) = [x_1, x_2, x_1^2 + x_2^2]$

Replace x with $\varphi(x)$ in our optimization problem!

Decision Boundary is a plane in 3D Space

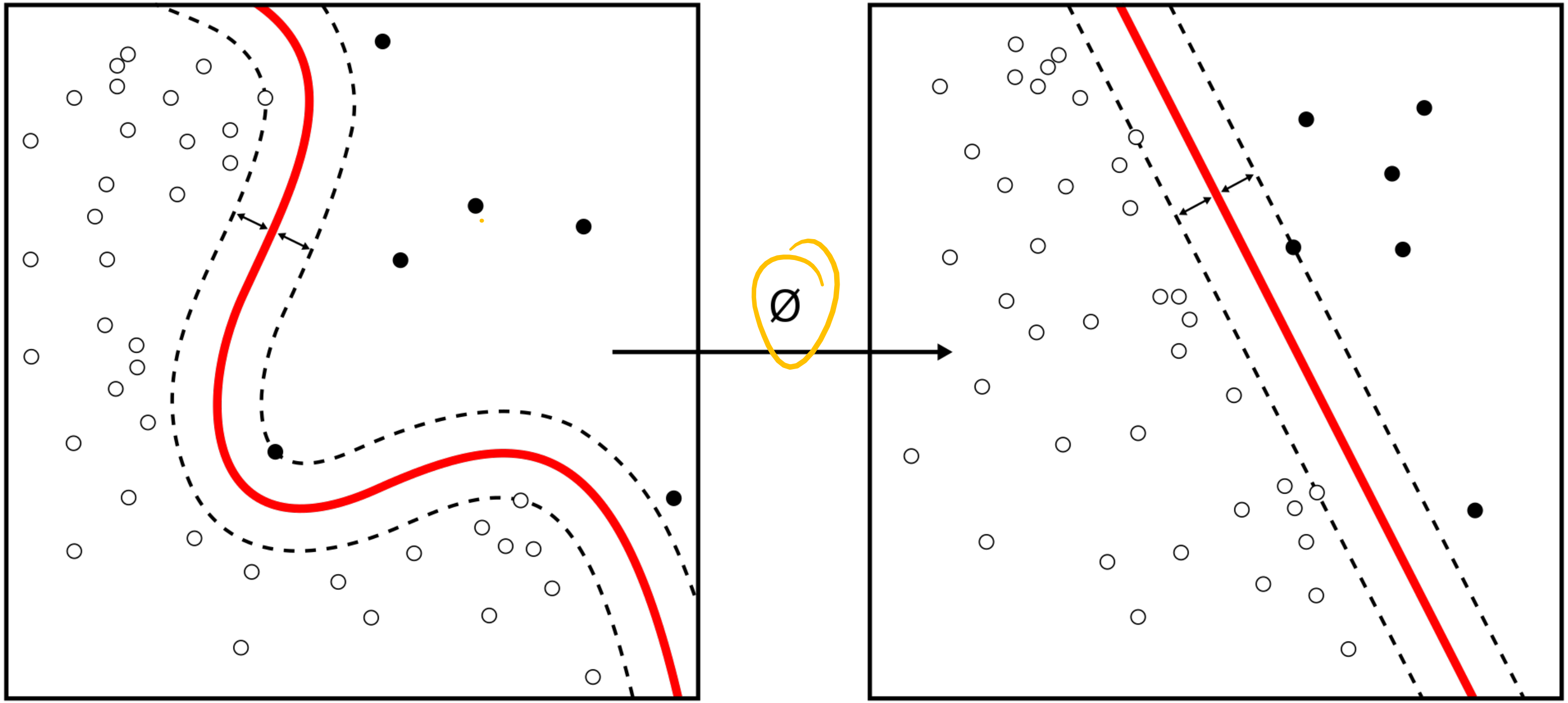
$$x_1^2 + x_2^2 = \textit{constant}$$

Our Optimization Problem

$$(\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$$

$$(\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j \varphi(x)_i \cdot \varphi(x)_j)$$

