11-755— Spring 2021
Large Scale Multimedia Processing

Lecture 3/6

Multimedia processing: General

Representation by quantization, classification by SVM

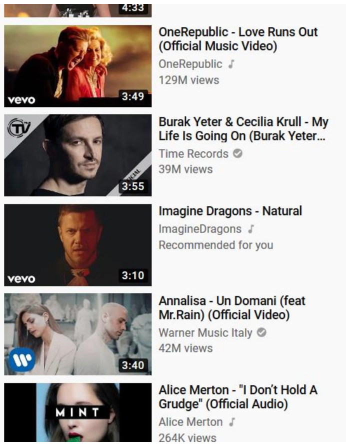
Rita Singh

Carnegie Mellon University

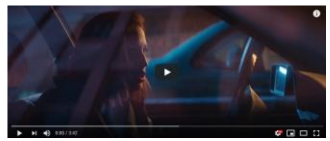
In this lecture

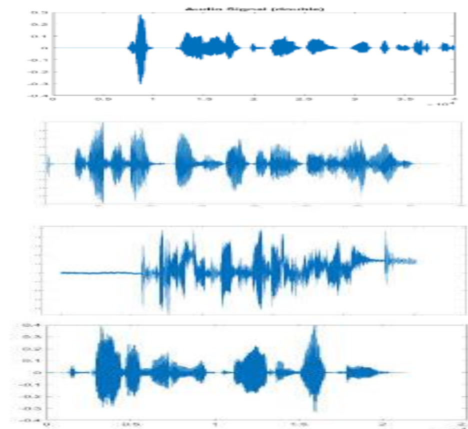
- Digital multimedia: Recording and devices
  - Audio
  - Images
  - Video
  - Text
- Digital multimedia: Processing
  - Audio processing
  - Two generic processing techniques

Typical problem

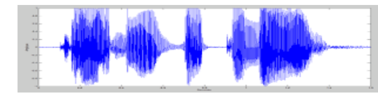


?





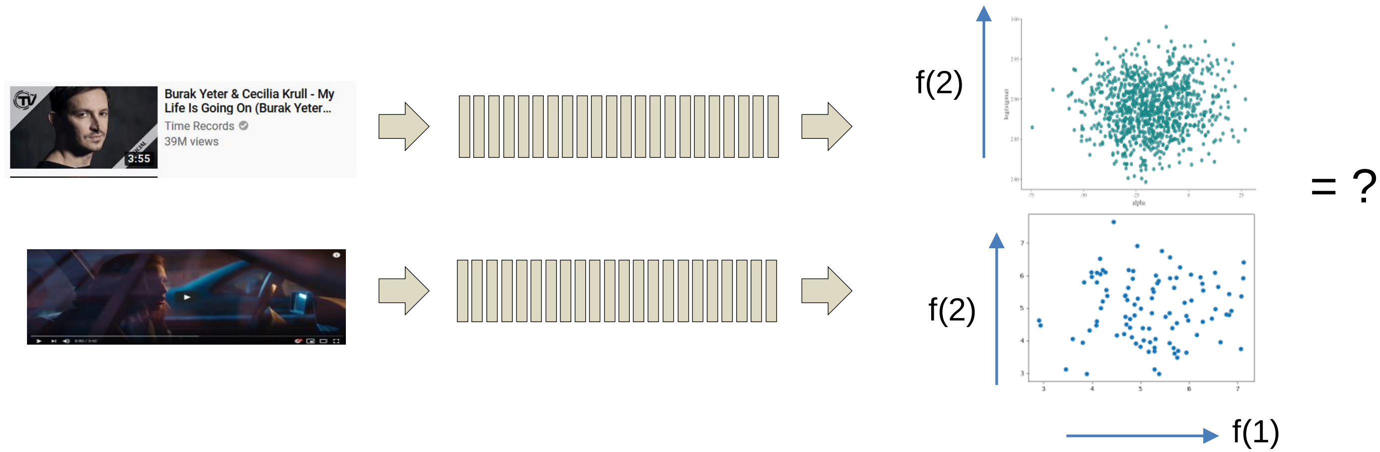
?



- Given two video recordings: determine if they are the same event
  - Generalization: Given a collection of video recordings, and a query recording, find the ones that match

- Given two audio recordings: determine if they represent the same type of event (or recordings from the same speaker)
  - Generalization: Given a collection of audio recordings, and a query recording, find the ones that match
- Other similar problems of comparison

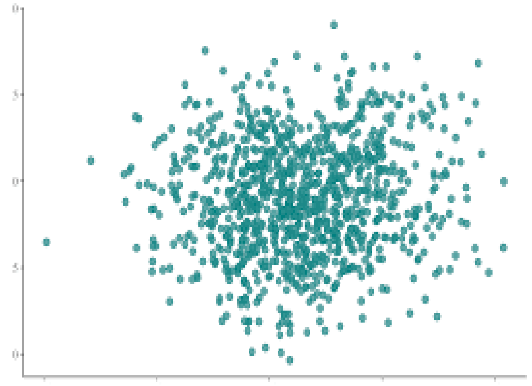
The actual problem

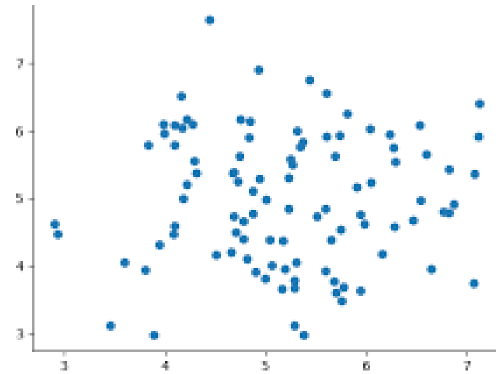


- A video comprises a collection of frames
  - Each of which may produce one or more "feature vectors"
  - The actual information in the content lies in the overall distribution of the features
  - Comparing a query video to a video: How similar is the distribution of features from these two recordings?

- An audio recording comprises a sequence of spectral (or derived) feature vectors
  - The information about the content lies in their distribution
  - Comparing two recordings: How similar are the distributions of the features in these two recordings?

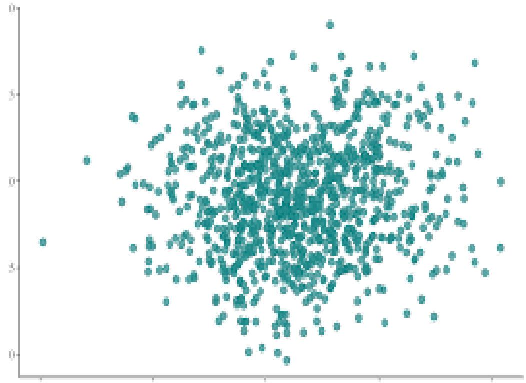
Comparing distributions

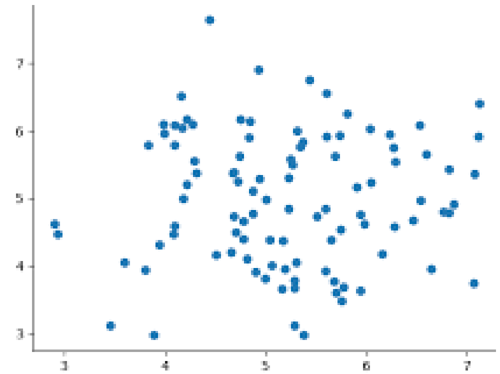




- How to represent the distributions?
- Possibility: KL Divergence or any other divergence function that compares distributions
  - But we don’t have formula for the PDFs?
- How to represent the distributions?
  - As histograms?
  - Using some kind of parametric model, e.g. GMM?

Comparing distributions

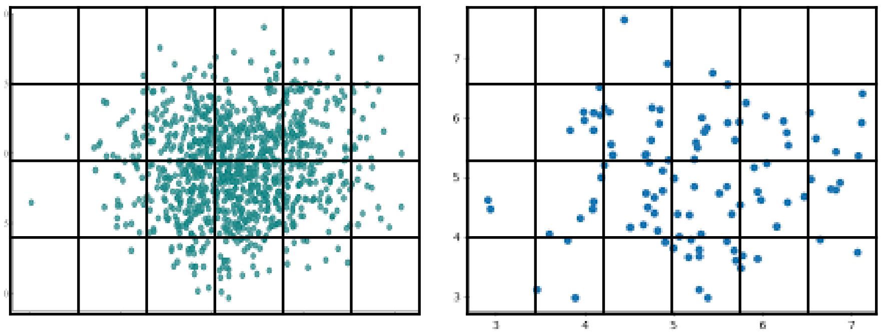




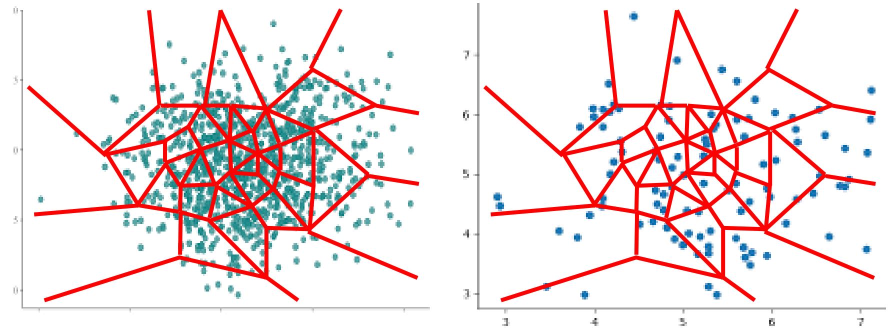
- How to represent the distributions?
- Possibility: KL Divergence or any other divergence function that compares distributions
  - But we don't have formula for the PDFs?
- How to represent the distributions?
  - As histograms?
  - Using some kind of parametric model, e.g. GMM?

Computing histograms

Uniform grid



Non-uniform (data-specific) grid



- Computing histograms is trivial for unidimensional data
  - Partition axis into uniform bins
  - Count instance per bin

- For higher dimensional data, this will not work
  - Most bins will be empty if we use a uniform grid
  - Better solution – partition the space into bins such that most bins are “focused” on data-rich regions, and the “empty” regions are placed into a small number of bins

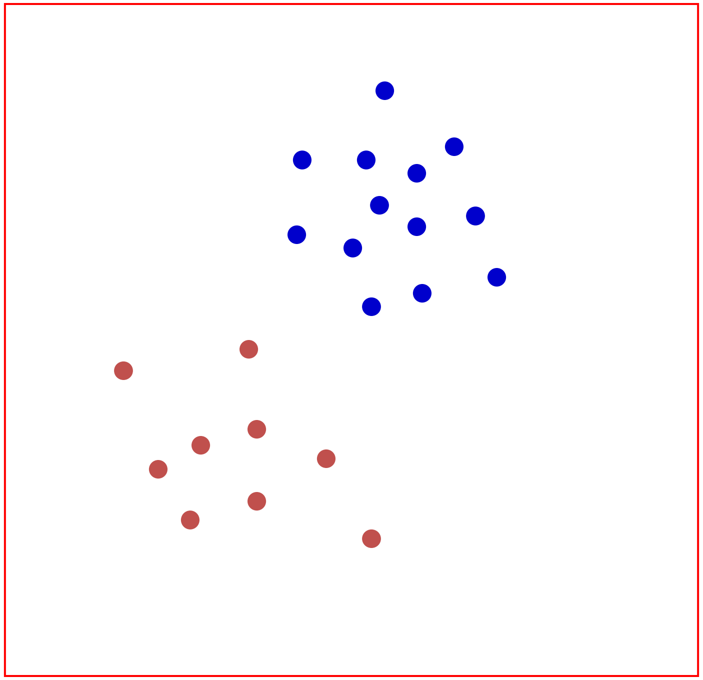
- Problem: How to determine the best way to partition the space for non-uniform bins?
  - Solution – K-means clustering

So lets look at clustering

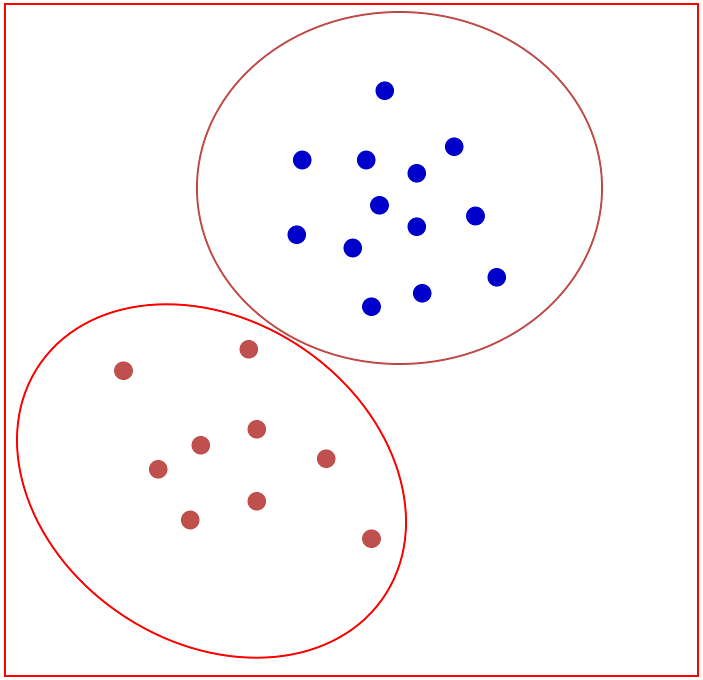
- And K-means clustering
- And how it can be used to form bins and histograms
  - Also called “bag of words” representations

Clustering

- What is clustering
  - Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)



Clustering



- What is clustering
  - Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)
- How is it done
  - Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind

Why Clustering

- Unsupervised categorization: Automatic grouping into "Classes"
  - Different clusters may show different behavior
- Representation: Quantization
  - All data within a cluster are represented by a single point
- Preprocessing step for other algorithms
  - Indexing, categorization, etc.

Clustering Criterion

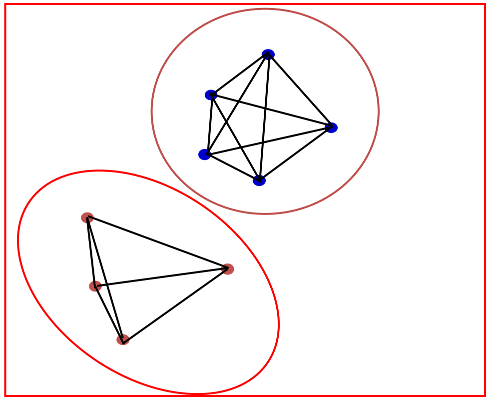
- The "Clustering criterion" actually has two aspects

- Cluster compactness criterion
  - Measure that shows how "good" clusters are

- Distance of a point from a cluster
  - To determine the cluster a data vector belongs to

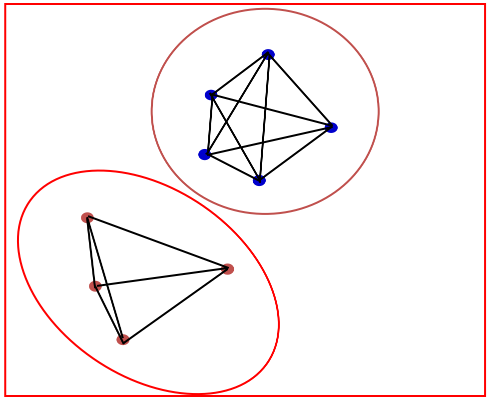
“Compactness” criteria for clustering

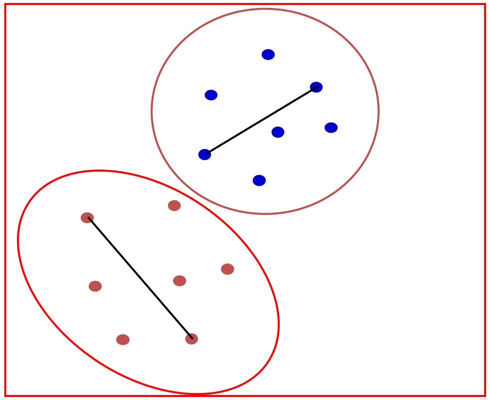
- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster



“Compactness” criteria for clustering

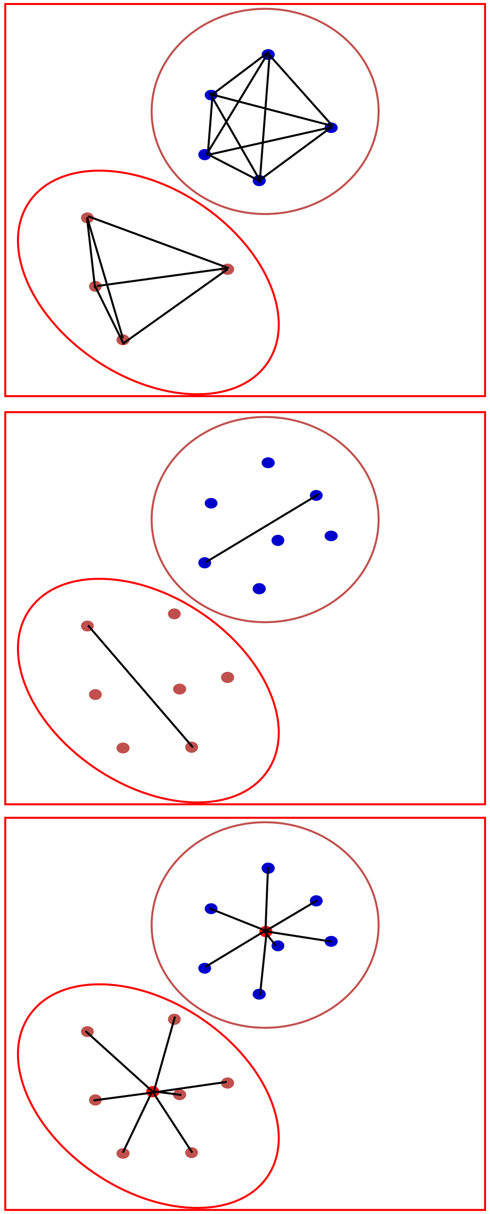
- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster





"Compactness" criteria for clustering

- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster
  - Total distance of every element in the cluster from the centroid of the cluster
    - Typically the Euclidean distance



Optimal clustering: Exhaustive enumeration

- All possible combinations of data must be evaluated
  – If there are M data points, and we desire N clusters, the number of ways of separating M instances into N clusters is

$$\frac{1}{M!}\sum_{i=0}^{N}(-1)^i\binom{N}{i}(N-i)^M$$

  – Exhaustive enumeration based clustering requires that the compactness measure be evaluated for every one of these, and the best one chosen

- This is the only correct way of optimal clustering
  – Unfortunately, it is also computationally unrealistic

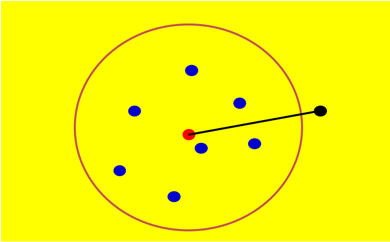
Greedy algorithms

- Exhaustive evaluation is not feasible

- Instead, we will use greedy algorithms based on iterative greedy assignment of instances to clusters

- Requires a way to determine how far an instance is from a cluster

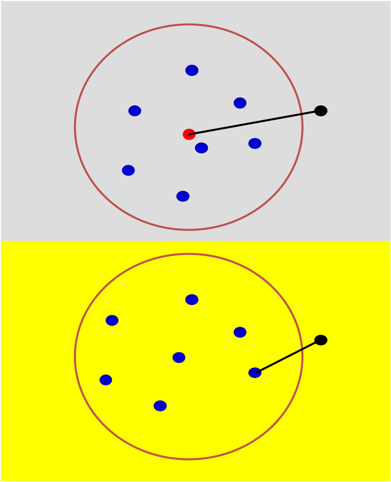
Clustering: Distance from cluster



- How far is a data point from a cluster?
  - Euclidean distance from the centroid of the cluster

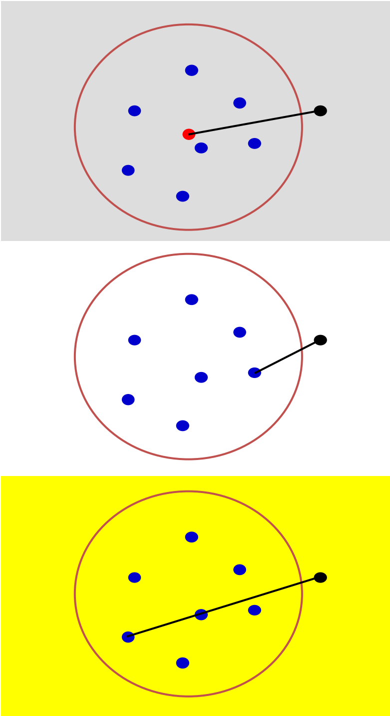
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean distance from the centroid of the cluster
  - Distance from the closest point in the cluster



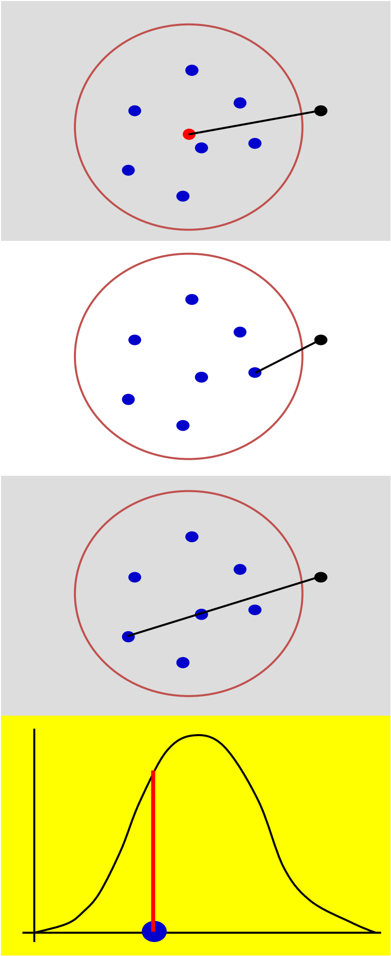
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster



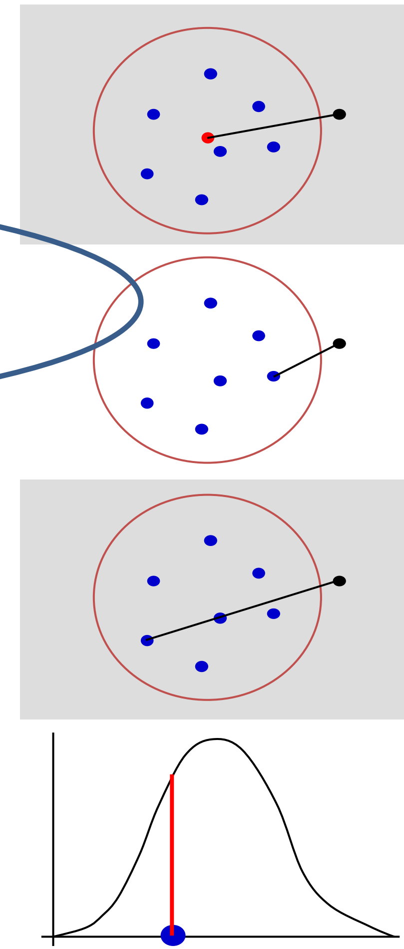
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster
  - Probability of data measured on cluster distribution



Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster
  - Probability of data measured on cluster distribution

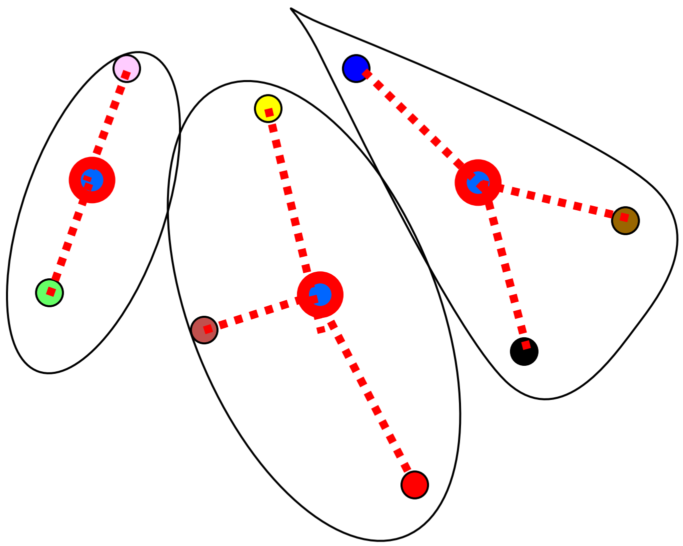


Generalized Lloyd Algorithm: K–means clustering

- K means is an iterative algorithm for clustering ***vector*** data
  - McQueen, J. 1967. “Some methods for classification and analysis of multivariate observations.” Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 281-297
- General procedure:
  - Initially group data into the required number of clusters somehow (initialization)
  - Assign each data point to the closest cluster
  - Once all data points are assigned to clusters, redefine clusters
  - Iterate

K–means

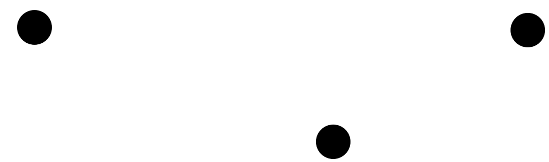
- Problem: Given a set of data vectors, find natural clusters
- Clustering criterion is **scatter**: distance from the centroid
  - Every cluster has a centroid
  - The centroid represents the cluster
- **Definition**: The **centroid** is the weighted mean of the cluster
  - Weight = 1 for basic scheme



$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

K–means

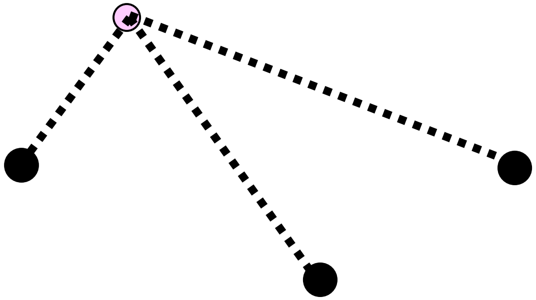
1. Initialize a set of centroids randomly



K–means

1. Initialize a set of centroids randomly

2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$

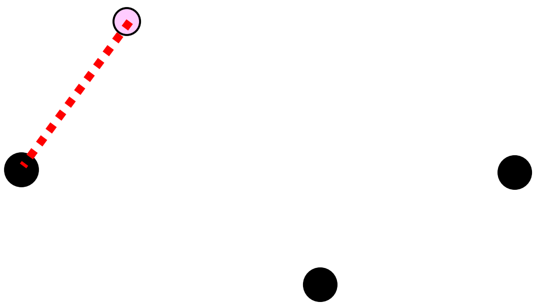


K–means

1. Initialize a set of centroids randomly

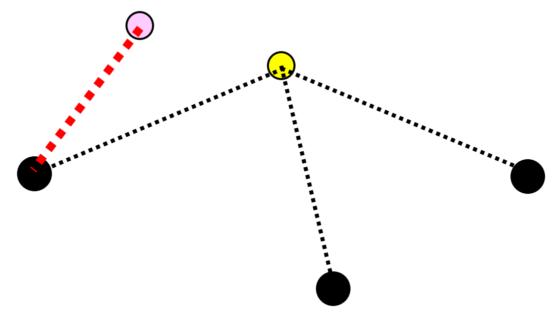
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
    - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$

3. Put data point in the cluster of the closest centroid
    - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



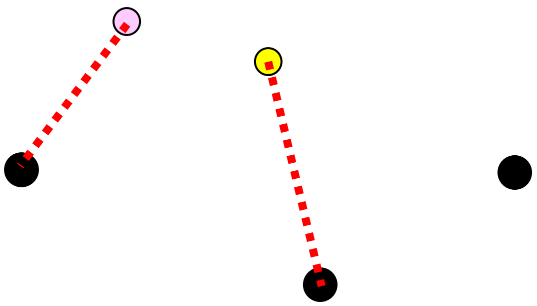
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $$d_{cluster} = \mathbf{distance}(x, m_{cluster})$$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



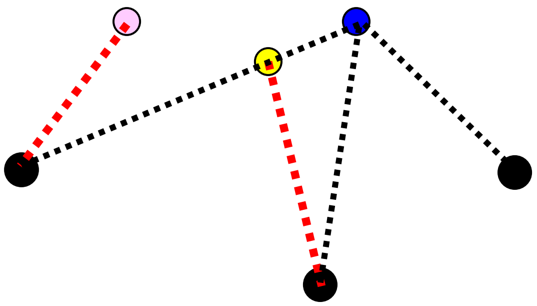
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $$d_{cluster} = \mathbf{distance}(x, m_{cluster})$$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $$d_{cluster} = \mathbf{distance}(x, m_{cluster})$$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



K–means

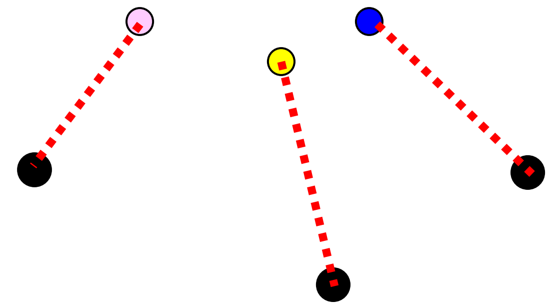
1. Initialize a set of centroids randomly

2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster

   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$

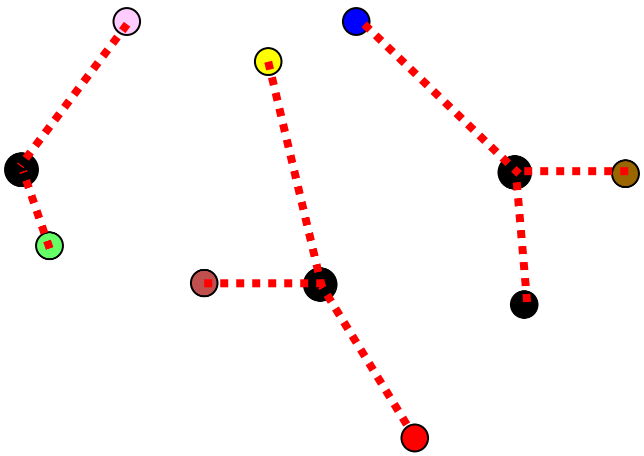
3. Put data point in the cluster of the closest centroid

   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



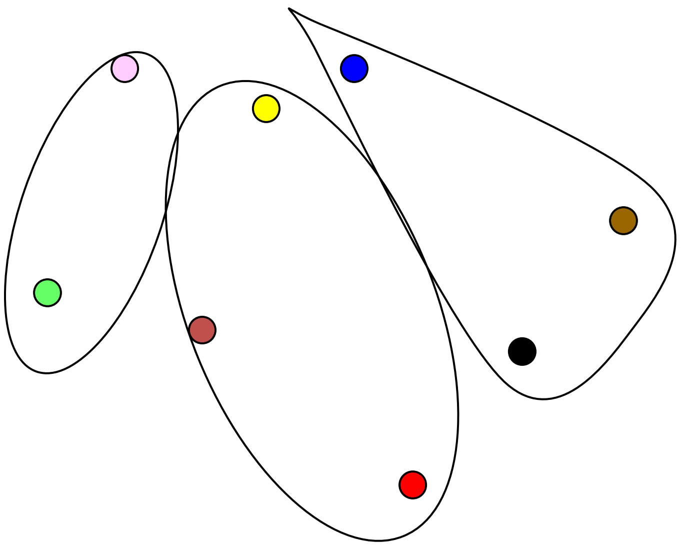
K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $$d_{cluster} = \mathbf{distance}(x, m_{cluster})$$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



K–means

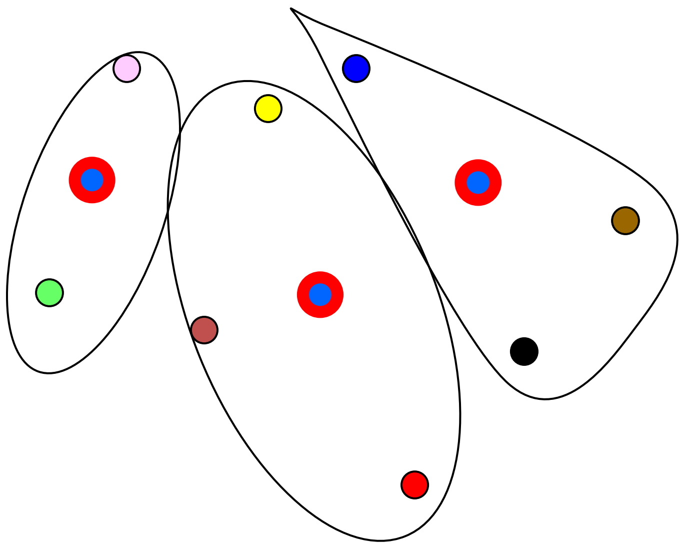
1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum



K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

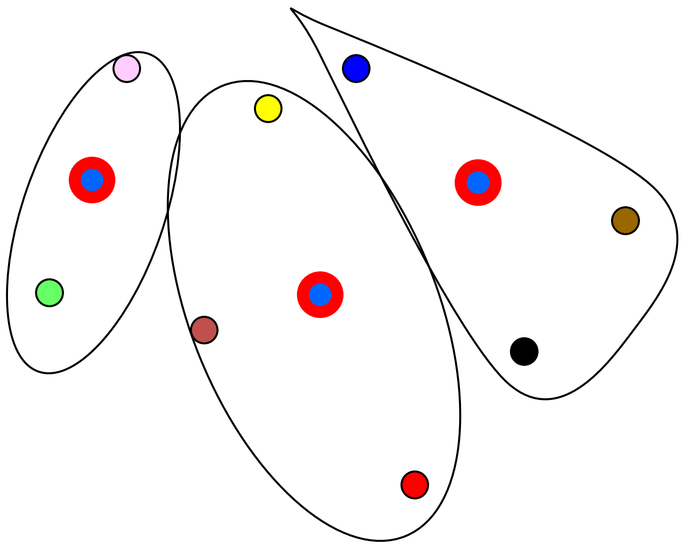


K–means

1. Initialize a set of centroids randomly
2. For each data point $\boldsymbol{x}$, find the distance from the centroid for each cluster
   - $d_{cluster} = \mathbf{distance}(x, m_{cluster})$
3. Put data point in the cluster of the closest centroid
   - Cluster for which $\boldsymbol{d}_{\text{cluster}}$ is minimum
4. When all data points are clustered, recompute centroids

$$m_{cluster} = \frac{1}{\sum_{i \in cluster} w_i} \sum_{i \in cluster} w_i x_i$$

5. If not converged, go back to 2



K-Means comments

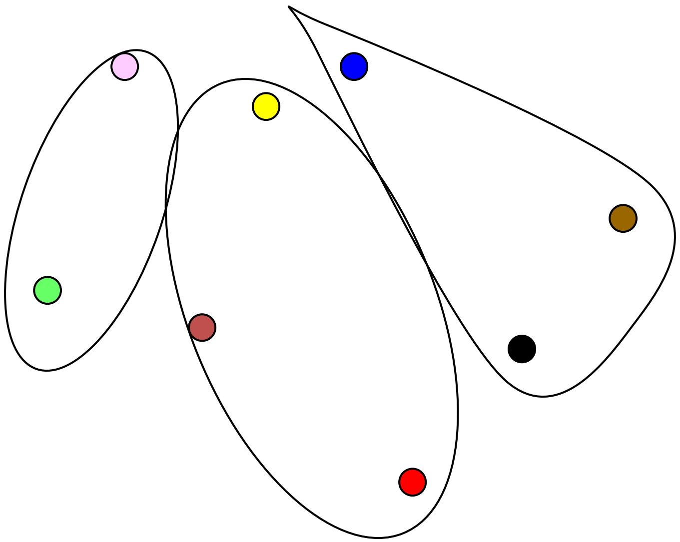
- The distance metric determines the clusters
  - In the original formulation, the distance is L_2 distance

$$\mathbf{distance}_{cluster}(x, m_{cluster}) = \| x - m_{cluster} \|_2$$

$$m_{cluster} = \frac{1}{N_{cluster}} \sum_{i \in cluster} x_i$$

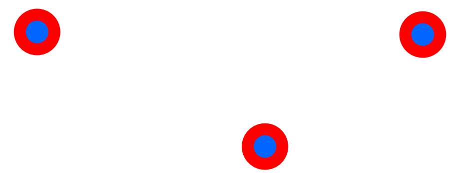
  - If we **replace** every x by $m_{cluster}(x)$, we get *Vector Quantization*
- K-means is an instance of *generalized* EM

The outcome of K-means clustering



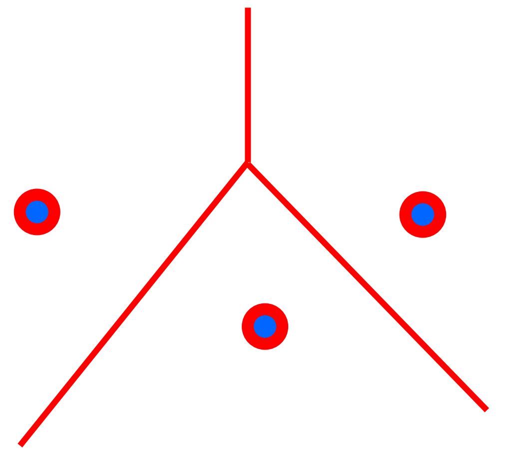
- The clusters

The outcome of K-means clustering



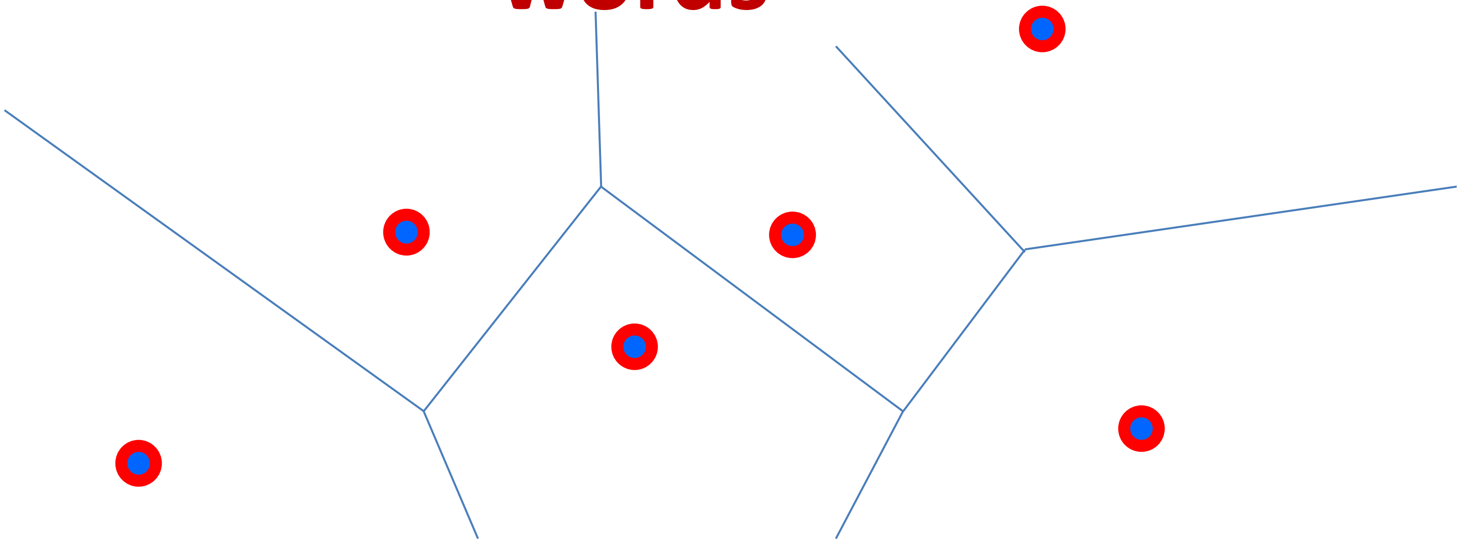
- The clusters
- The *centroids* of the clusters (the "codebook")

The outcome of K-means clustering



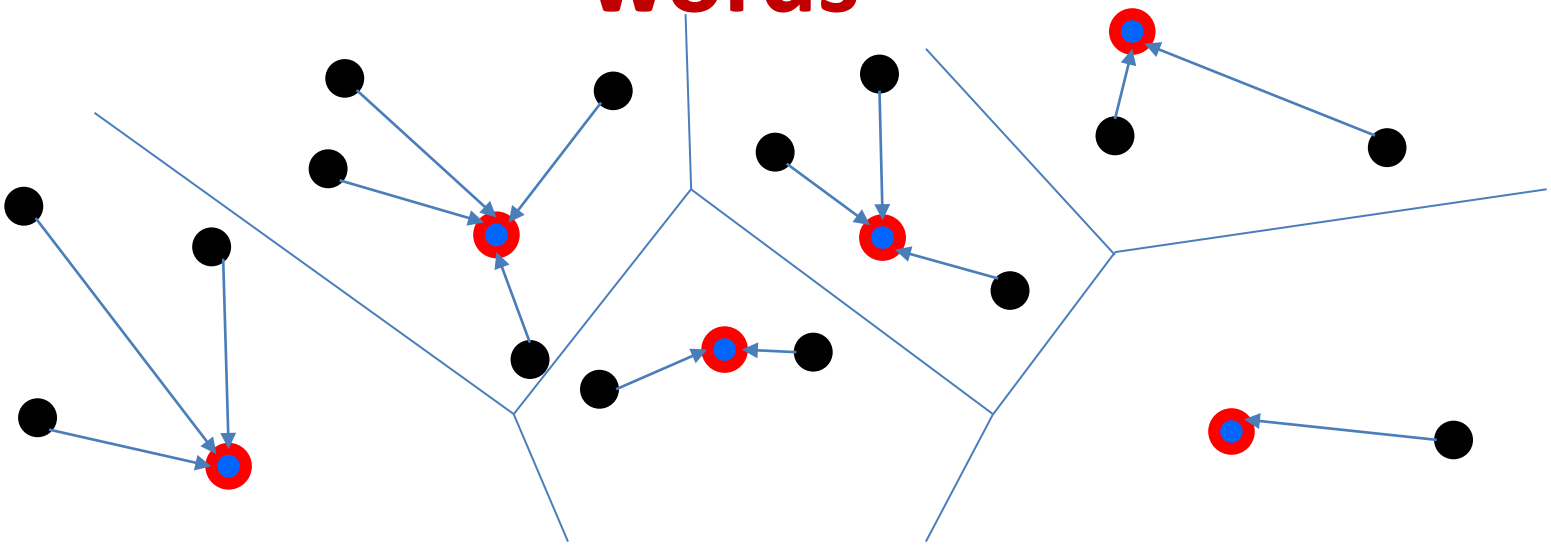
- The clusters
- The *centroids* of the clusters (the "codebook")
- An implicit partitioning of the space into bins, based on which centroid any point in the space is closest to
  - Partitions the space into "voronoi buckets"
  - Which gives us the grid required to compute multi-dimensional histograms
    - AKA "bag of words"

Computing histograma AKA “bag of words”



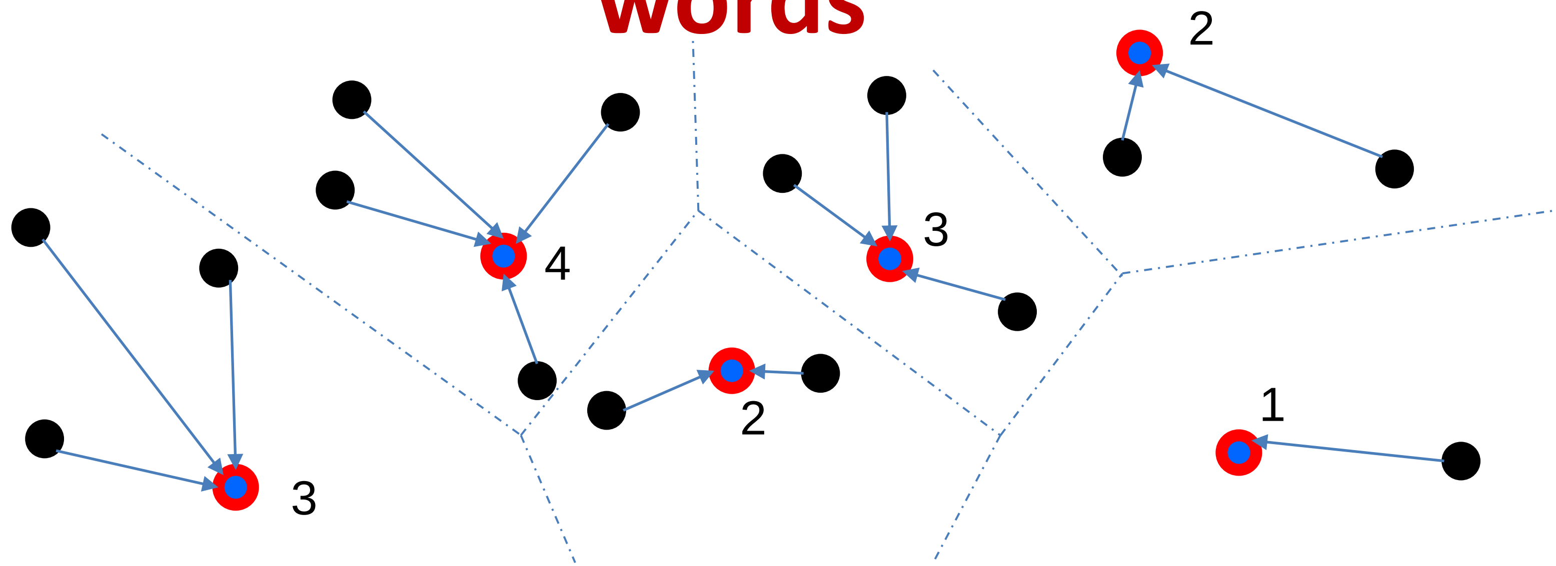
- Generate a codebook by running K-means on some training data
  - Implicitly also gets bin boundaries

Computing histograma AKA "bag of words"



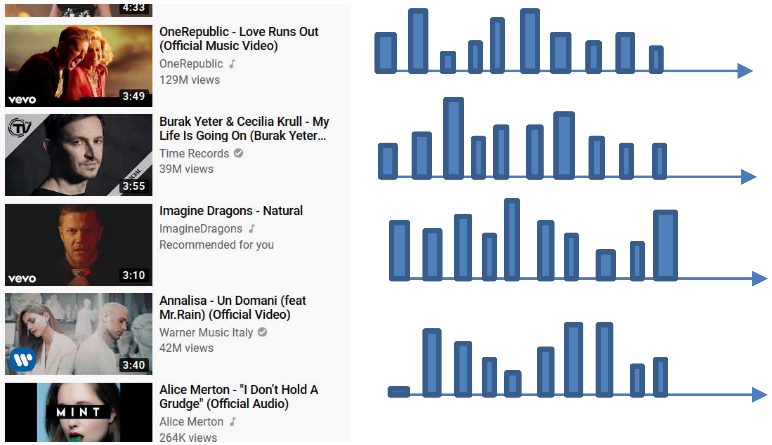
- Generate a codebook by running K-means on some training data
  - Implicitly also gets bin boundaries (not shown)
- On test data recording, replace every vector by the closest code word
  - Effectively replaces every vector by the representative of the bin it falls in

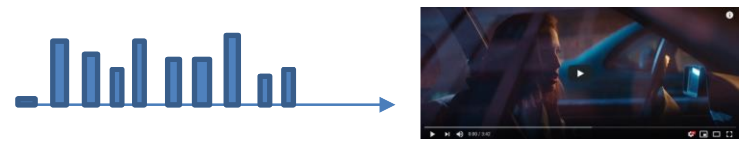
Computing histograma AKA “bag of words”



- Generate a codebook by running K-means on some training data
  – Implicitly also gets bin boundaries (not shown)
- On test data recording, replace every vector by the closest code word
  – Effectively replaces every vector by the representative of the bin it falls in
- Compute a histogram of the number of times each codeword has occurred in the data
  – Gives you the “bag of words” representation for the recording

How?