Generalizing our Optimization



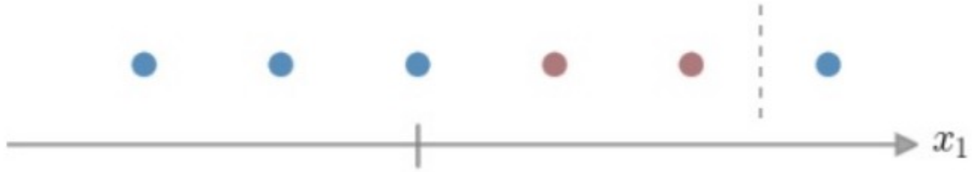
Our Optimization Problem – In Practice

For a 3-Dimensional Case…

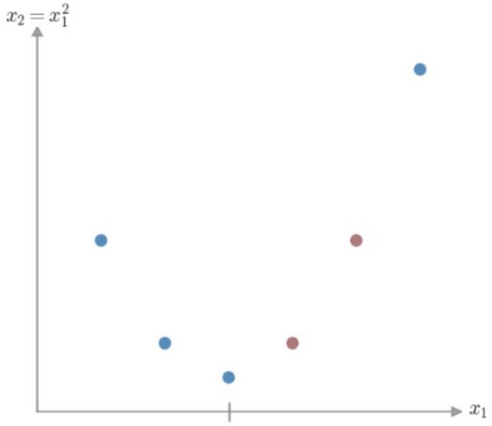
$X = [x_1, x_2, x_3]$

$\varphi(x) = (x_i^T x_j)^2$

Linear Inseparability – 1D Case



Project our data into a higher dimensional space



Our Optimization Problem – In Practice

For a 3-Dimensional Case...

$X = [x_1, x_2, x_3]$

$\varphi(x) = (x_i^T x_j)^2$

3^2

$\varphi[X] = [x_1\, x_1,\ x_1\, x_2,\ x_1\, x_3,\ x_2\, x_1,\ x_2\, x_2,\ x_2\, x_3,\ x_3\, x_1,\ x_3\, x_2,\ x_3\, x_3]$

Generalizing our Optimization – Curse of Dimensionality

Take a 64-feature vector

Transform it using $\varphi(x) = (x_i^T x_j)^4$

How many dimensions do I have?

64^2

Generalizing our Optimization – Curse of Dimensionality

Take a 64-feature vector

Transform it using $\varphi(x) = (x_i^T x_j)^2$

How many dimension do I have?

64^2 = 4096

Our Optimization Problem

$(\Sigma_i \alpha_i - \frac{1}{2} \Sigma_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{x_i \cdot x_j})$

$(\Sigma_i \alpha_i - \frac{1}{2} \Sigma_{i,j} \alpha_i \alpha_j y_i y_j \underbrace{\varphi(x)_i \cdot \varphi(x)_j})$

Redefine $\varphi(x)_i \cdot \varphi(x)_j$ as $K(x, z)$ (x_i becomes x, x_j becomes z)*

*there are restrictions on this mapping – we'll get there!

Our Optimization Problem – In Practice

x, z are vectors of real numbers in N-Dimensional Space

$K(x, z) = (x^T z)^2$

Our Optimization Problem – In Practice

x, z are vectors of real numbers in N-Dimensional Space

$K(x, z) = (x^T z)^2$

$K(x, z) = (\Sigma_{i=1 \rightarrow n} (x_i z_i)) (\Sigma_{j=1 \rightarrow n} (x_j z_j))$

Our Optimization Problem – In Practice

x, z are vectors of real numbers in N-Dimensional Space

$K(x, z) = (x^T z)^2$

$K(x, z) = (\Sigma_{i=1 \rightarrow n} (x_i z_i)) (\Sigma_{j=1 \rightarrow n} (x_j z_j))$

$= \Sigma_{i=1 \rightarrow n} \Sigma_{j=1 \rightarrow n} (x_i z_i x_j z_j)$

$= \Sigma_{i,j=1 \rightarrow n} (x_i z_i x_j z_j)$

$(\Sigma_i \alpha_i - ½ \Sigma_{i,j} \alpha_i \alpha_j y_i y_j (x_i z_i x_j z_j))$

Kernels Save Time!