- Train a codebook
- Convert every entry in your "gallery" to a histogram
- Compute a histogram for your "query"
- Sort gallery entries by the similarity of their histograms to the query histogram
  - Similarity: KL-divergence, L2-divergence, or any other suitable metric

So, how to do this?

- How to retrieve all music videos by this guy?
  - Assuming you have converted all videos on youtube to a sequence of feature vectors
  - And have this one example recording (also converted to a feature vector sequence)

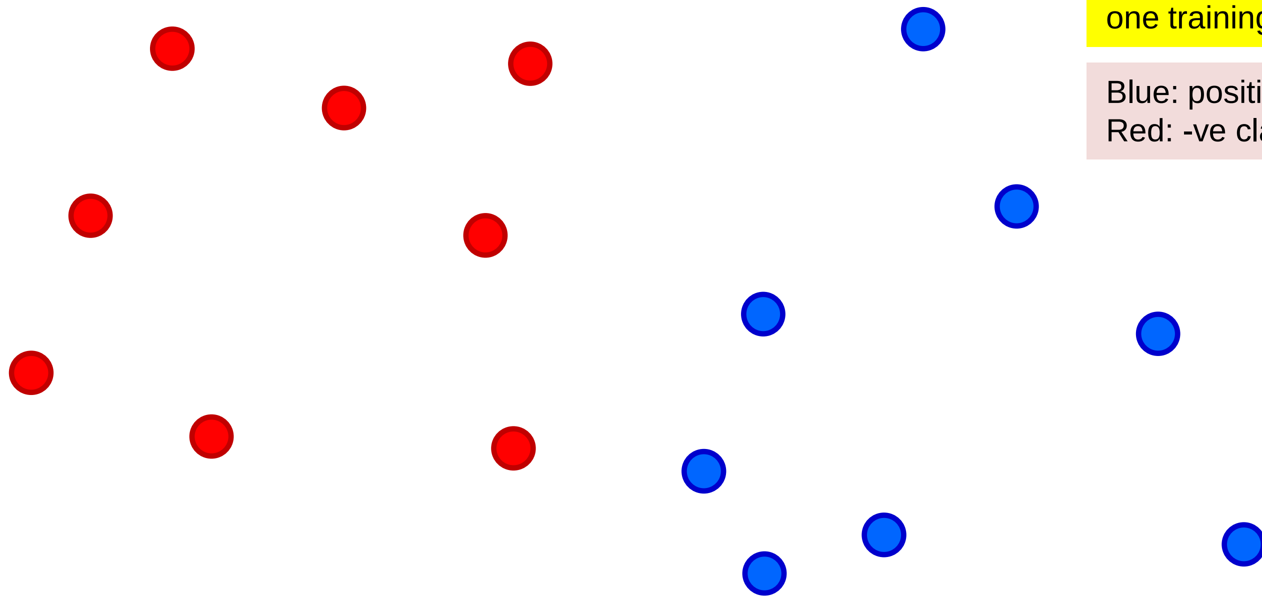
A different problem

- You are given a collection of birthday party videos and videos that are not of birthday parties
- You are given a new test video
- Must decide if this is a birthday party
- This is a problem of *classification*

Classification

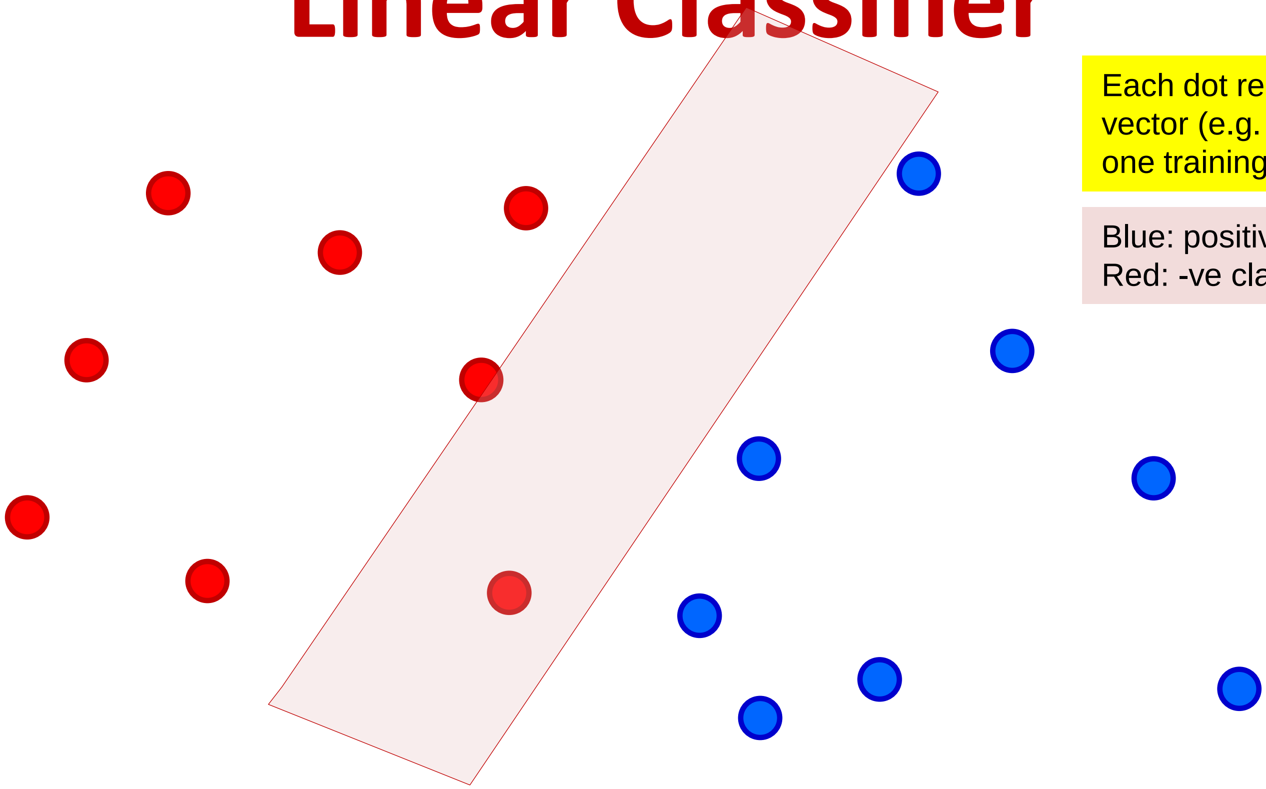
Each dot represents a feature vector (e.g. a BOW vector) of one training instance (recording)

Blue: positive class,
Red: -ve class



- Given a bunch of +ve and –ve training instances (representing a positive class, e.g. birthday, and a negative class, e.g. not-birthday),
  - Find a rule that correctly assigns a new test instance to one of the two classes

Linear Classifier

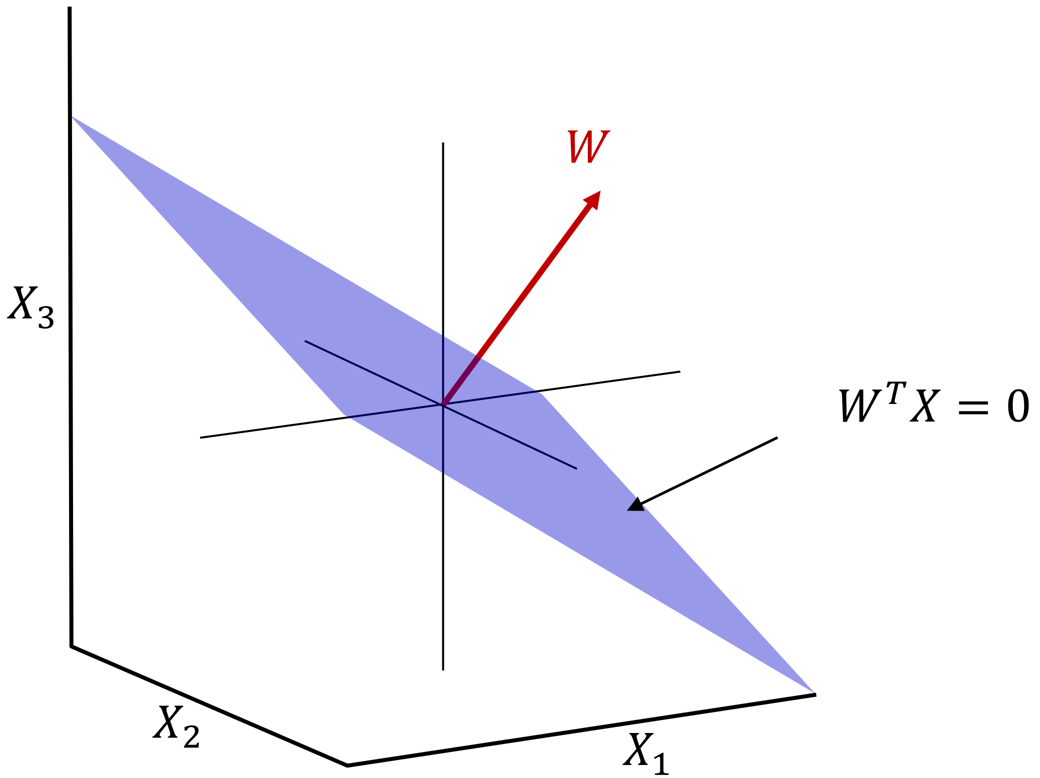


Each dot represents a feature vector (e.g. a BOW vector) of one training instance (recording)

Blue: positive class,
Red: -ve class

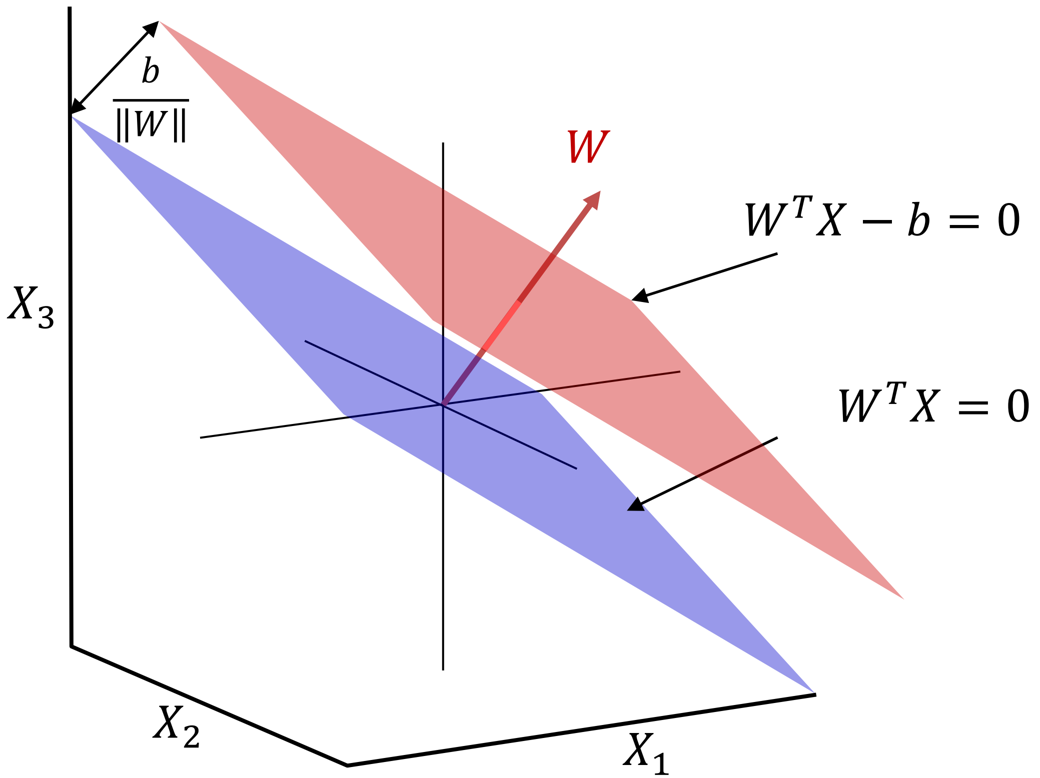
- Initially assume that the classes are separable by a hyperplane
  - A *linear* classifier
- Also that the training data are perfectly separable by the hyperplane
- We will fix these assumptions later

The equation for a hyperplane



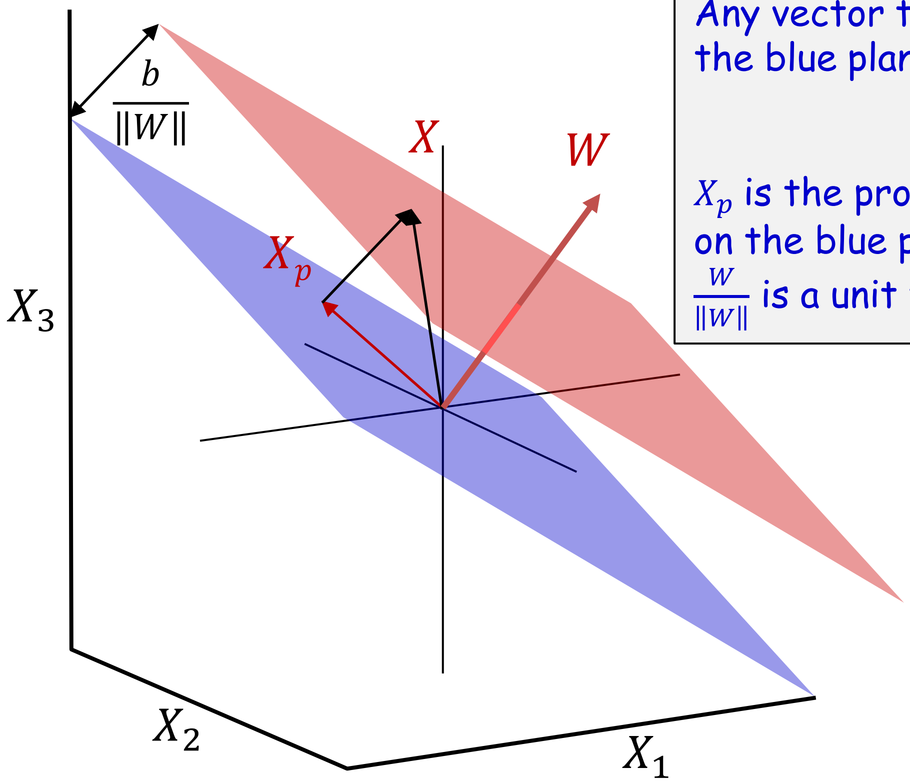
- $W^T X = 0$ is the equation representing the set of all vectors that are orthogonal to W

The equation for a hyperplane



- $W^T X - b = 0$ is the equation representing plane that is orthogonal to W and a distance $\frac{b}{\|W\|}$ from origin
  - The set of all vectors that are a distance $\frac{b}{\|W\|}$ from the blue plane

The equation for a hyperplane



Any vector that is a distance d from the blue plane can be written as

$$X = X_p + d\frac{W}{\|W\|}$$

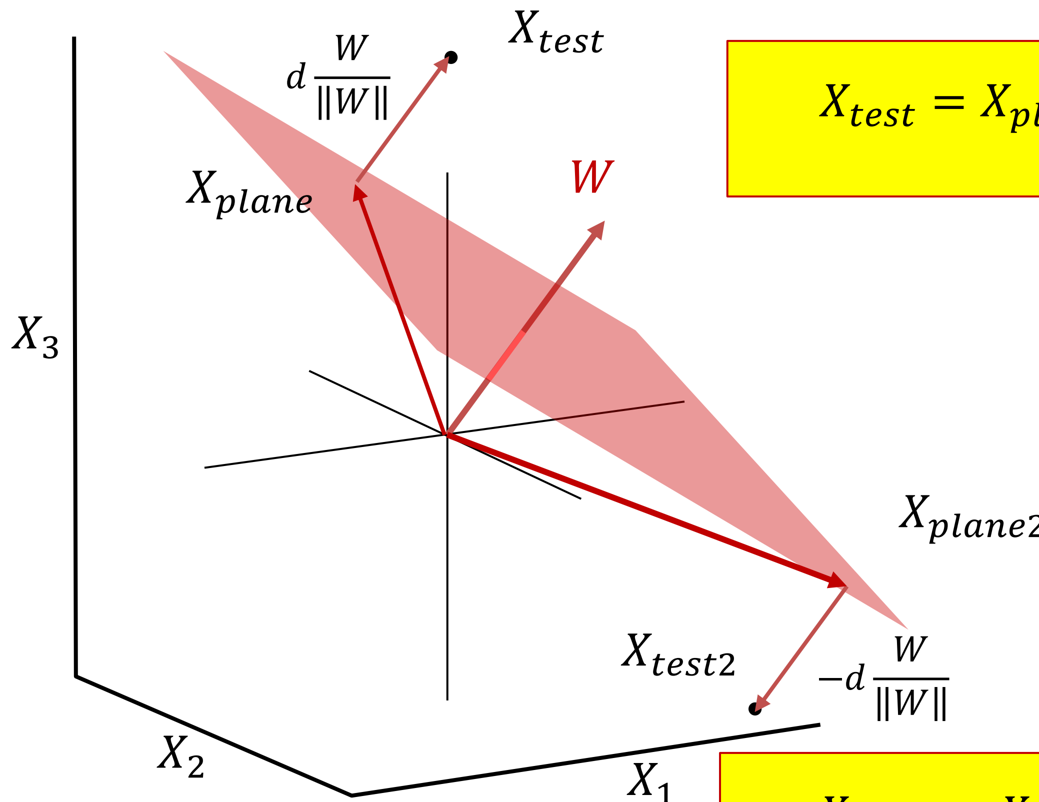
X_p is the projection of the vector on the blue plane

$\frac{W}{\|W\|}$ is a unit vector in the direction of W

Trivial proof:

- On the red plane any $X = X_p + \left(\frac{b}{\|W\|}\right)\frac{W}{\|W\|}$
- $W^T X = W^T X_p + b\frac{W^T W}{\|W\|^2} = b$

Distance from a hyperplane

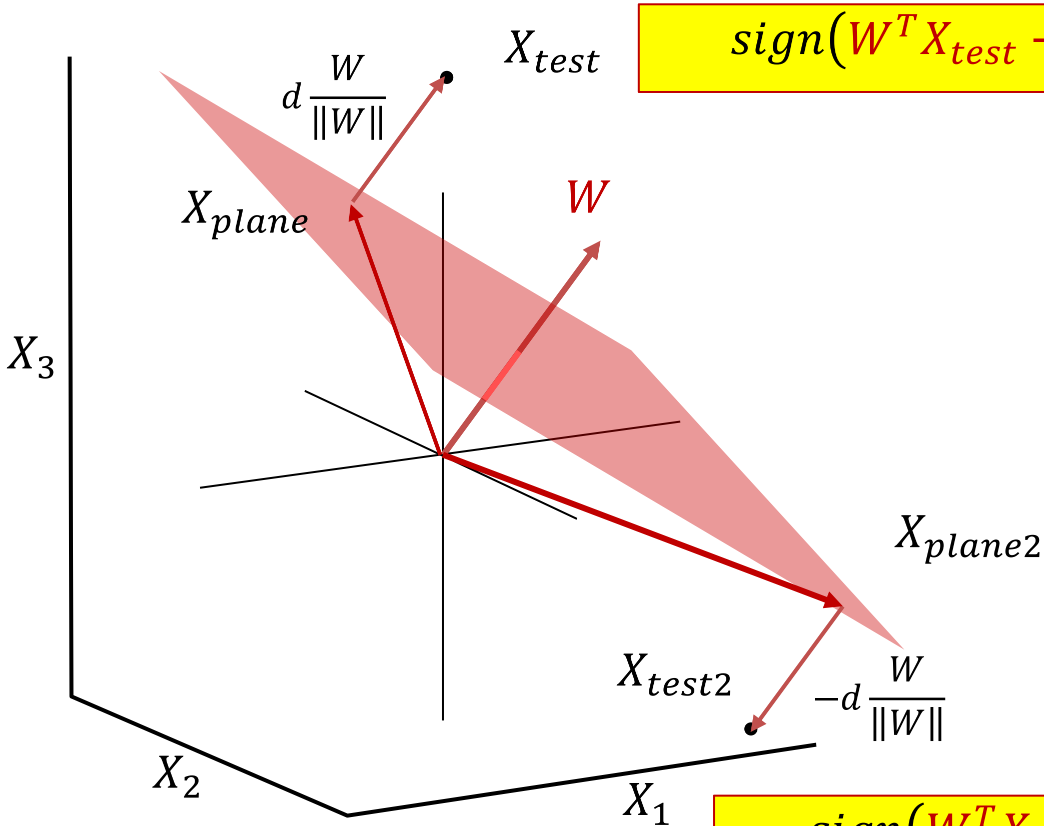


$$X_{test} = X_{plane} + d\frac{W}{\|W\|}$$

$$X_{test2} = X_{plane2} - d\frac{W}{\|W\|}$$

- The distance of any X_{test} from the plane $W^TX - b = 0$ is $d = \frac{W^TX_{test} - b}{\|W\|}$
- This can be positive (in the direction of W) or negative (opposite to W)