Computing $\varphi(x) = (x_i^T x_j)^2$ takes $O(n^2)$

Computing $K(x,z)$ takes $O(n)$

Boundaries of Valid Kernels

For any set of points (x_m), you can store $K(x_i,x_j)$ in an *m* by *m* matrix K

Boundaries of Valid Kernels – Symmetry

For any set of points (x_m), you can store $K(x_i,x_j)$ in an *m* by *m* matrix K

$K_{ij} = K(x^i) , x^j) = \varphi(x^i))^T \varphi(x^j) = \varphi(x^j)^T \varphi(x^i) = K(x^j , x^i) = K_{ji}$

K must be symmetric

Boundaries of Valid Kernels – Positive Semi-Definite

For any set of points (x_m), you can store $K(x_i, x_j)$ in an m by m matrix K

$z^T K z \geq 0$

Boundaries of Valid Kernels – Mercer's Theorem

Theorem (Mercer). Let $K : R^n \times R^n \longrightarrow \mathbb{R}$ be given. Then for K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x^1, \ldots, x^m\}$, $(m < \infty)$, the corresponding kernel matrix is symmetric positive semi-definite.

Kernels Intuition

Dot product $x \cdot y$ tells you about the similarity between x and y

If both vectors are positive along a dimension, dot product will grow
If one vector is 0 along a dimension, dot product does not grow
If vectors are opposite signs along a dimension, dot product will shrink

Kernels just add dimension combinations to dot product!