Sign of distance from hyperplane

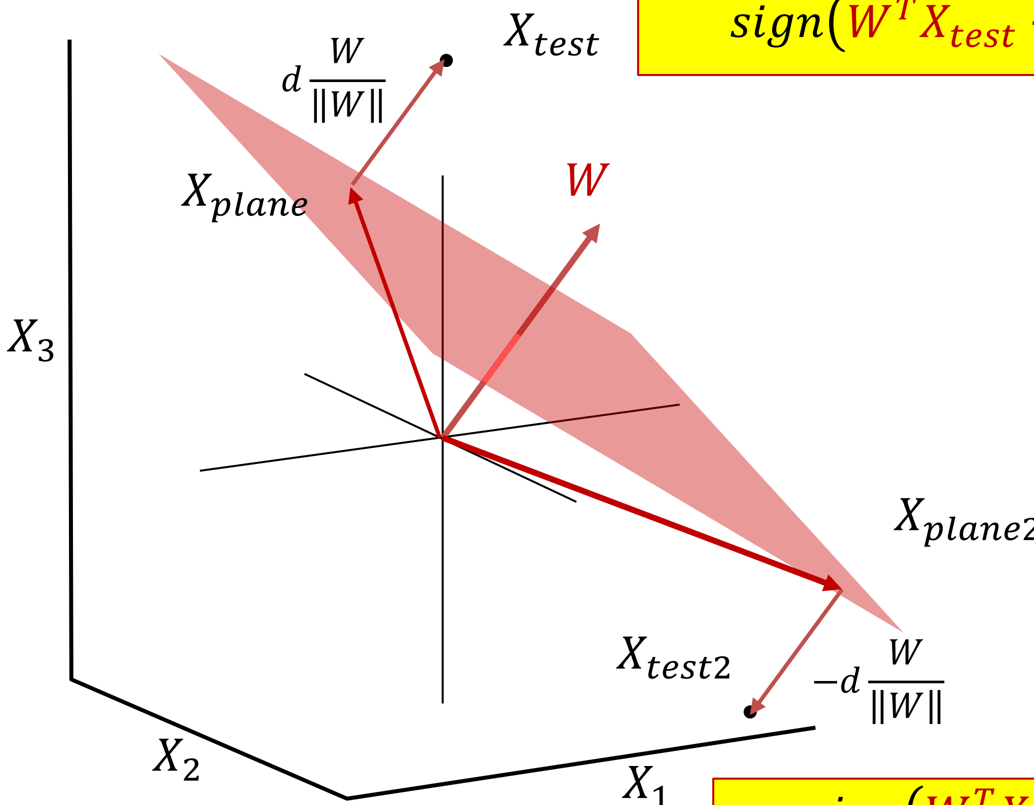


$sign(W^T X_{test} - b) > 0$

$sign(W^T X_{test2} - b) < 0$

- The sign of $W^T X - b$ signifies which side of the plane the point X is on

Linear Classifier

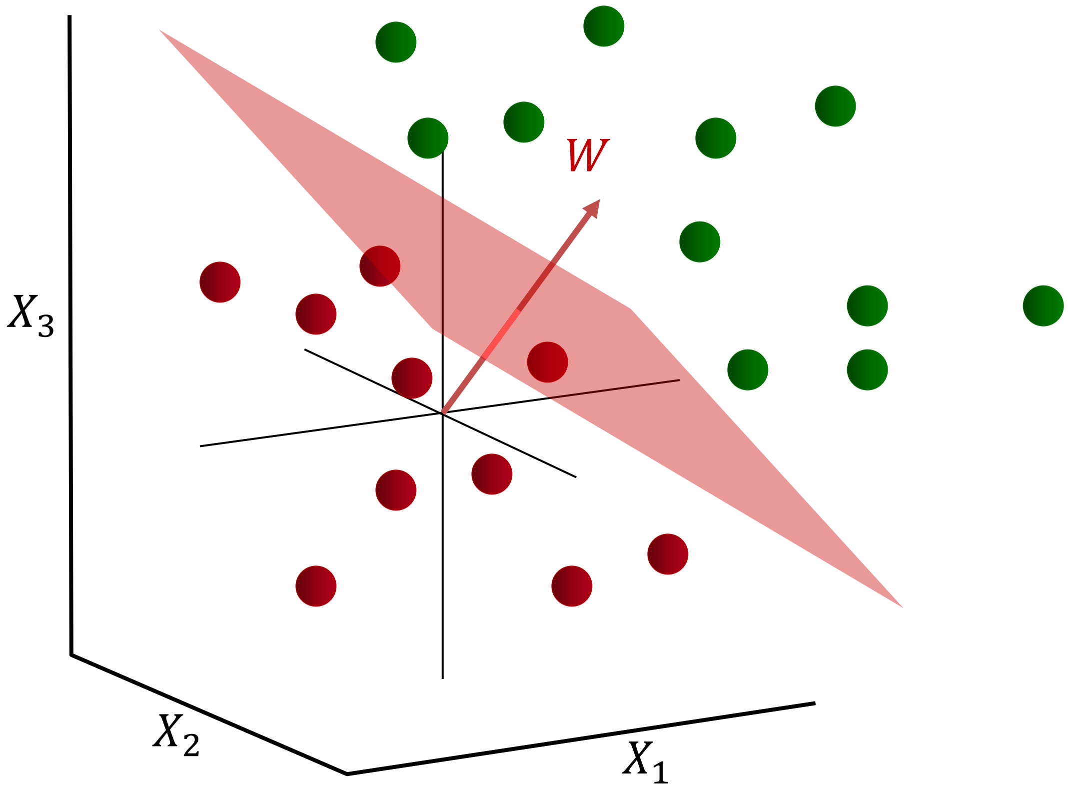


$$\boxed{sign\left(W^T X_{test} - b\right) > 0}$$

$$\boxed{sign\left(W^T X_{test2} - b\right) < 0}$$

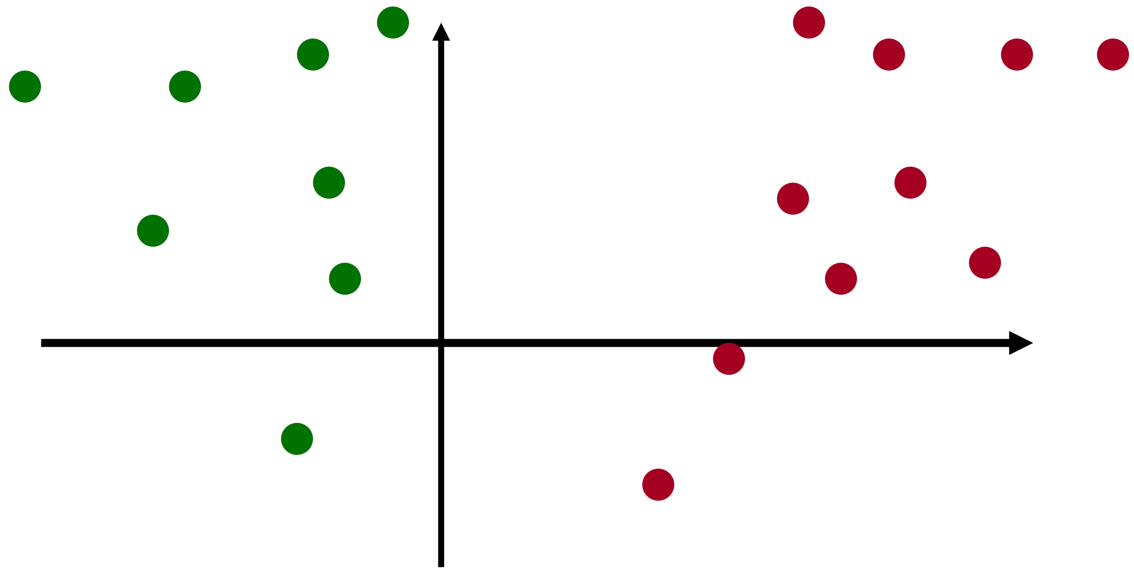
- The plane $W^T X - b$ is a linear classifier
  - The class is given by $sign\left(W^T X_{test} - b\right)$

Linearly separable data



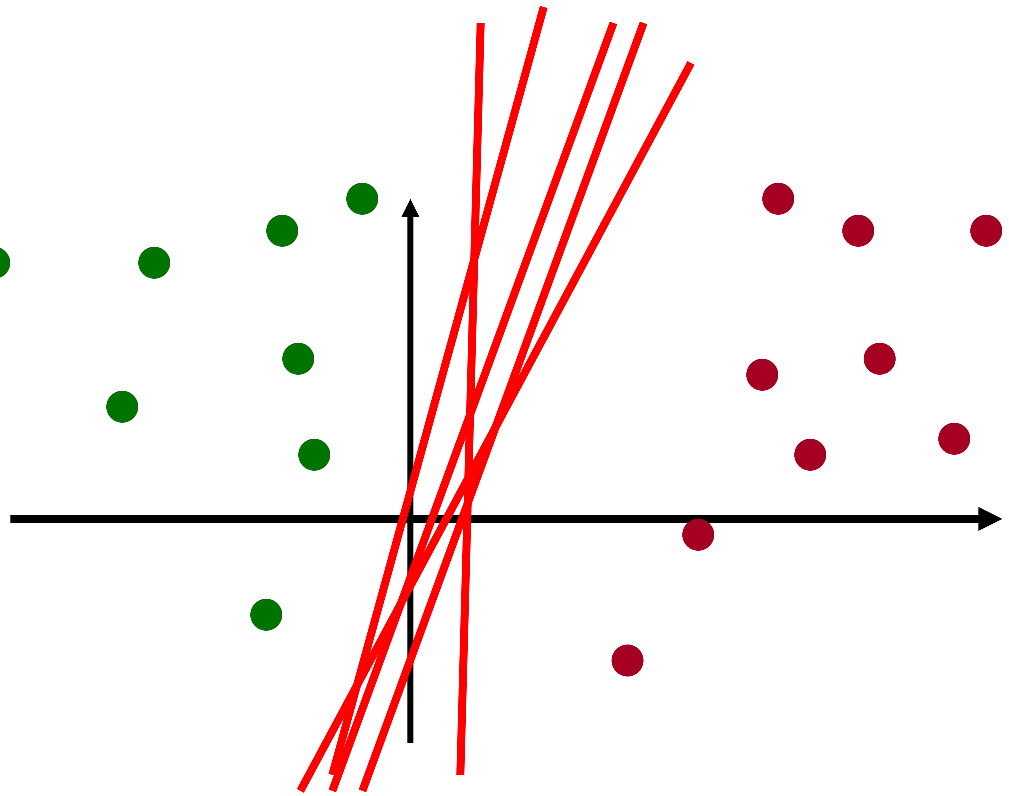
- Data where the two classes are separated by a hyperplane
  - And classification can be performed by $sign\left(W^T X_{test} - b\right)$ for any separating hyperplane

2D illustration, linearly separable data



- Classes are linearly separable
- Dots represent “training” instances
- **Training problem**: Given these training instances find a separating hyperplane

The separating hyperplane

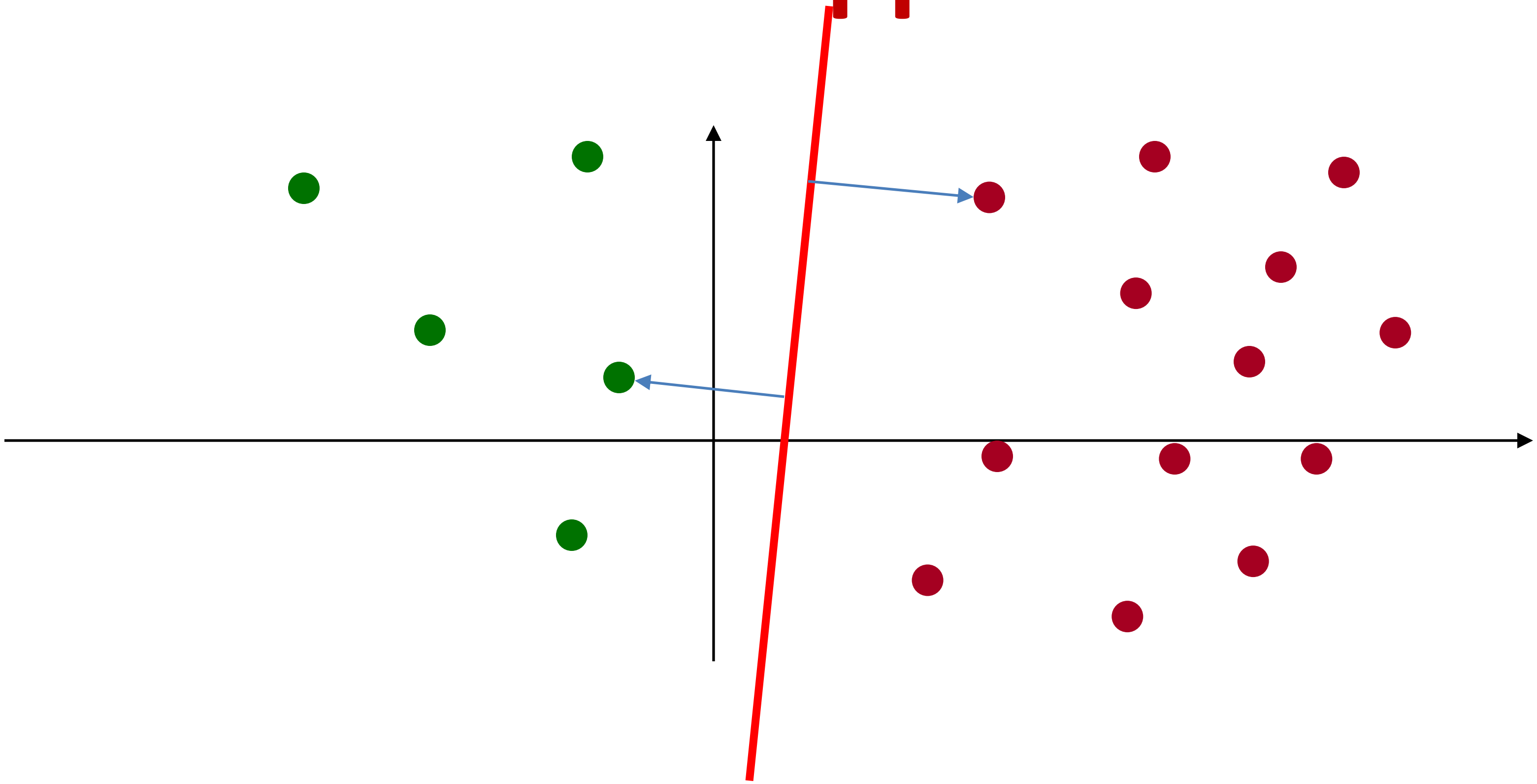


- Problem: Given these training instances find a separating hyperplane
- **Many ways of finding this hyperplane**
  - Any number of solution algorithms are possible

Enter: Support Vector Machines

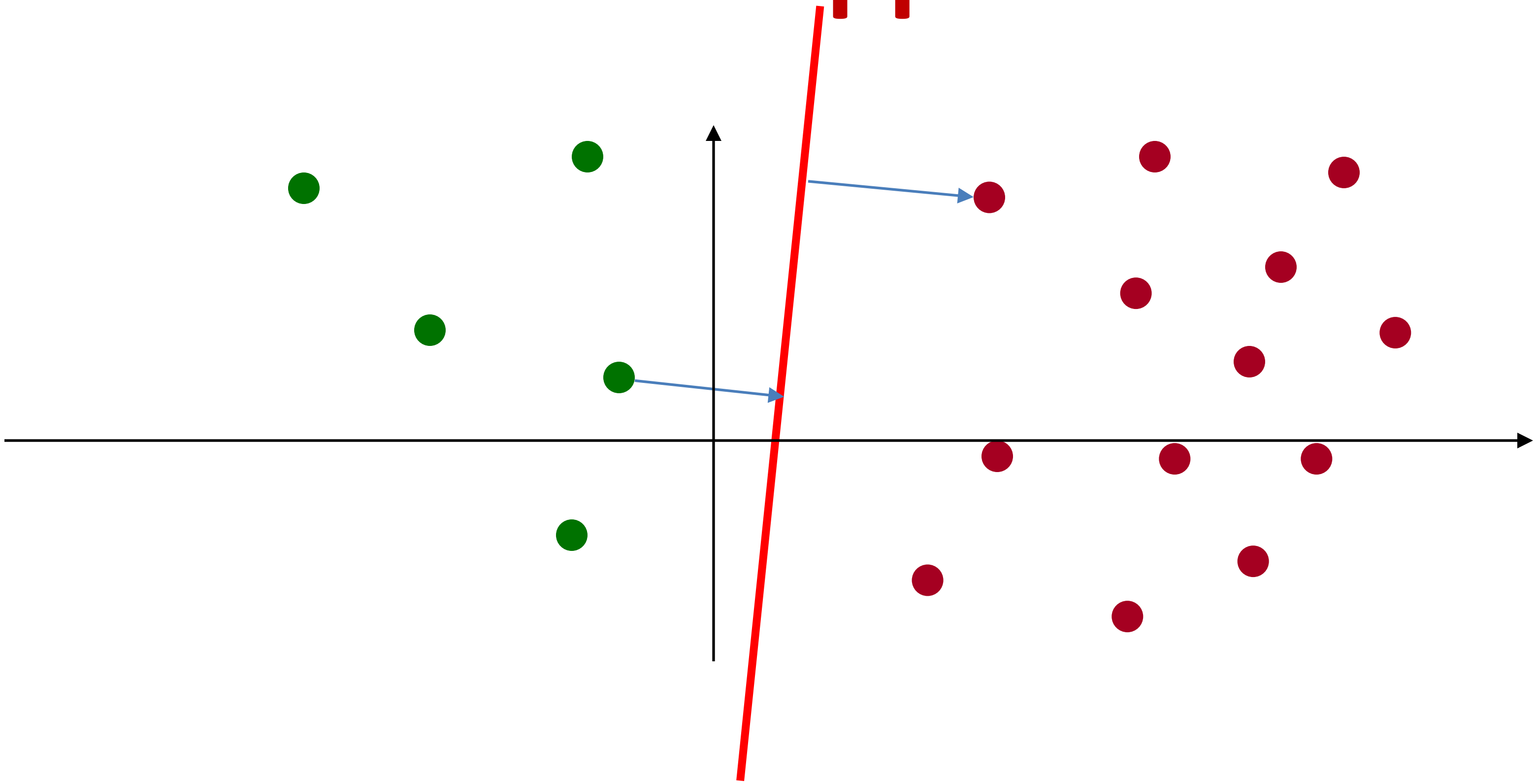
- Find a classifier that is maximally distant from the *closest* instances from either class

A Better Approach



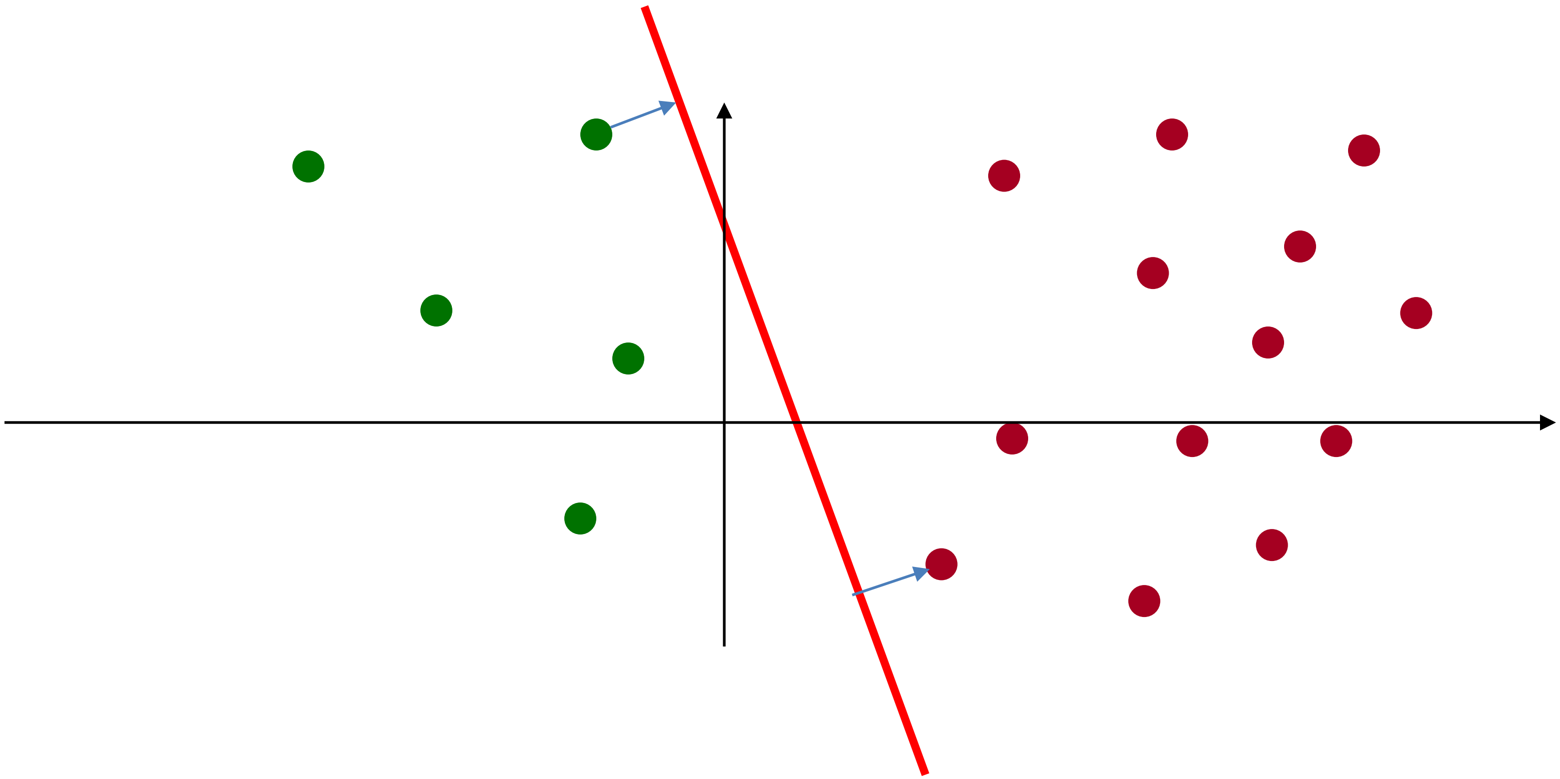
- Any linear classifier has some *closest* instances
- These instances will be at some distance from the boundary
- Changing the classifier will change both, the closest instance, and their distance from the boundary

A Better Approach



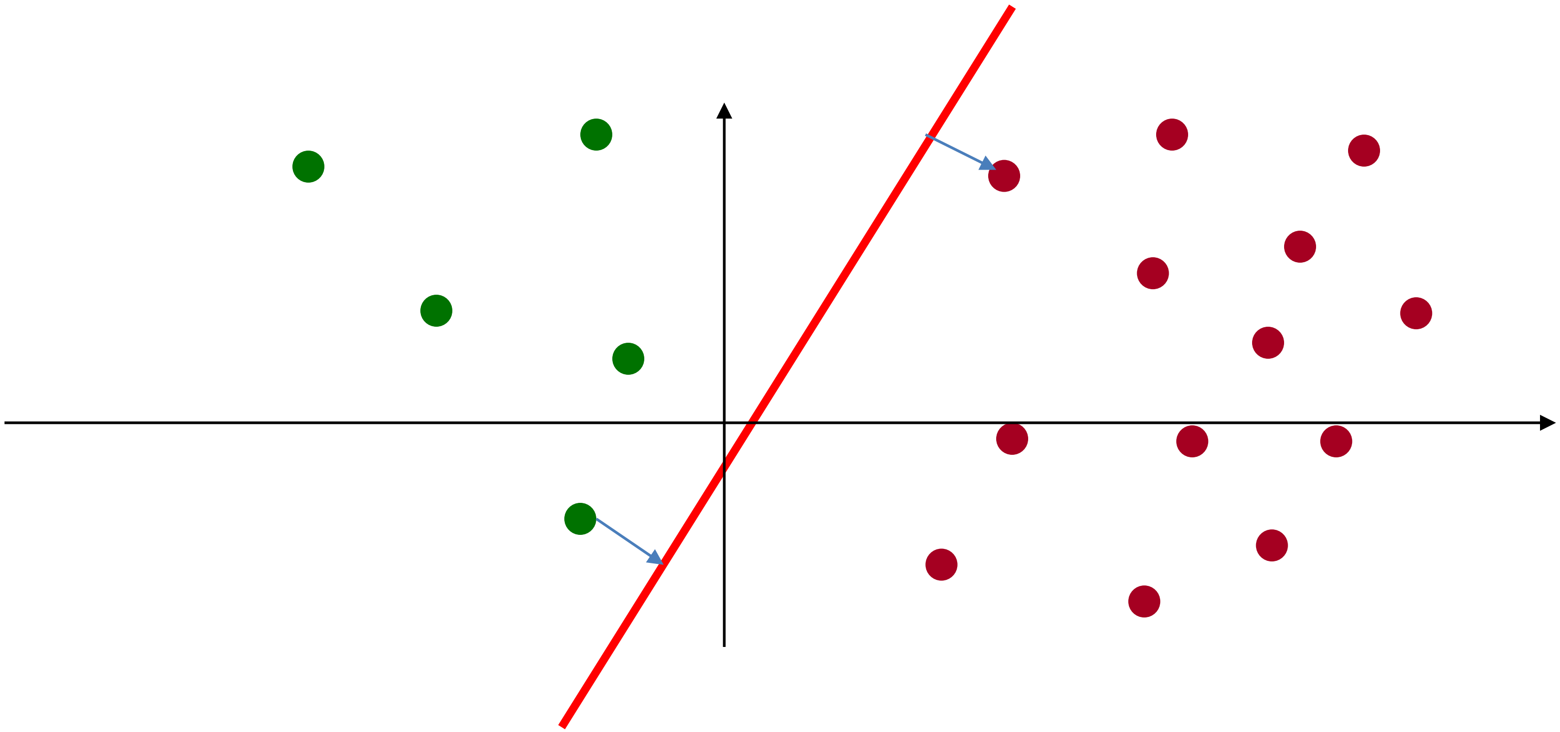
- Search through all classifiers such that the distance to the closest points is maximized
  - Very conservative
  - Focuses on *worst-case* scenario
  - Maximizes the chance that the classifier will work well on new unseen data

A Better Approach



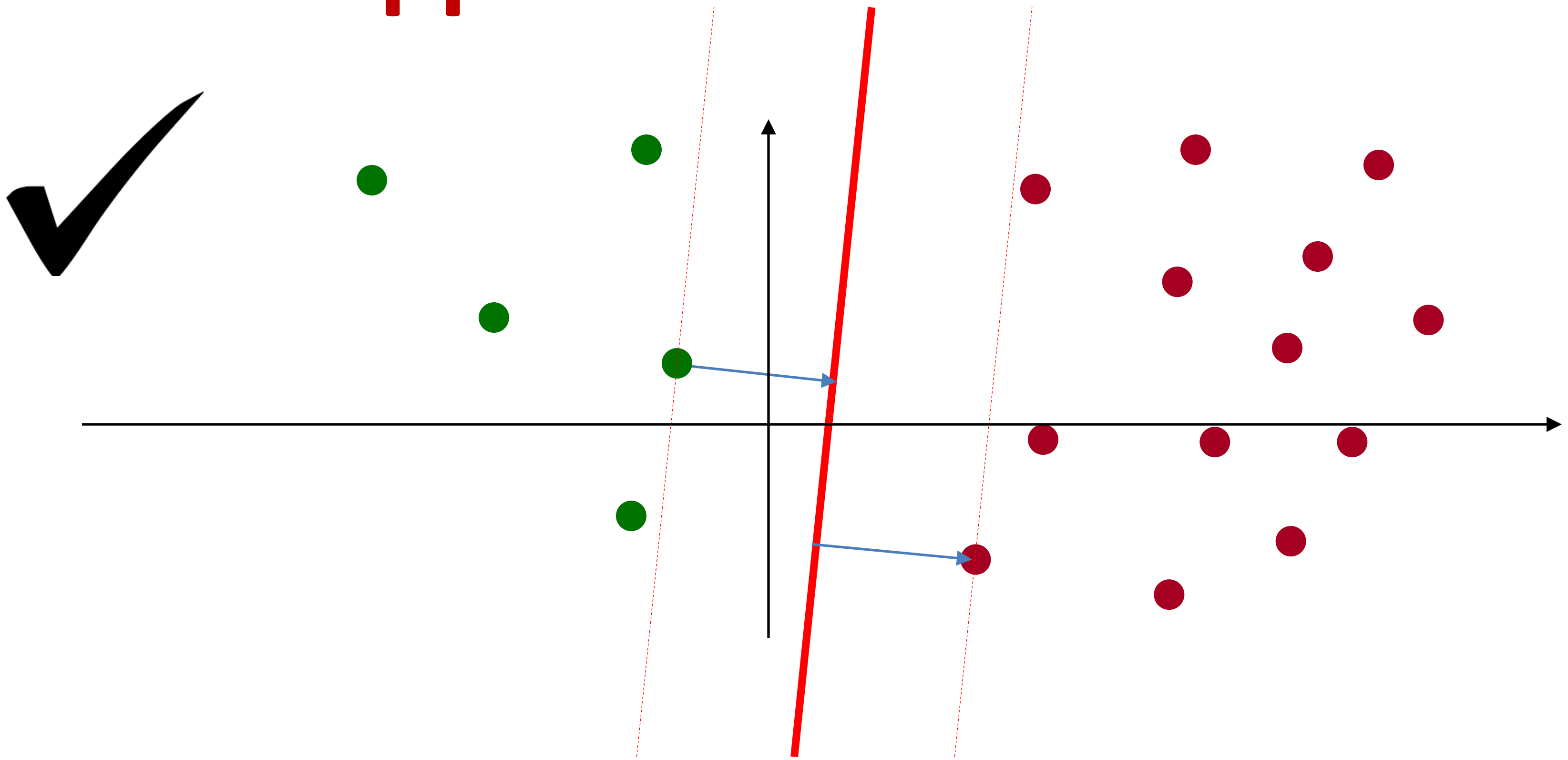
- Search through all classifiers such that the distance to the closest points is maximized
  - Very conservative
  - Focuses on *worst-case* scenario
  - Maximizes the chance that the classifier will work well on new unseen data

A Conservative Approach



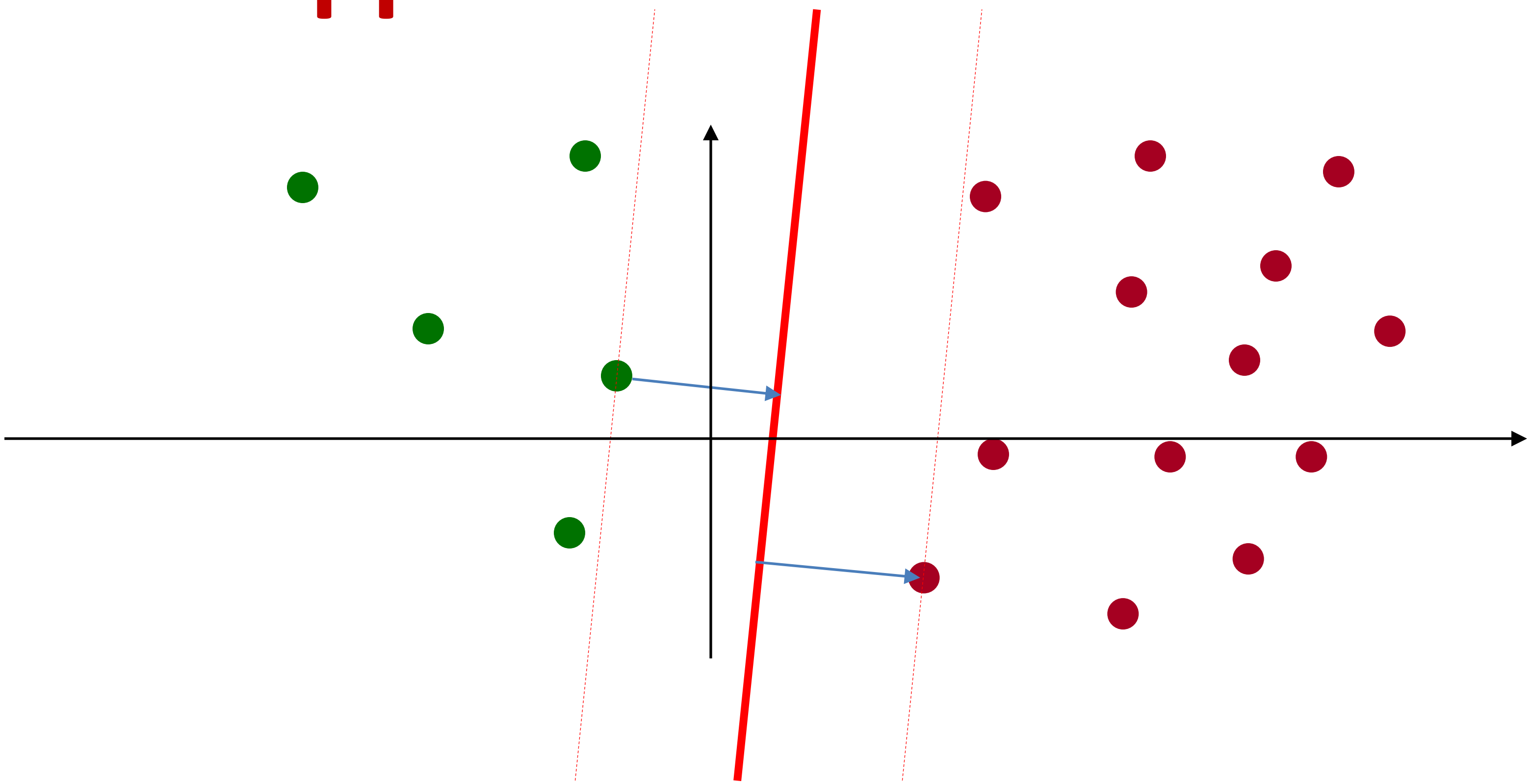
- Search through all classifiers such that the distance to the closest points is maximized
  - Very conservative
  - Focuses on *worst-case* scenario
  - Maximizes the chance that the classifier will work well on new unseen data

Support Vector Machine



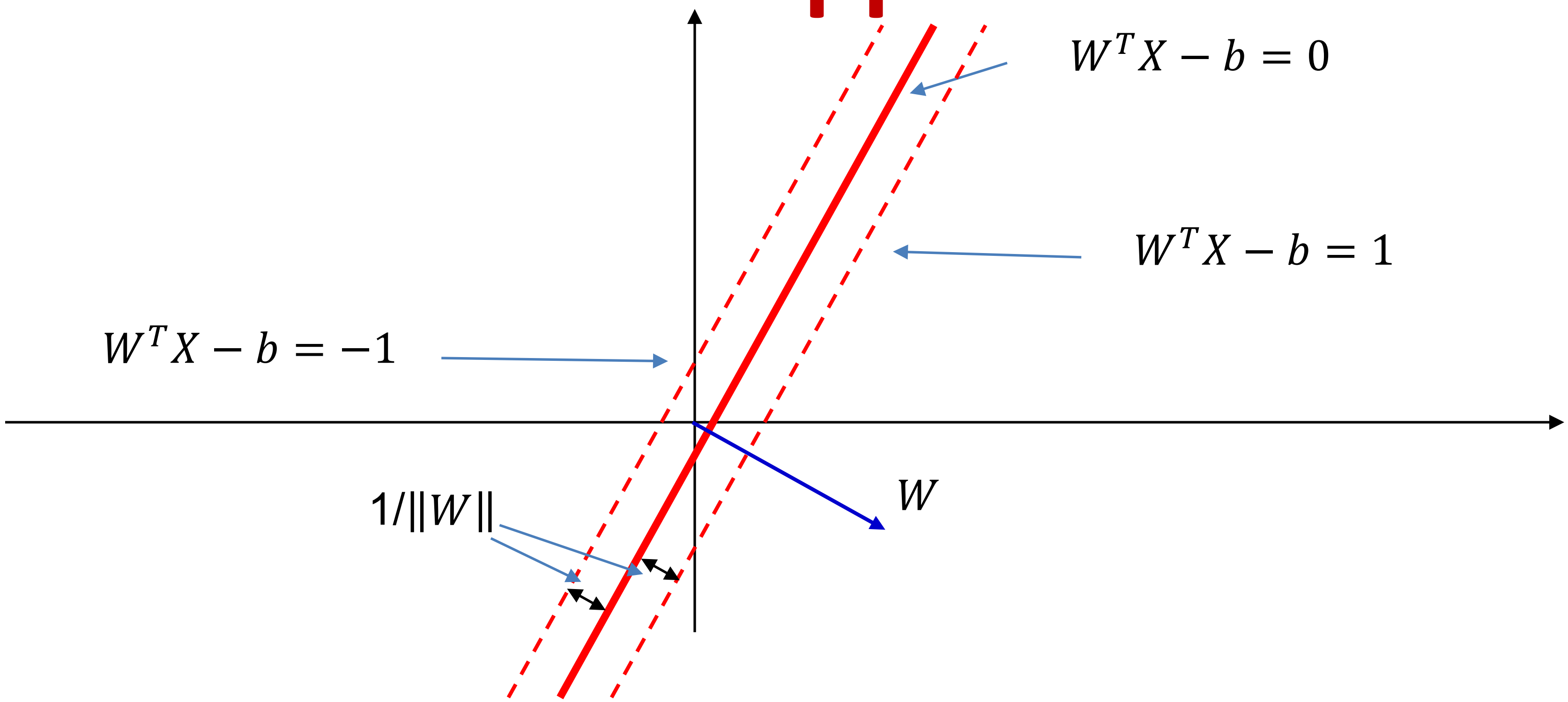
- Search through all classifiers such that the distance to the closest points is maximized
  - Very conservative
  - Focuses on *worst-case* scenario
  - Maximizes the chance that the classifier will work well on new unseen data

Support Vector Machine



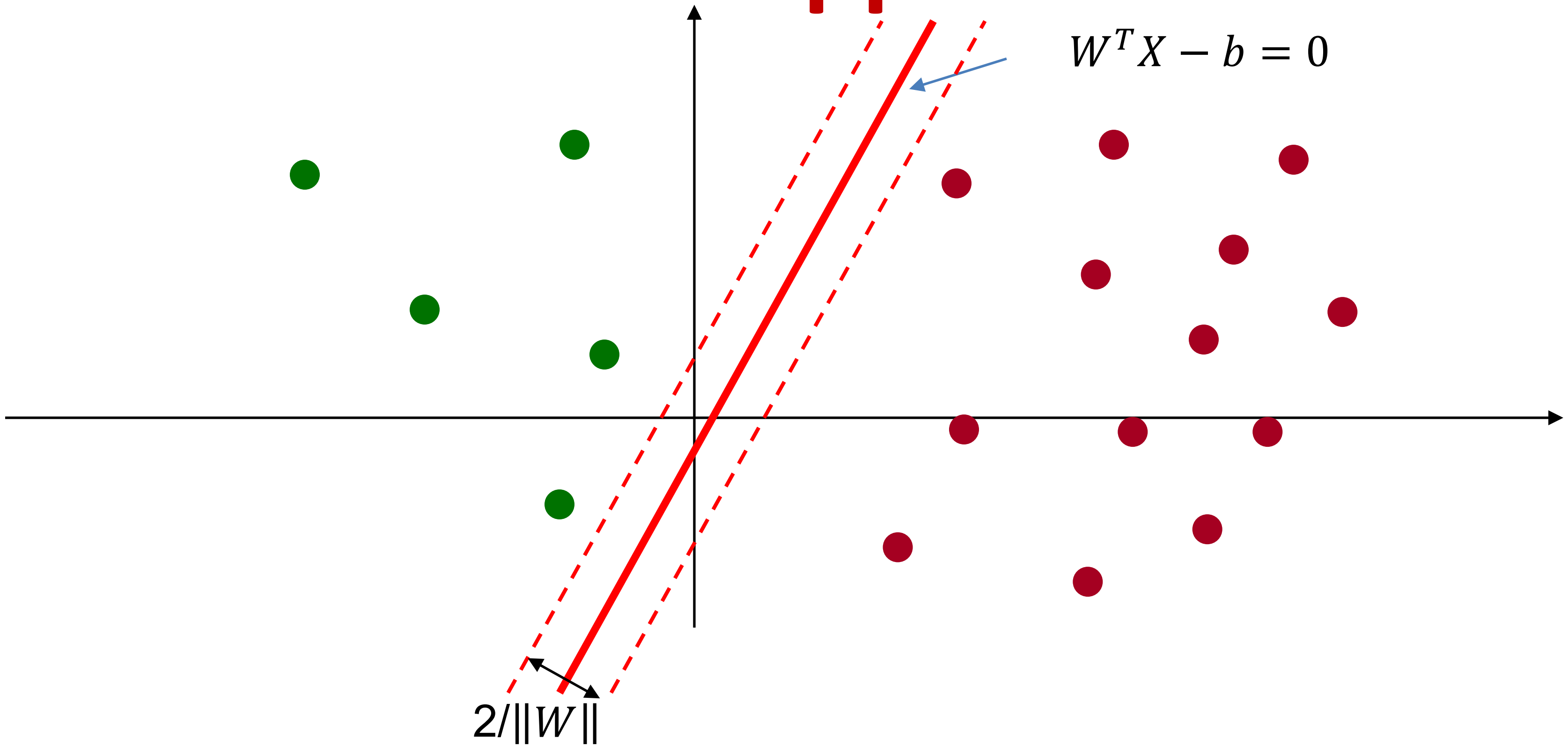
- Find the classifier such that the distance to the *closest* points is maximized
- I.e. solve *two* problems: find the closest points, and the classifier, such that the distance is maximum
  - Position the classifier in the *middle* so that the distance to the closest green = distance to the closest red
- Is this a combinatorial optimization problem??