Popular Kernels

Linear Kernel: $K(x, z) = (x^T z)$

Popular Kernels

Linear Kernel: $K(x, z) = (x^T z)$

Polynomial Kernel: $K(x, z) = (x^T z + c)^p$

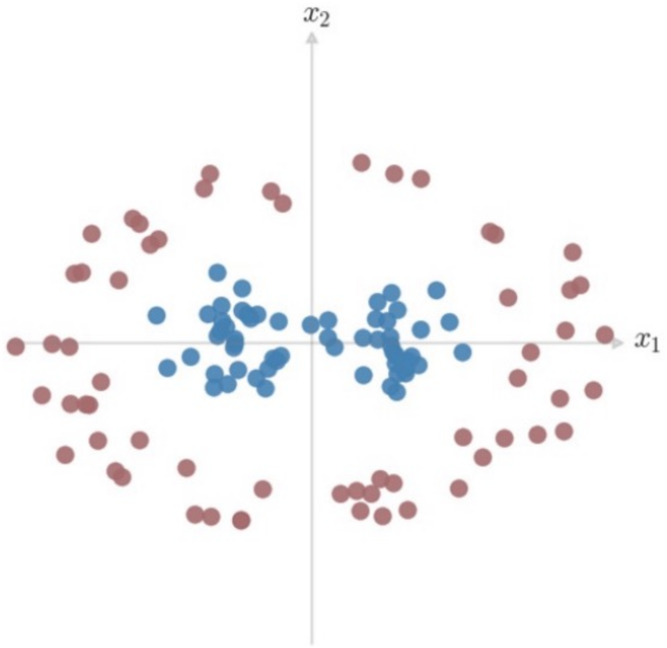
Popular Kernels

Linear Kernel: $K(x, z) = (x^T z)$

Polynomial Kernel: $K(x, z) = (x^T z + c)^p$

Radial Basis Function(RBF) Kernel: $K(x, z) = \exp(-\gamma ||x - z||^2)$

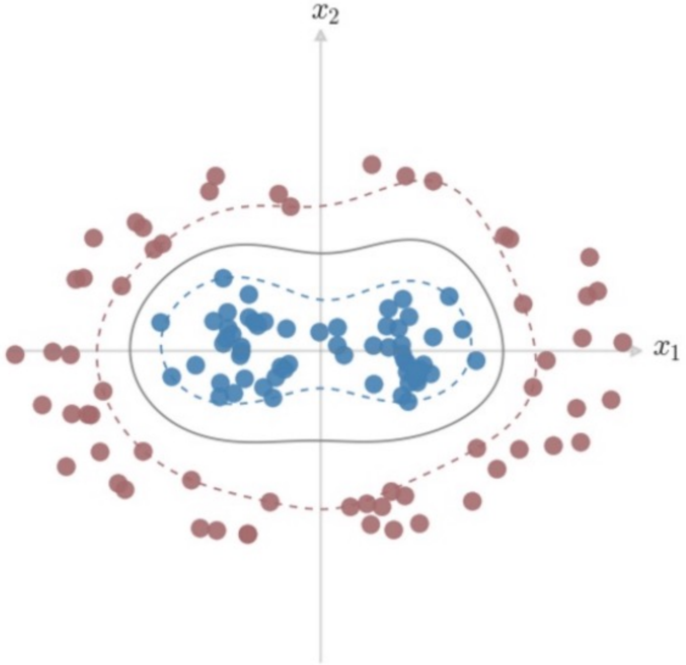
RBF Kernel for SVM



RBF Kernel for SVM

$\gamma = 3$

$C = 5$



Popular Kernels

Linear Kernel: $K(x, z) = (x^T z)$

Polynomial Kernel: $K(x, z) = (x^T z + c)^p$

Radial Basis Function(RBF) Kernel: $K(x, z) = \exp(-\ \gamma||x - z||^2)$

Gaussian RBF Kernel: $K(x, z) = \exp((-\ ||x - z||^2) / 2\sigma^2)$

SVM + Kernels

Pros

Works great “out-of-the-box”

Maintains convex objective function – guaranteed optimal solution

Solutions are *sparse* thanks to Kernels

Flexibly map to arbitrary decision boundary

SVM + Kernels

Pros

Works great “out-of-the-box”

Maintains convex objective function – guaranteed optimal solution

Solutions are *sparse* thanks to Kernels

Flexibly map to arbitrary decision boundary

Cons

Optimization can be slow

Choosing a Kernel is manual

Perceptrons

Linear Classification Task

$w^T x + b \rightarrow \pm 1$

If $w^T x + b > 0$, $y = 1$

If $w^T x + b < 0$, $y = -1$

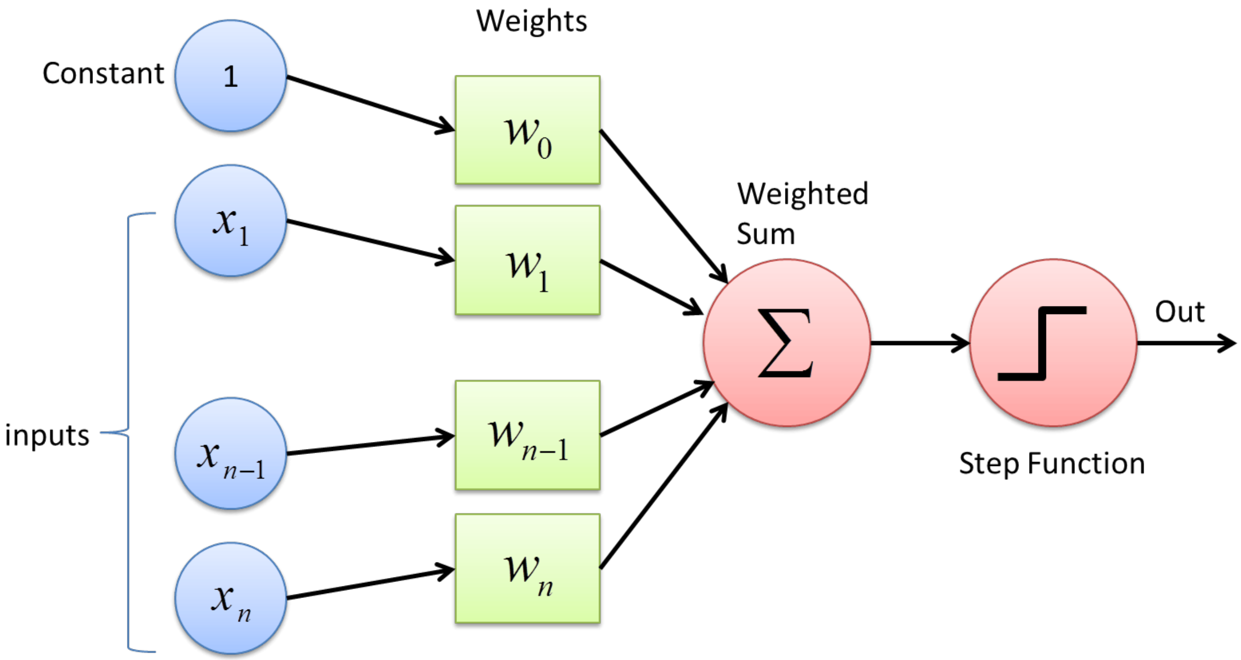
$w^T x + b = 0$ is an edge case

Visualizing Classification – The Perceptron

$w^T x + b \rightarrow \pm 1$

If $w^T x + b > 0, y = 1$

If $w^T x + b < 0, y = -1$



Training a Perceptron – "Online" Learning

Initialize w to an all-zero vector.