Solution Approach



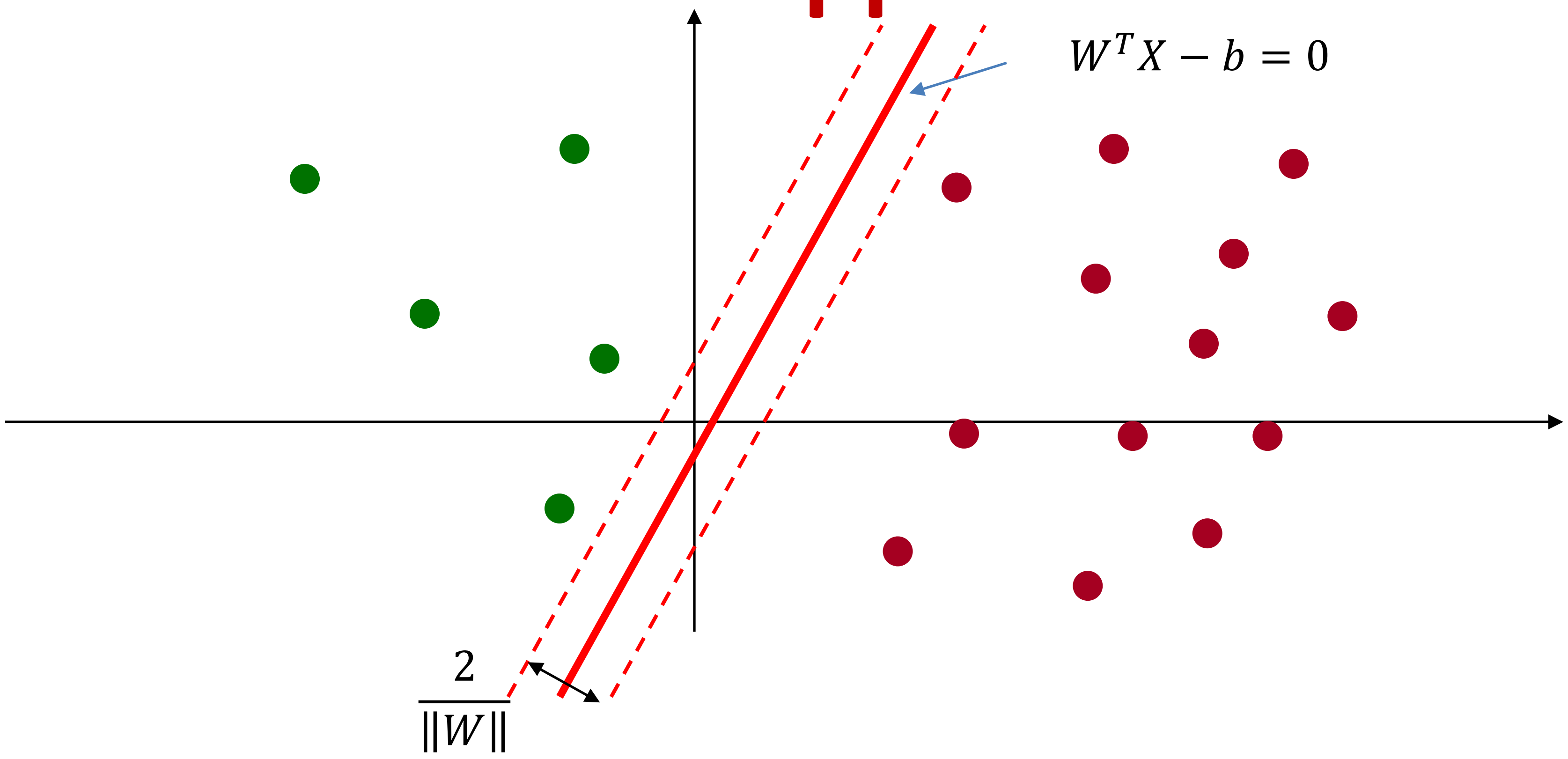
- For any hyperplane (linear classifier) $W^T X - b = 0$
- Choose two hyperplanes $W^T X - b = 1$ and $W^T X - b = -1$
  - The distance of these hyperplanes from the classifier is $1/||W||$
  - The total distance between the hyperplanes is $2/||W||$

Solution Approach



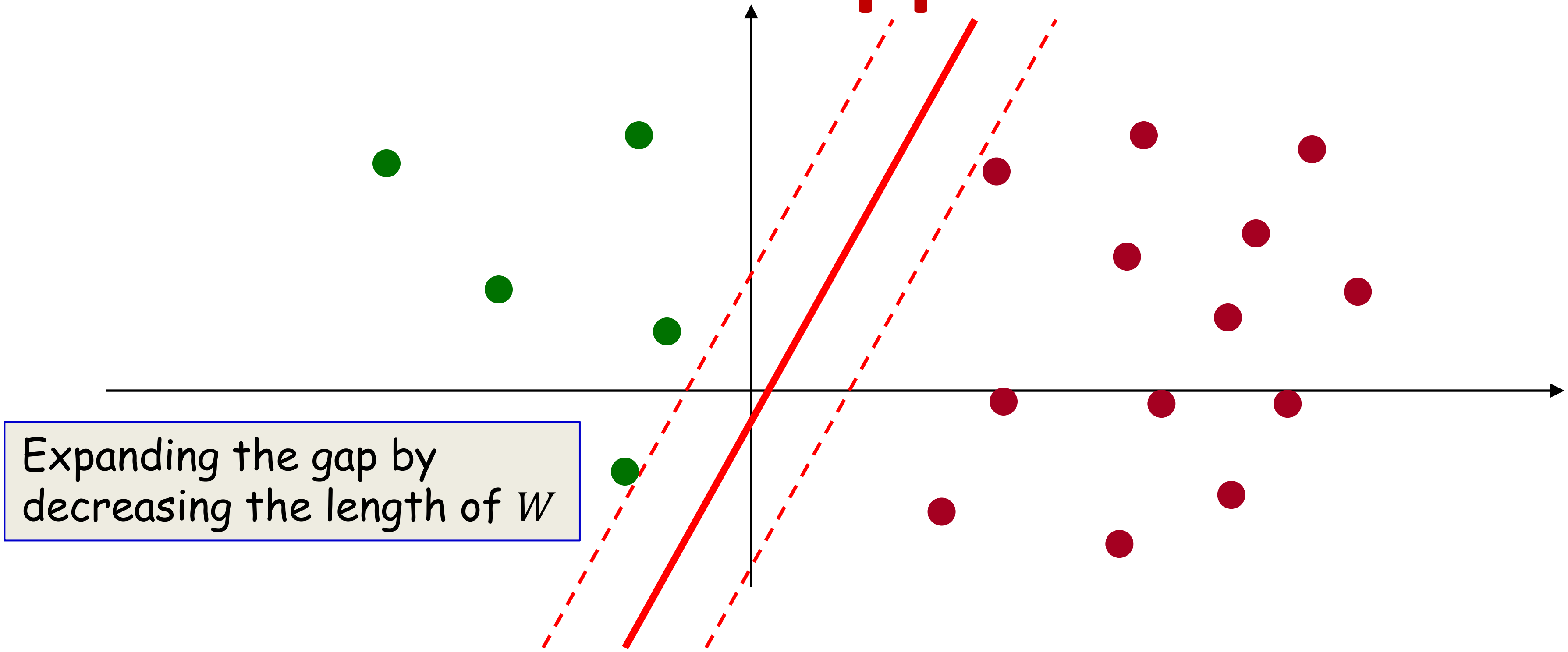
- Constraint: Perfect classification with a margin
- Choose the hyperplanes such that
  - All positive points are on the positive side of the positive hyperplane
  - All negative points are on the negative side of the negative hyperplane

Solution Approach



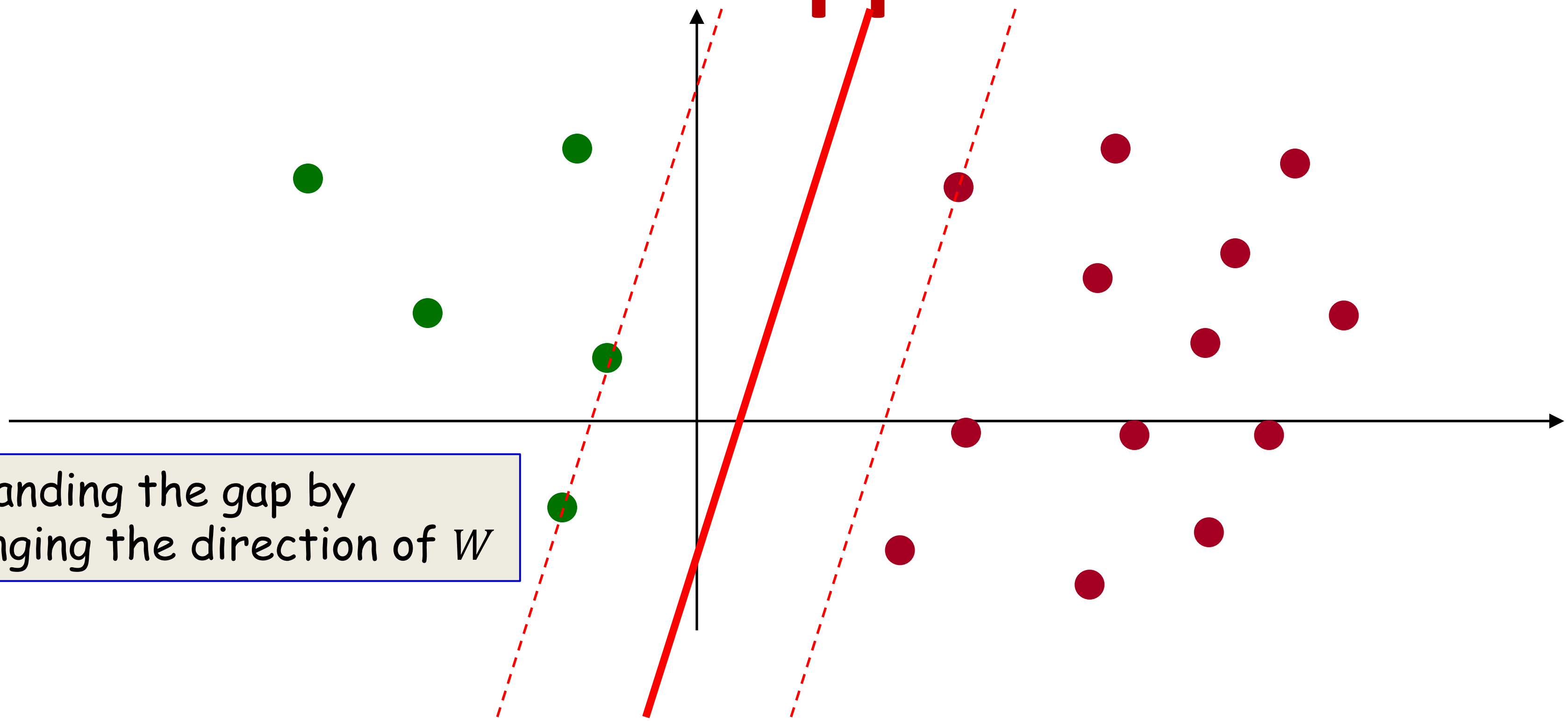
- The distance between the hyperplanes is $\frac{2}{\|W\|}$
- Find the W (and b) such that this is maximized, while maintaining the constraint that all training points are on the "outside" of the appropriate hyperplane

Solution Approach



- The distance between the hyperplanes is $\frac{2}{\|W\|}$
- Find the W (and b) such that this is maximized, while maintaining the constraint that all training points are on the "outside" of the appropriate hyperplane

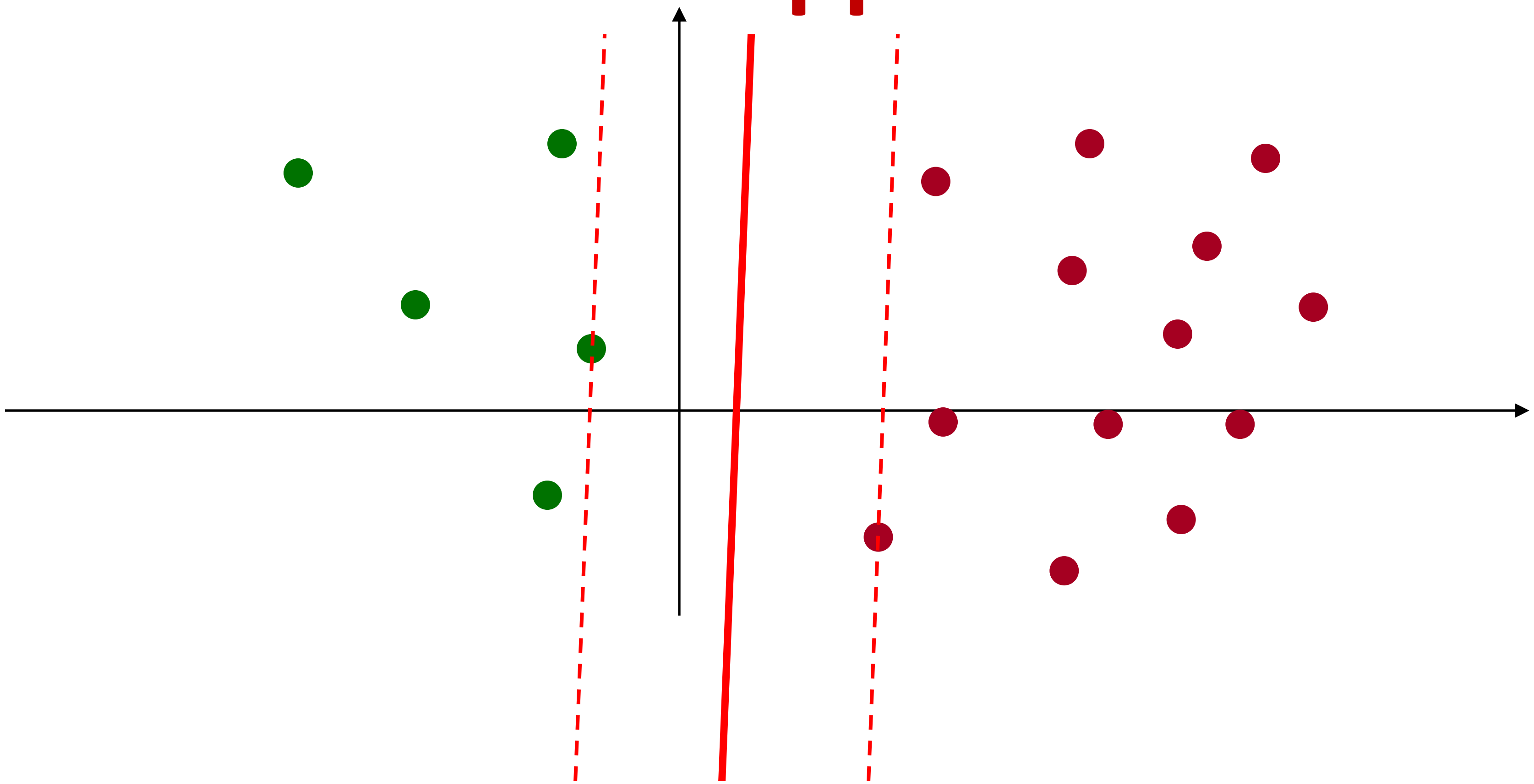
Solution Approach



Expanding the gap by changing the direction of W

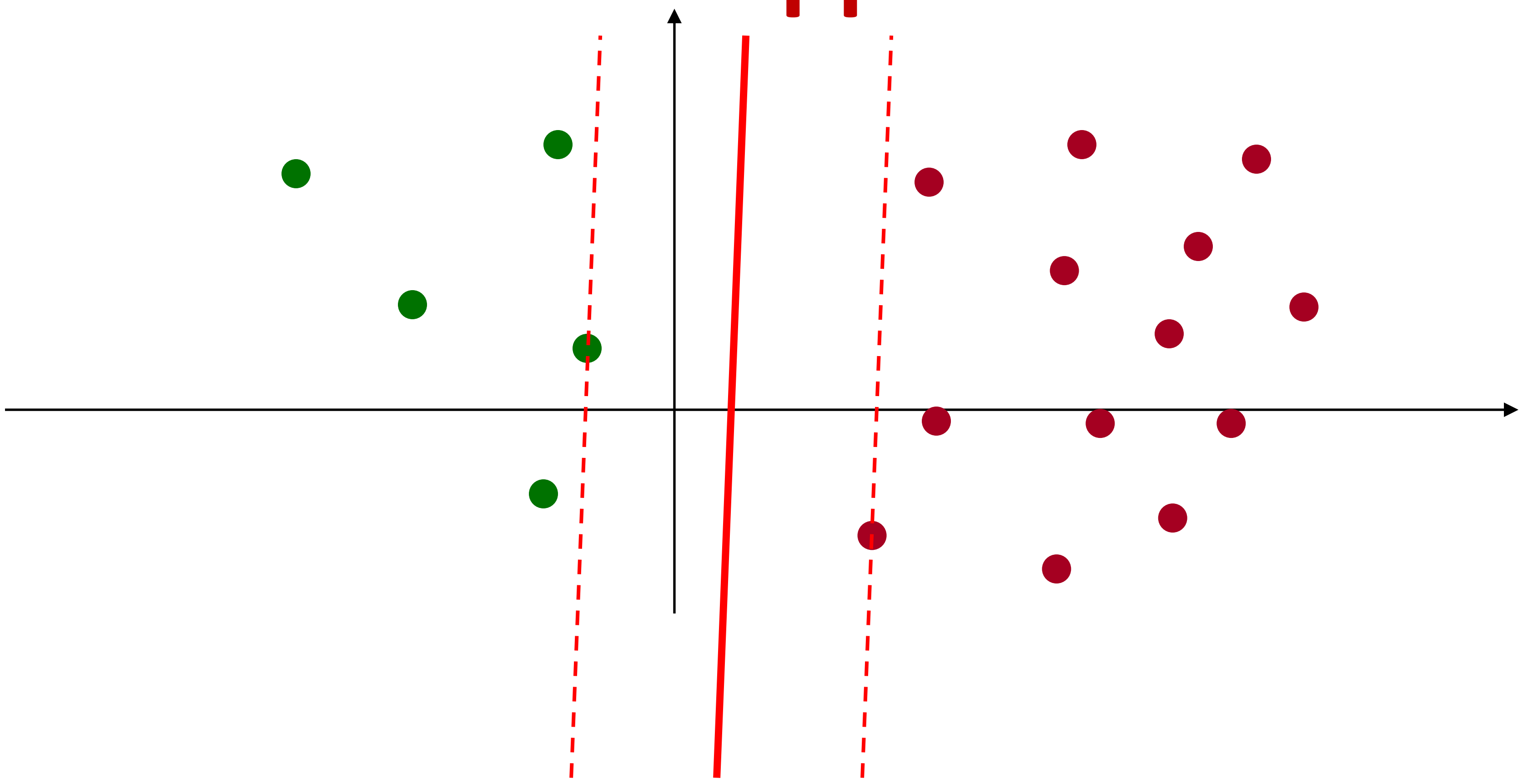
- The distance between the hyperplanes is $\frac{2}{\|W\|}$
- Find the W (and b) such that this is maximized, while maintaining the constraint that all training points are on the "outside" of the appropriate hyperplane

Solution Approach



- The distance between the hyperplanes is $\frac{2}{\|W\|}$
- Find the W (and b) such that this is maximized, while maintaining the constraint that all training points are on the "outside" of the appropriate hyperplane

Solution Approach



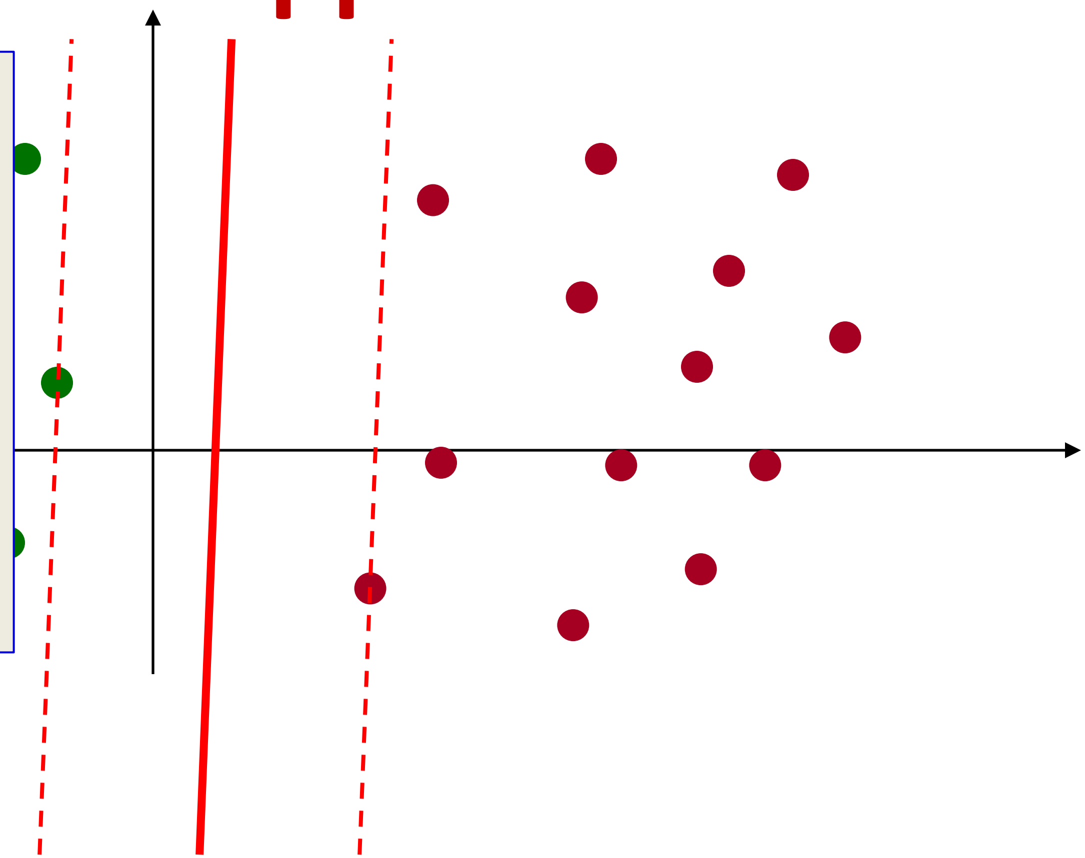
- The distance between the hyperplanes is $\frac{2}{\|W\|}$
- Maximize this distance. I.e. ..
- **Minimize $\|W\|$ such that**
  - all training points are on the “outside” of the appropriate hyperplane

Solution Approach

Decreasing the length of W will expand the gap between the boundary planes

Rotating it will *also* increase this length

Must find a formalism that explores both options simultaneously



- The distance between the hyperplanes is $\frac{2}{\|W\|}$
- Maximize this distance. I.e. ..
- **Minimize $\|W\|^2$ such that**
  - all training points are on the "outside" of the appropriate hyperplane

Lets formalize this

- Constraint: Ensuring that all training instances are on the proper side of their respective hyperplanes
- For positive training instances X_i:

$$W^T X_i - b \geq 1$$

- For negative instances

$$W^T X_i - b \leq -1$$

- Generically stated, for all instances we want

$$Y_i\left(W^T X_i - b\right) \geq 1$$

Solution Formalism

- Minimize $||W||$ such that
- For all training instances

$$Y_i\left(W^T X_i - b\right) \geq 1$$

- Formally

$$\widehat{W} = \underset{W,b}{\operatorname{argmin}} ||W||^2$$
$$s.t.\ \forall\, i \quad Y_i\left(W^T X_i - b\right) \geq 1$$

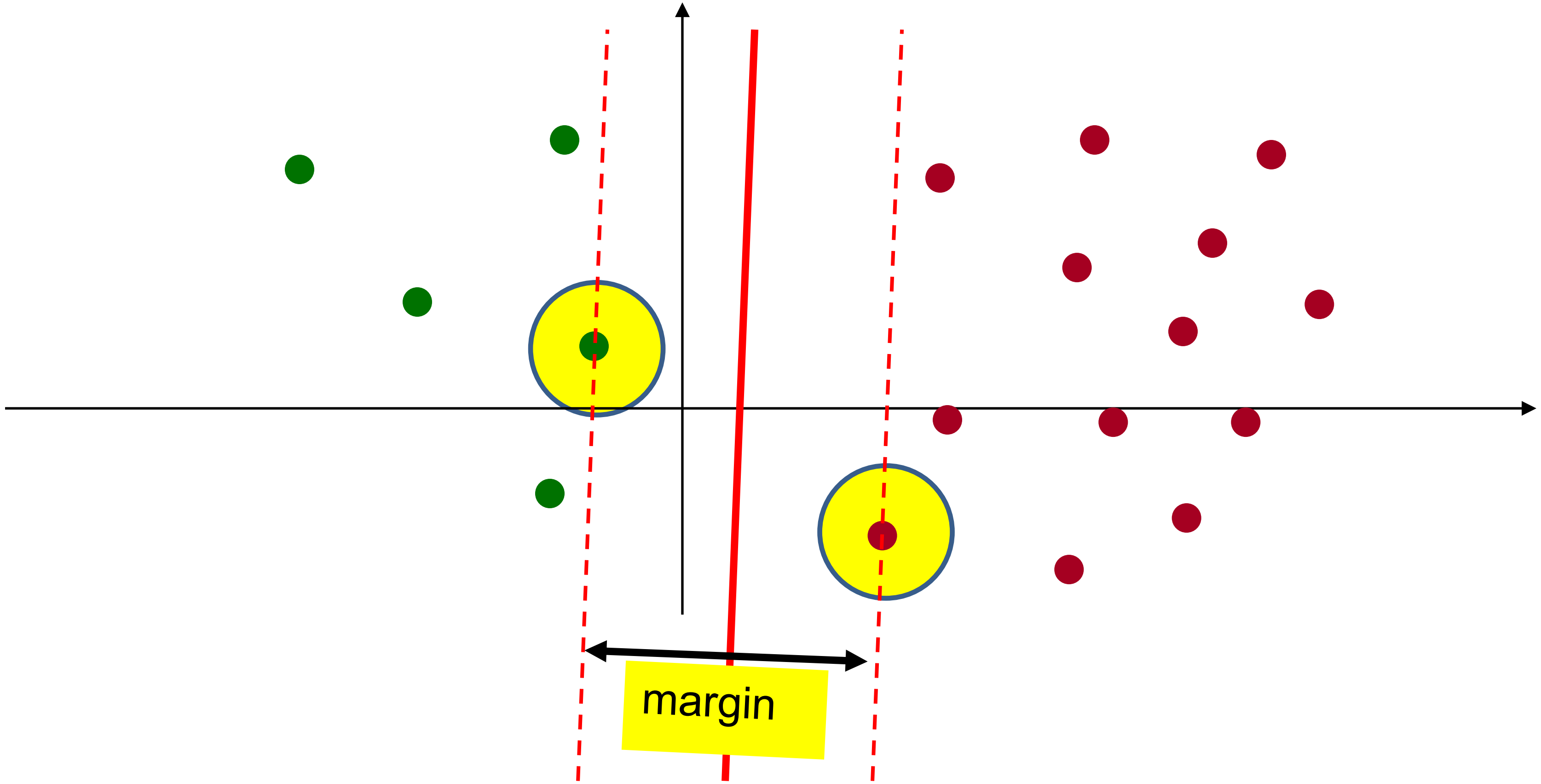
Solving the optimization

- This is a quadratic programming problem!

$$\widehat{W} = \underset{W,b}{\operatorname{argmin}} \|W\|^2$$
$$s.t. \; \forall i \quad Y_i\left(W^T X_i - b\right) \geq 1$$

- A variety of techniques can be applied
  - Interior point methods, active set methods, gradient descent, conjugate gradient
  - The objective function is convex, QP *will* find the (near) optimal solution
- Most useful solution is based on *Lagrangian duals*
  - Later..

The solution

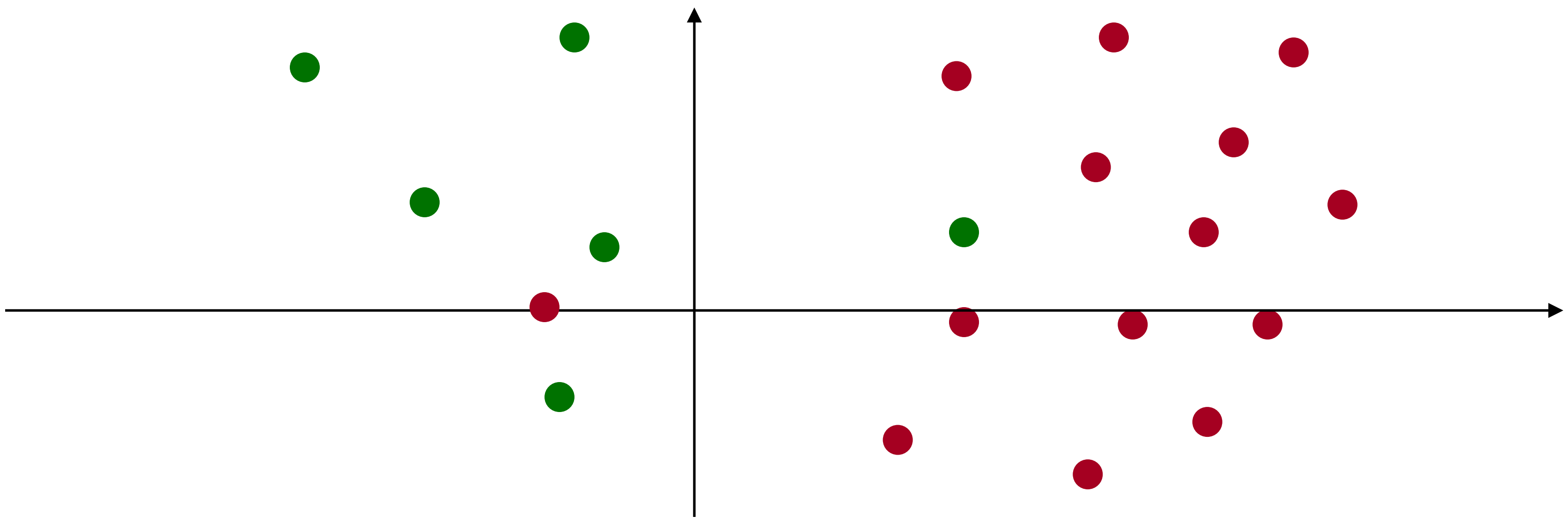


- Maximizes the *margin*
- This is a *max-margin* classifier
- The boundary samples are called support vectors
  - All the information about the classifier is in these support vectors

Challenges

- What if the classes are not linearly *separable*
- What if the classes are not *linearly* separable?
- What if the classes are not *linearly separable*?

What if they are not separable?



- What if the data are not separable?

Original Problem

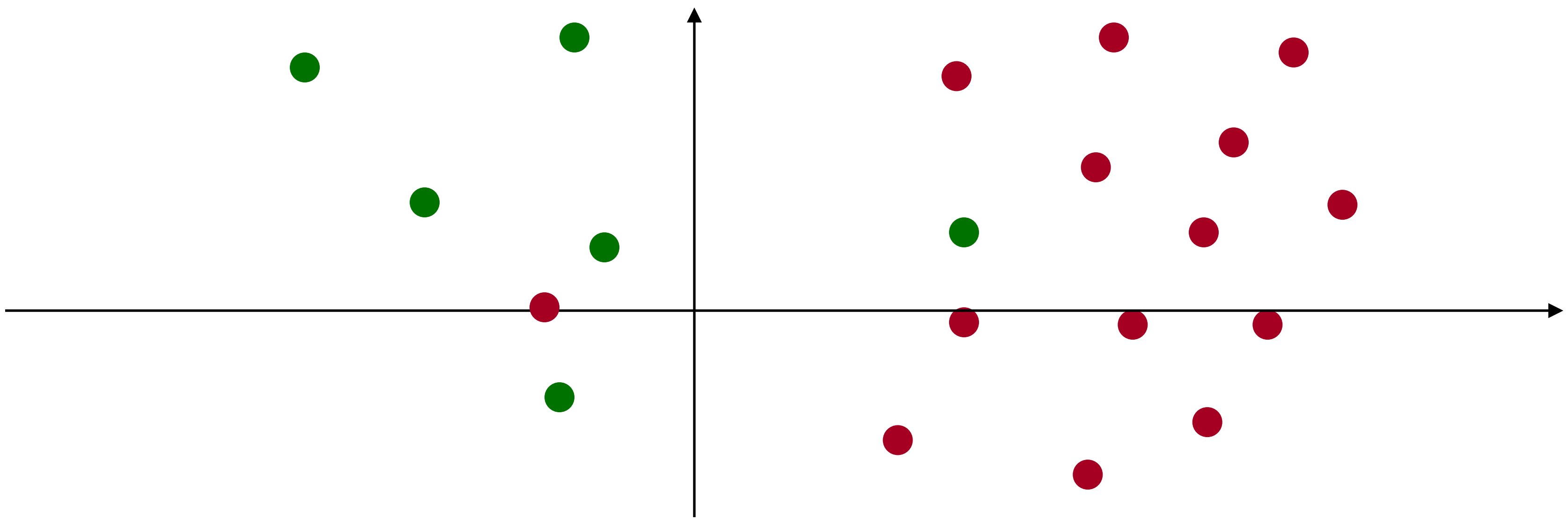
- This is a quadratic programming problem!

$$\widehat{W} = \underset{W}{\operatorname{argmin}} \|W\|^2$$

$$s.t.\ \forall i \quad Y_i\left(W^T X_i - b\right) \geq 1$$

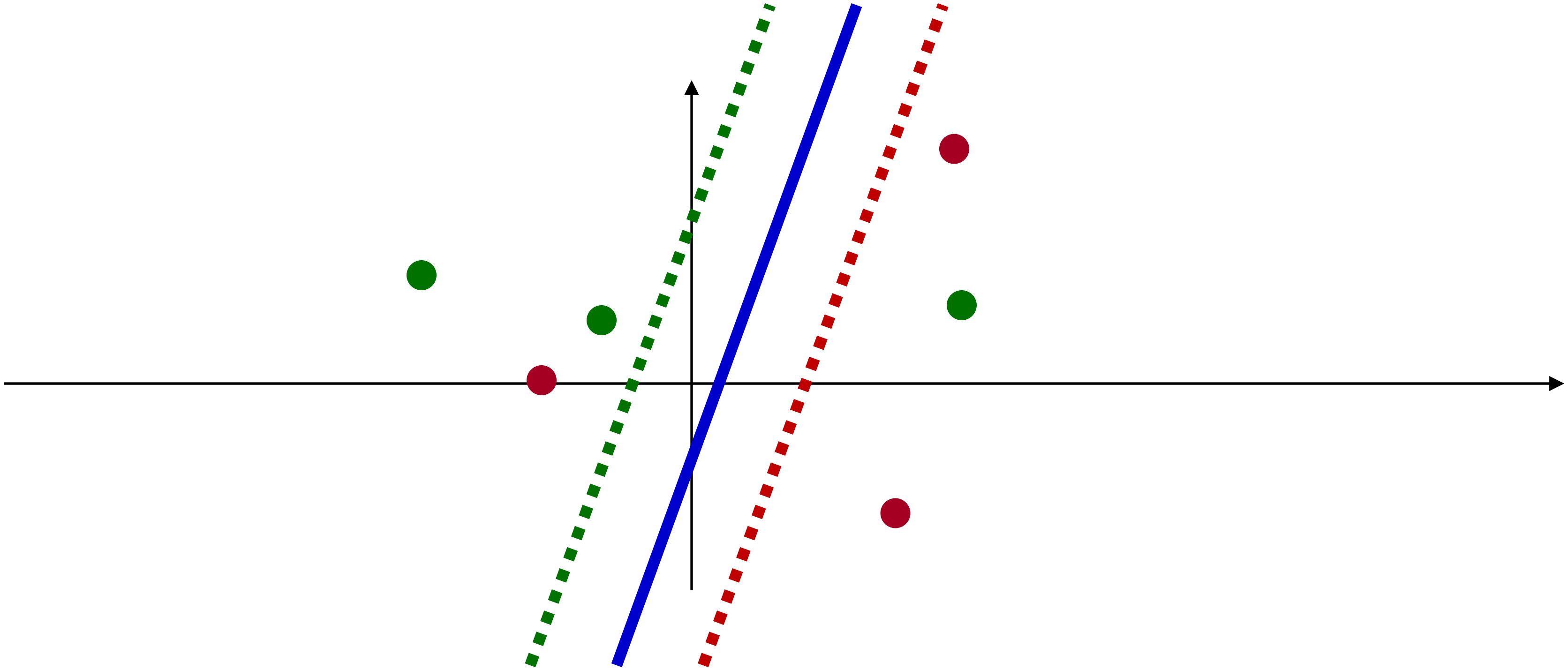
- Maximize the distance between the planes
- Subject to the constraint that all training data instances are on the “correct” side of the plane
- When data are not linearly separable, this constraint can never be satisfied

Introducing the *slack* variable



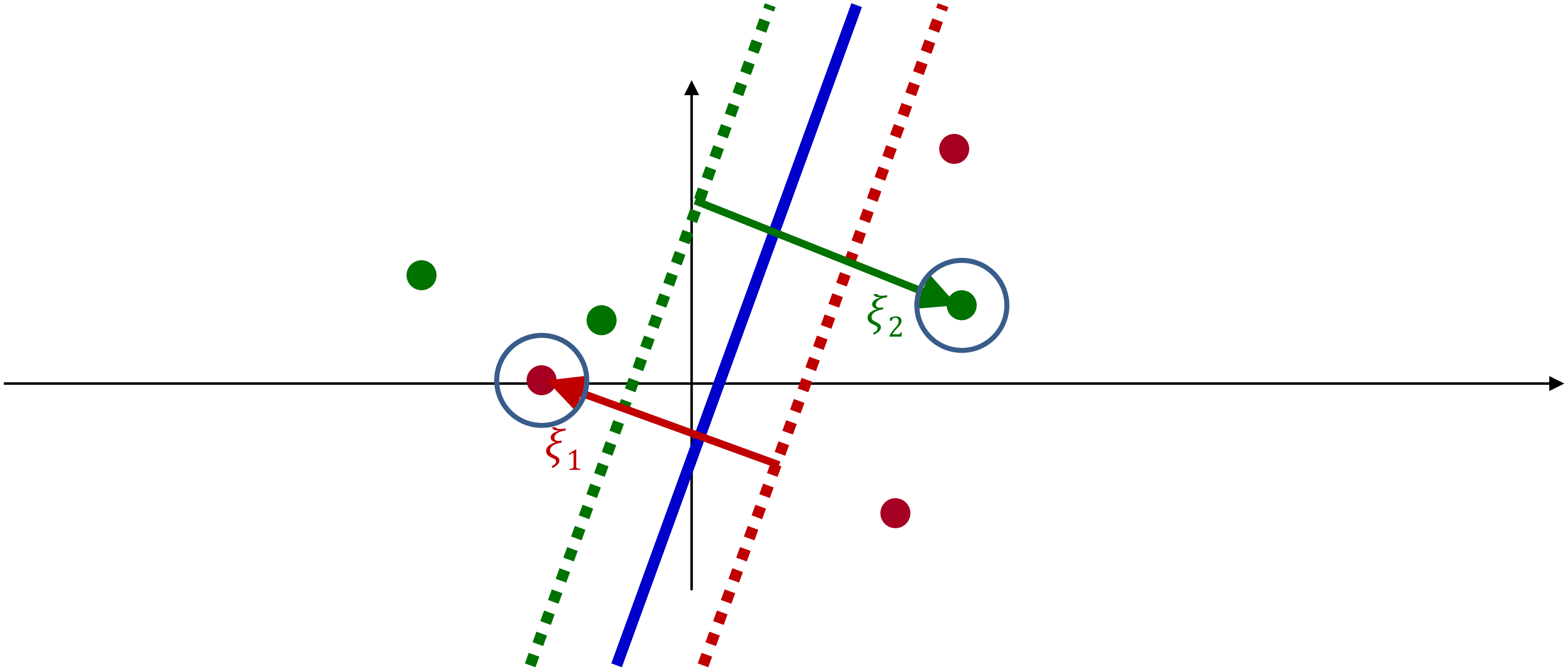
- What if the data are not separable?

Introducing the *slack* variable



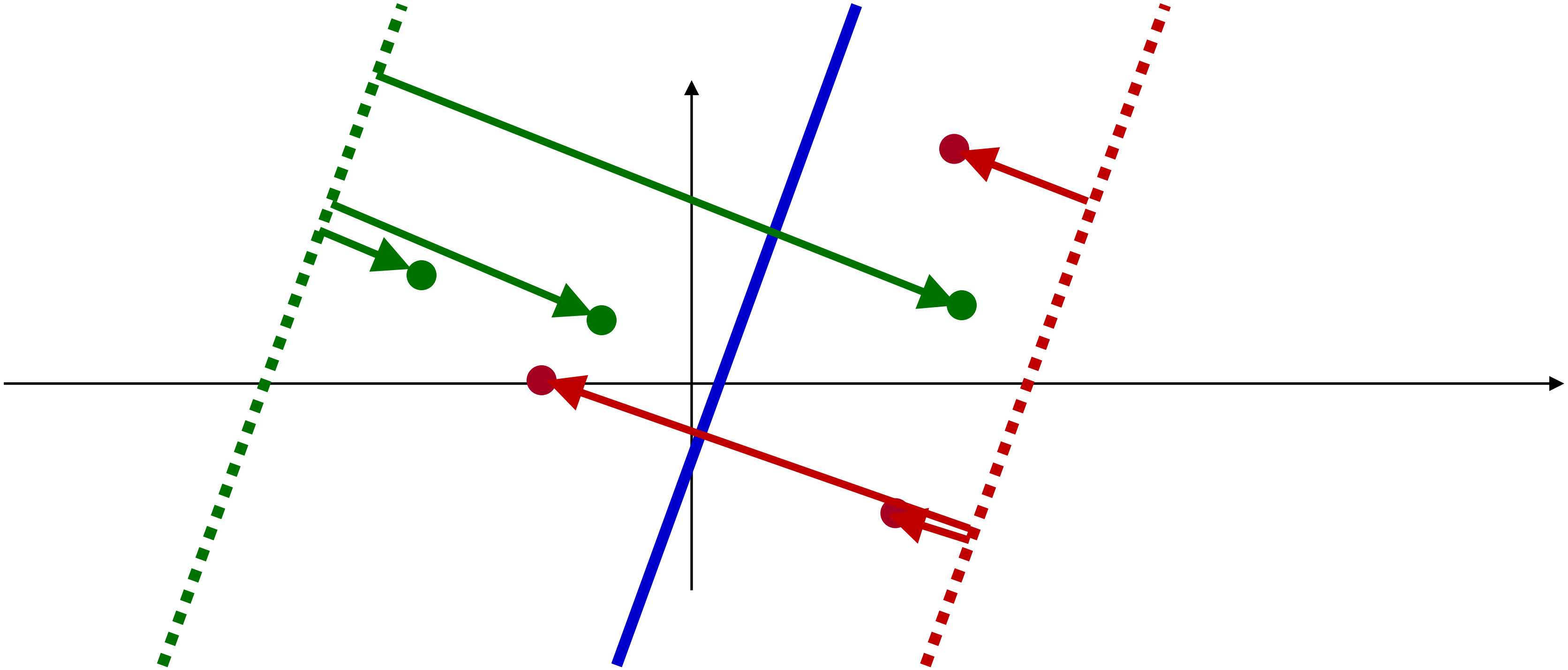
- For every training instance, introduce a *slack* variable ξ
- The slack variable is the maximum distance you have to *shift* the boundary plane to move the point to the "correct" side

Introducing the *slack* variable



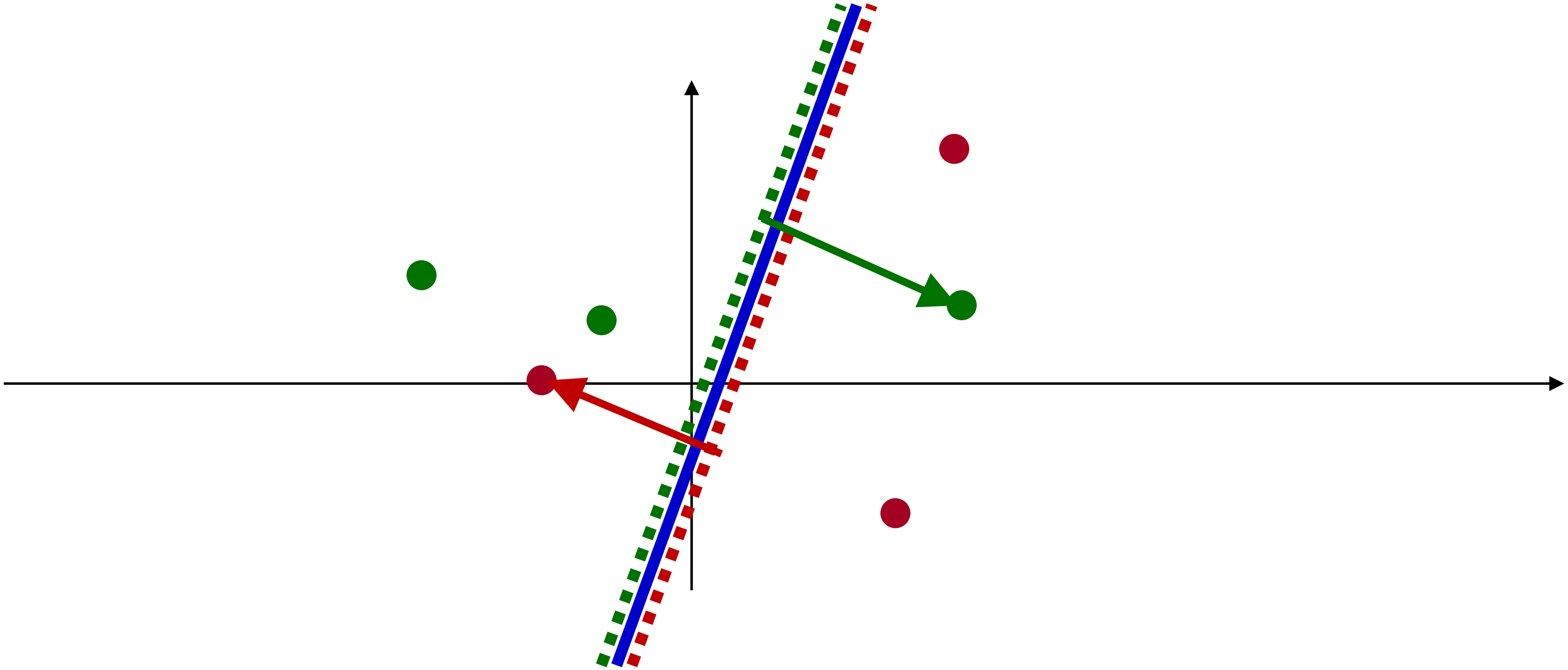
- For every training instance, introduce a *slack* variable ξ
- The slack variable is the *reverse* distance from the *margin* plane of the training instance
  - This will be non-zero only for some instances
  - **Ideally this should be minimum**