For some fixed number of iterations, or until some stopping criterion is met:

For each training example x_i with ground truth label $y_i \in \{-1, 1\}$:

Let $\hat{y} = \text{sign}(w^T x_i)$.

If $\hat{y} \neq y_i$, update $w \leftarrow w + y_i x_i$.

Kernel Perceptron

Perform our Kernel transformation of our data, then generate our perceptron!

Kernel Perceptron

Create the matrix of our Kernel representation of our data

Linear Kernel: $K(x, z) = (x^T z)$

Polynomial Kernel: $K(x, z) = (x^T z + c)^p$

Radial Basis Function(RBF) Kernel: $K(x, z) = \exp(-\gamma||x - z||^2)$

Gaussian RBF Kernel: $K(x, z) = \exp((-||x - z||^2) / 2\sigma^2)$

Training our Perceptron (Fit)

We no longer have “weights” to train on…

Training our Perceptron (Fit)

We no longer have “weights” to train on...

But we can use our Dual Expression of the problem!

$$\Sigma_{i=1 \rightarrow n} \alpha_i y_i K(x_i, x_j)$$

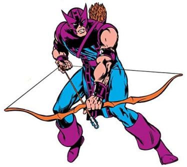
And use the same perceptron update as before, but increment our α by 1 each time.

If an example incremented α, then it’s a *support vector*

Interlude: Pros and Cons

When to use what classifiers

K-Nearest Neighbors



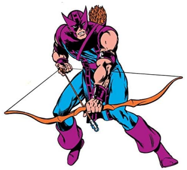
Pro

- Easy
- "train"?
- Non linear

Con

- Store whole data set for fitted model
- Not the best usually...

K-Nearest Neighbors



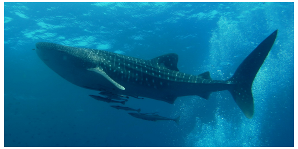
Pro

- Works with small N
- Works for non-binary cases
- Can be run without training

Con (Ex)

- Doesn't give much of a description of results beyond "similar to A, B, C..."
- Has trouble with features at differing scales

Naïve Bayes




Pro

Interpretable

Easy to Implement → Fast(ish)

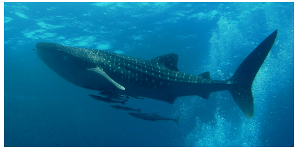
Probability or Score

Con

Assume Independence

Naïve Bayes

Pro

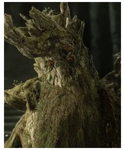


Con

- Works for non-binary cases
- Can be run without training

- Hard to implement for non-categorical data (without binning)

Decision Tree



Pro

Con (Ex)

Interpretable

Pruning

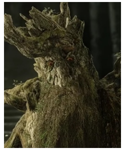
underfit

overfit

Nonlinear

Nonlinear

Decision Tree



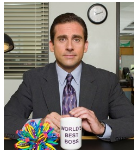
Pro

- Easy to implement
- Easy to follow
- Easy(ish) to interpret

Con (Ex)

- Prone to overfitting

Decision Forest / Adaboost



Pro

Deal w/ overfit/underfit

pretty effective

pseudo-feature selection

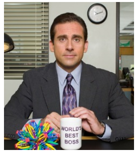
Con

Time to train

Depend on weak learner

Decision Forest / Adaboost

Pro



Con (Ex)

- Adaptive to difficult-to-classify cases

- Difficult to interpret

Logistic Regression

Pro

Con

Logistic Regression

Pro

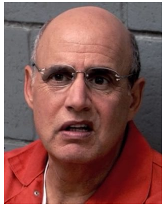
- Linear decision boundary

Con

- Binary only

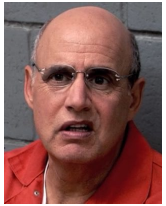
Support Vector Machines

Pro



Con

Support Vector Machines



Pro

- Linear decision boundary

Con (Ex)

- Binary only

slow

Next time...

Neural Networks!