Introducing the *slack* variable



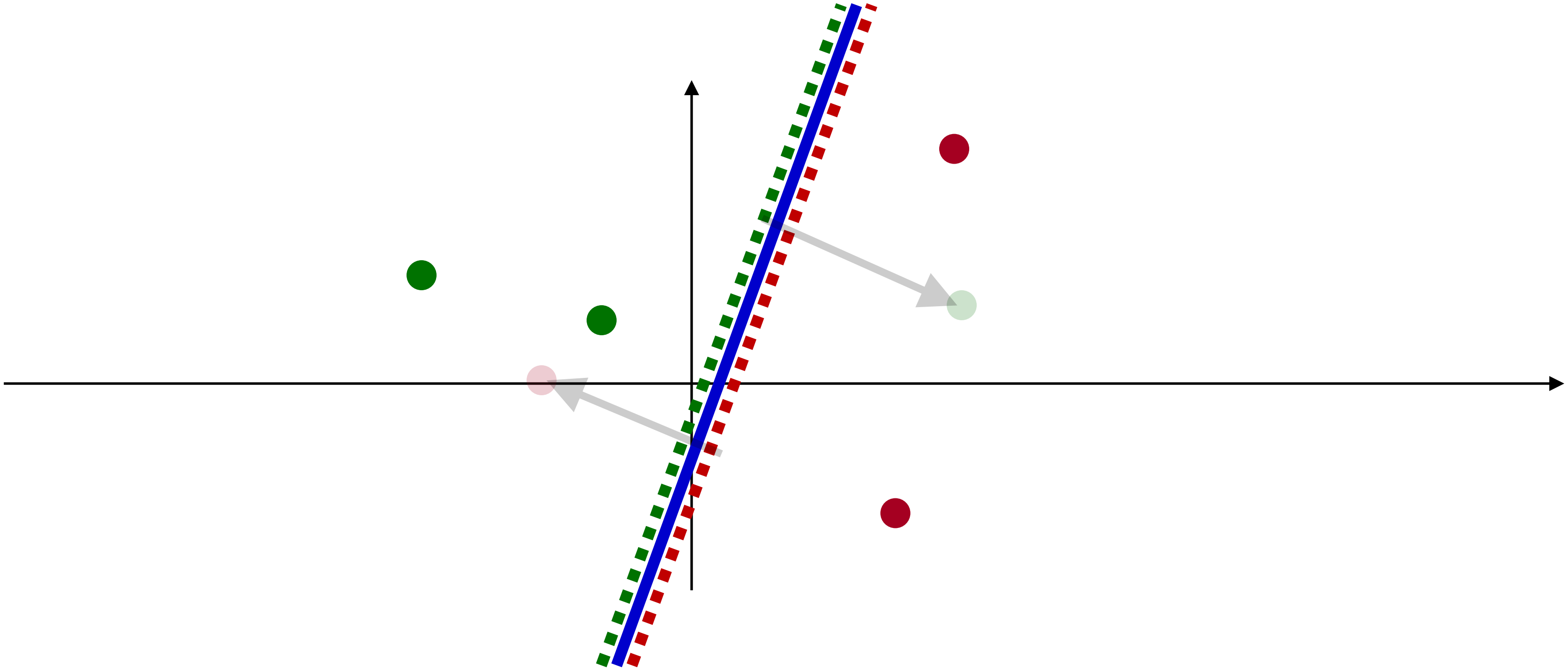
- The total length of slack variables varies with the boundary
- If you push the boundaries too far you will have a greater length of slack variable
  - Which contradicts our desire that they should be minimum

Introducing the *slack* variable



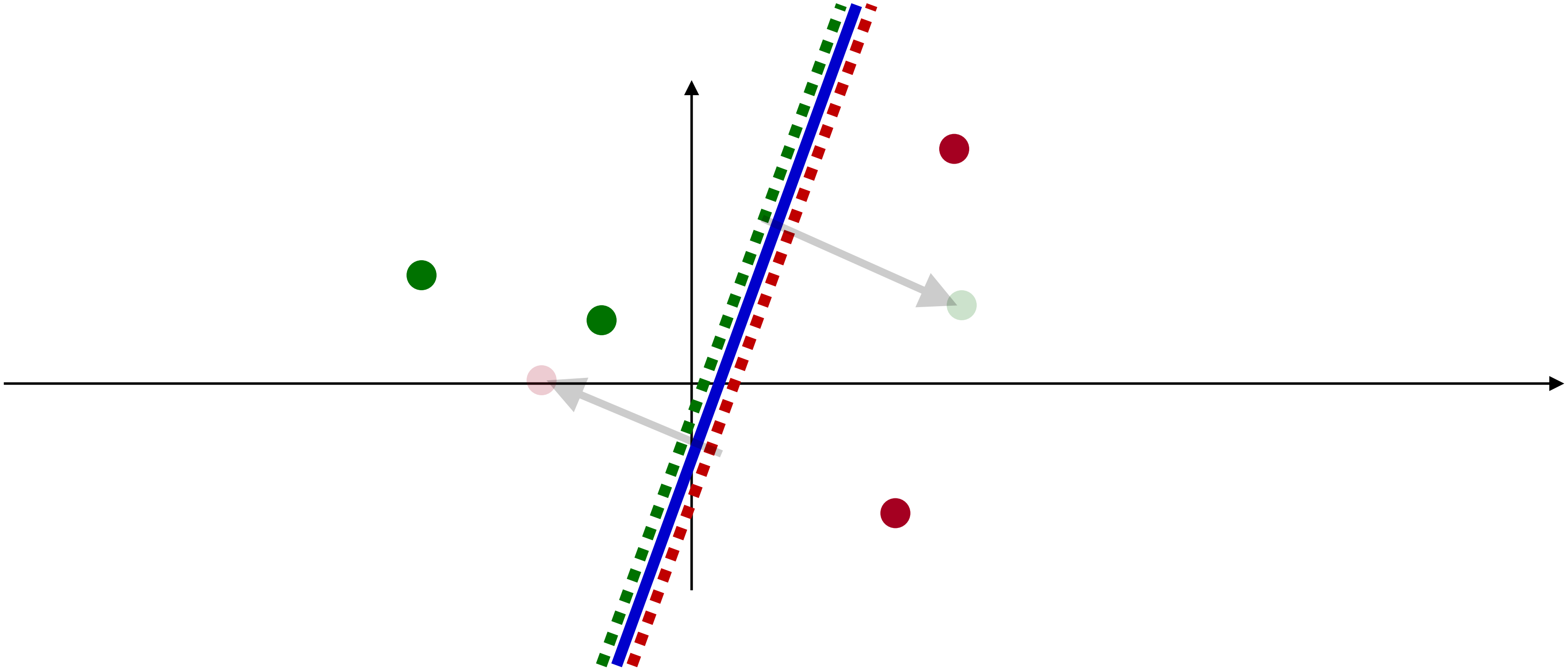
- If they are very close, only the *inseparable points* will have non-zero slack variable
  - The minimum slack value is when the margin planes coincide with the linear classifier

Introducing the *slack* variable



- If they are very close, only the *inseparable points* will have non-zero slack variable
  - The minimum slack value is when the margin planes coincide with the linear classifier
- For linearly separable classes, if the boundary planes are close enough, the total slack length will be 0

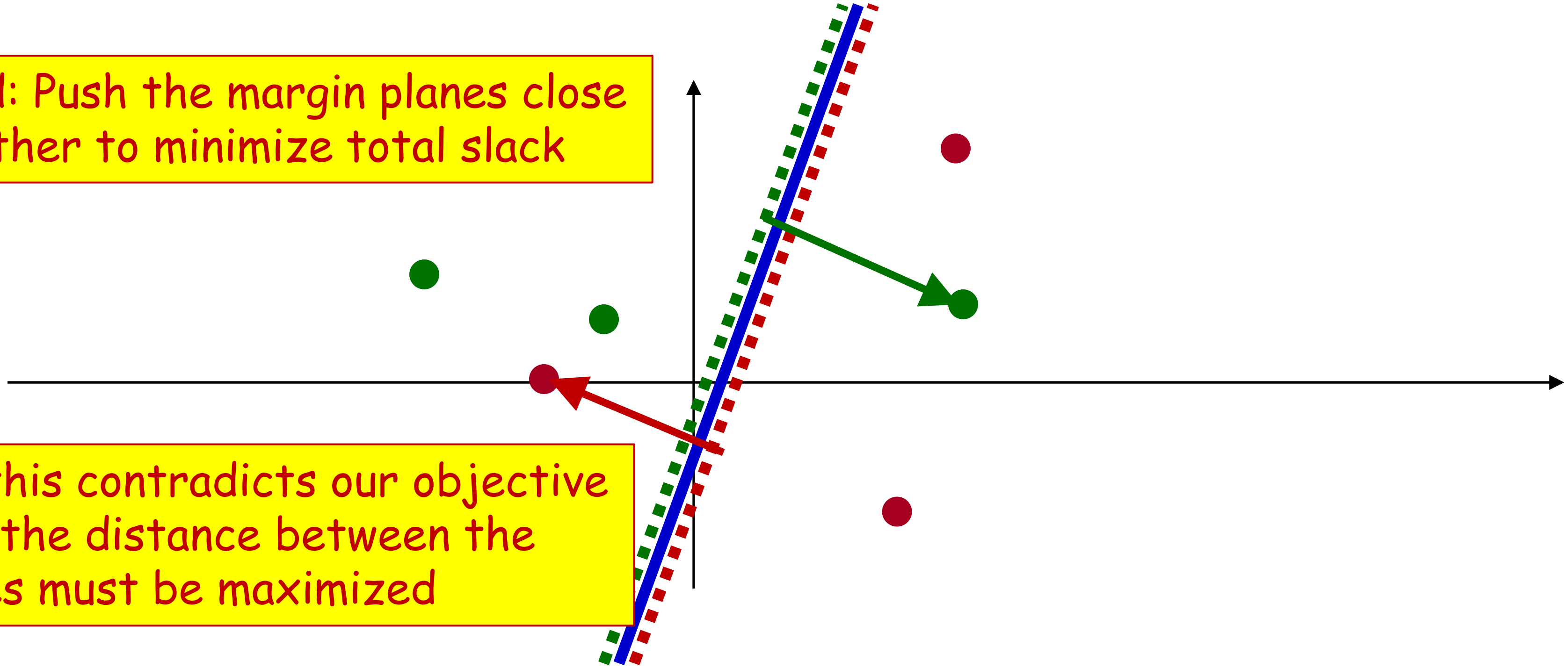
Introducing the *slack* variable



- Problem: If they are too close, the planes violate our desire to *maximize* the margin

Introducing the *slack* variable

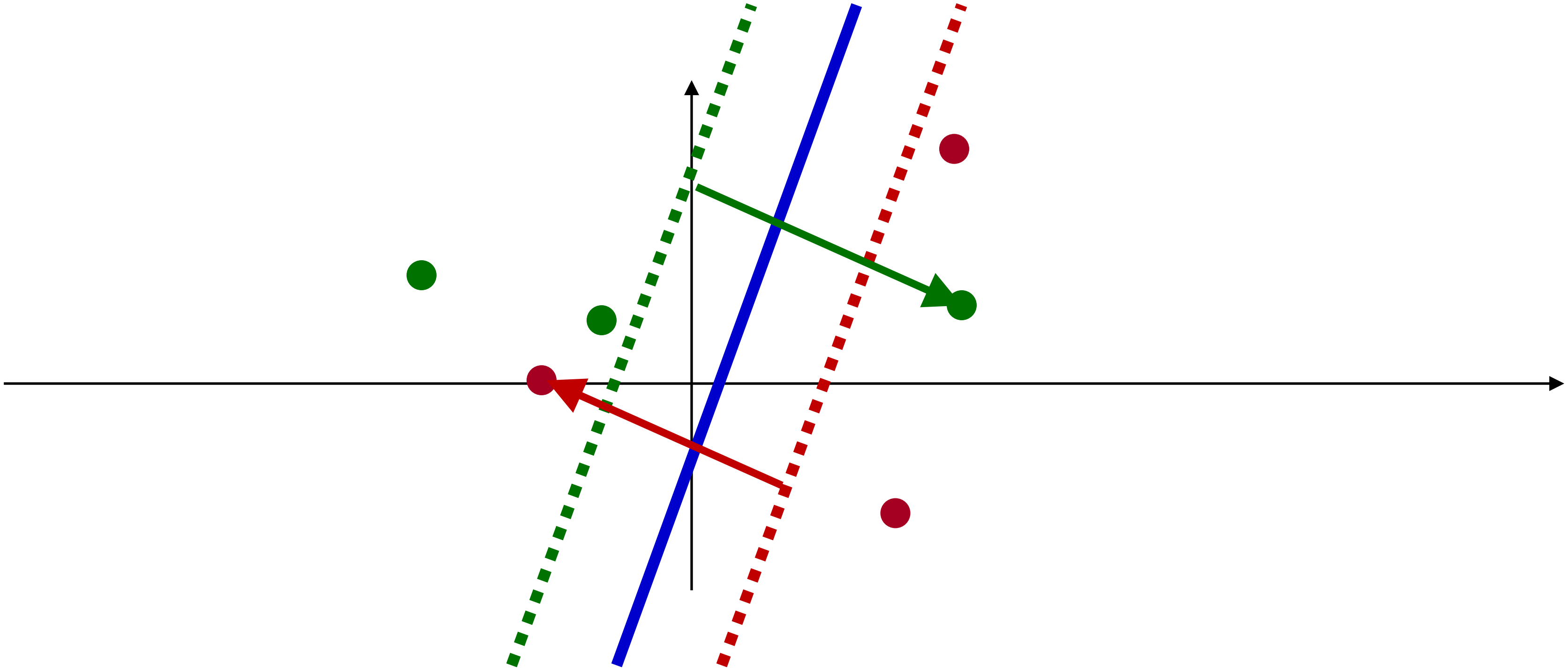
Need: Push the margin planes close together to minimize total slack



But this contradicts our objective that the distance between the planes must be maximized

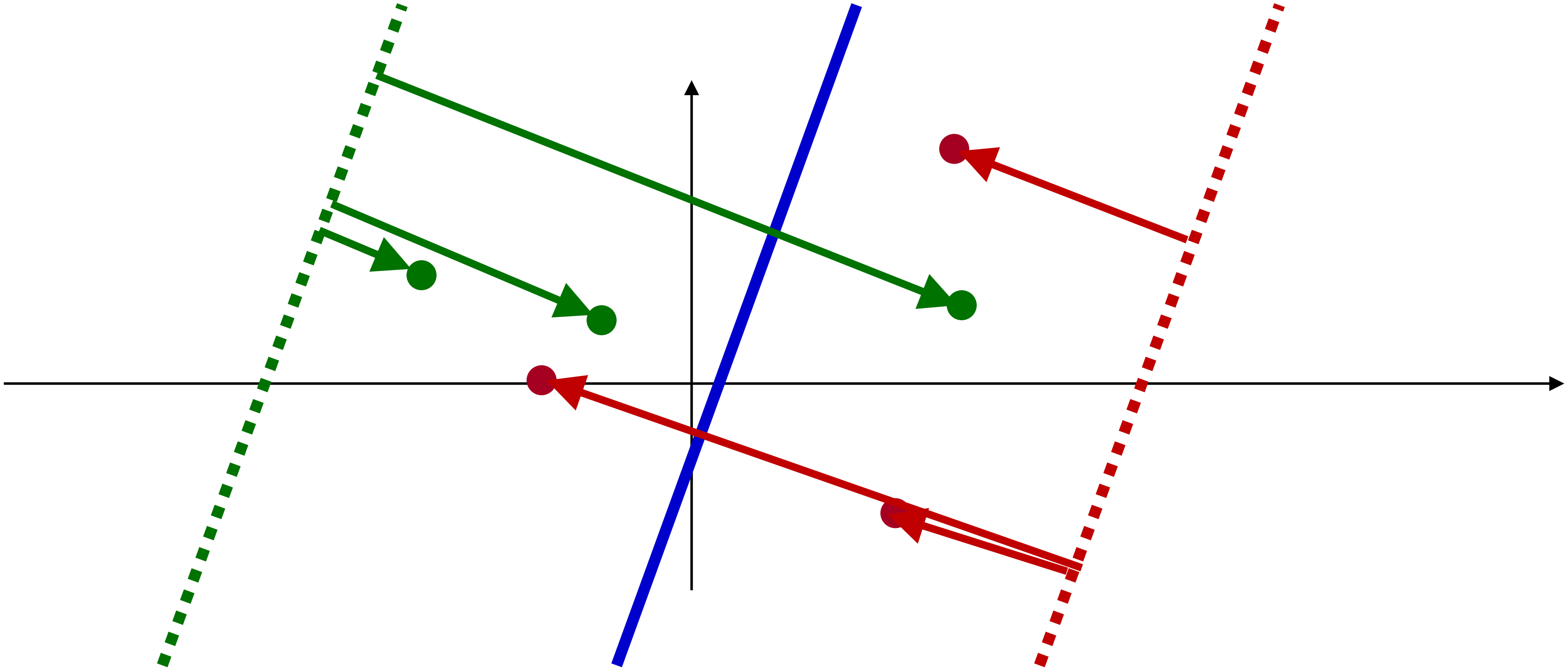
- Contradicting requirements..

New Objective



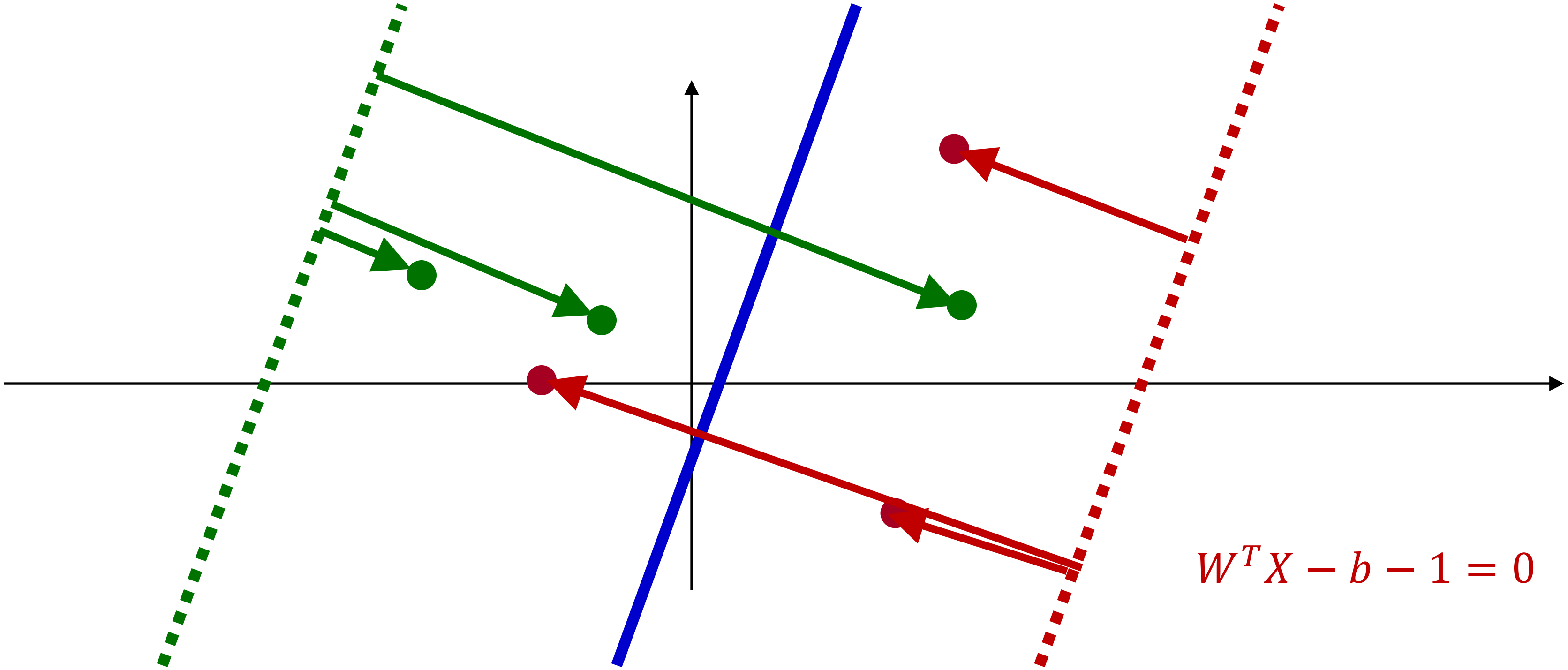
- Simultaneously
  - Maximize distance between planes
  - Minimize total slack length

Quantifying Slack Length



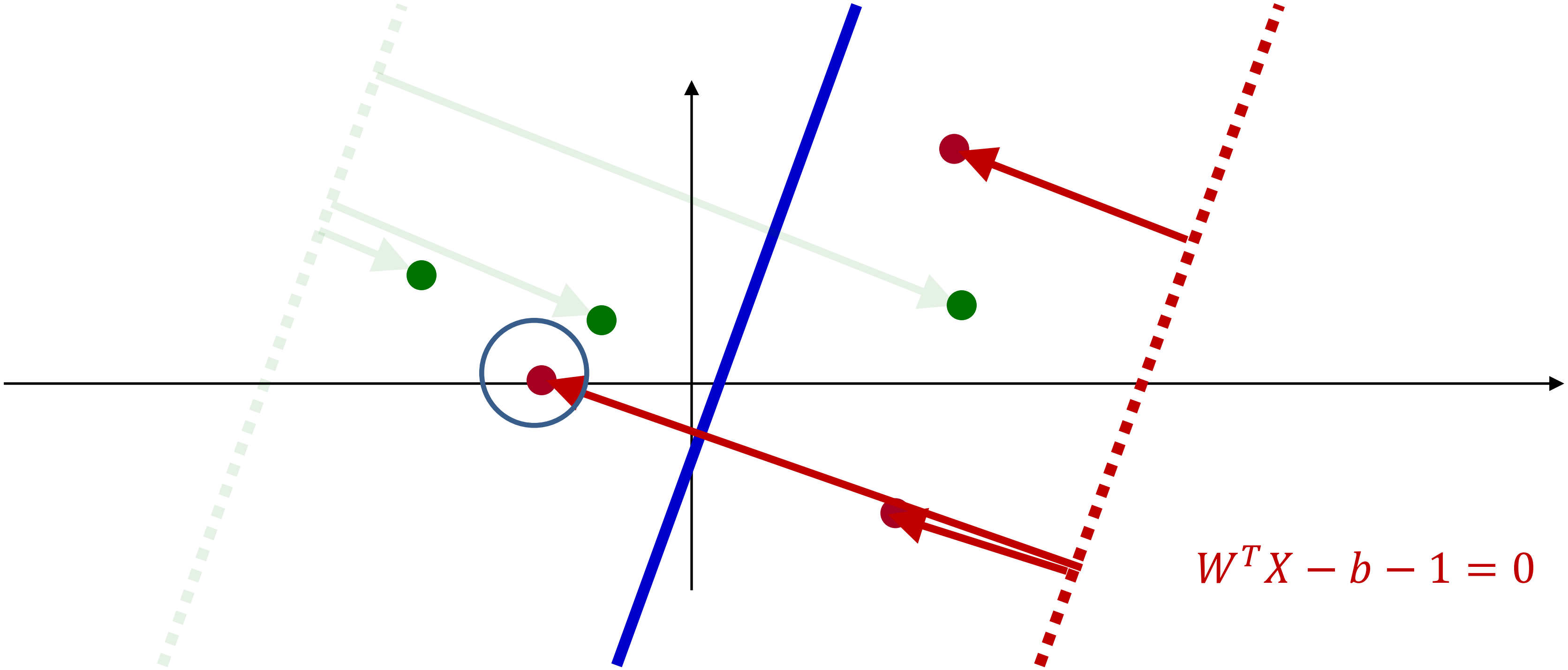
- We need a formula for the total slack length first..

Quantifying Slack Length



- The *positive* margin plane is given by
- $W^T X - b - 1 = 0$
- This plane is at a distance is $\frac{1}{\|W\|}$ from the decision boundary on the *positive* side of the decision plane (in the direction of W)
  - Ideally all positive training points would be to the right of it

Quantifying Slack Length

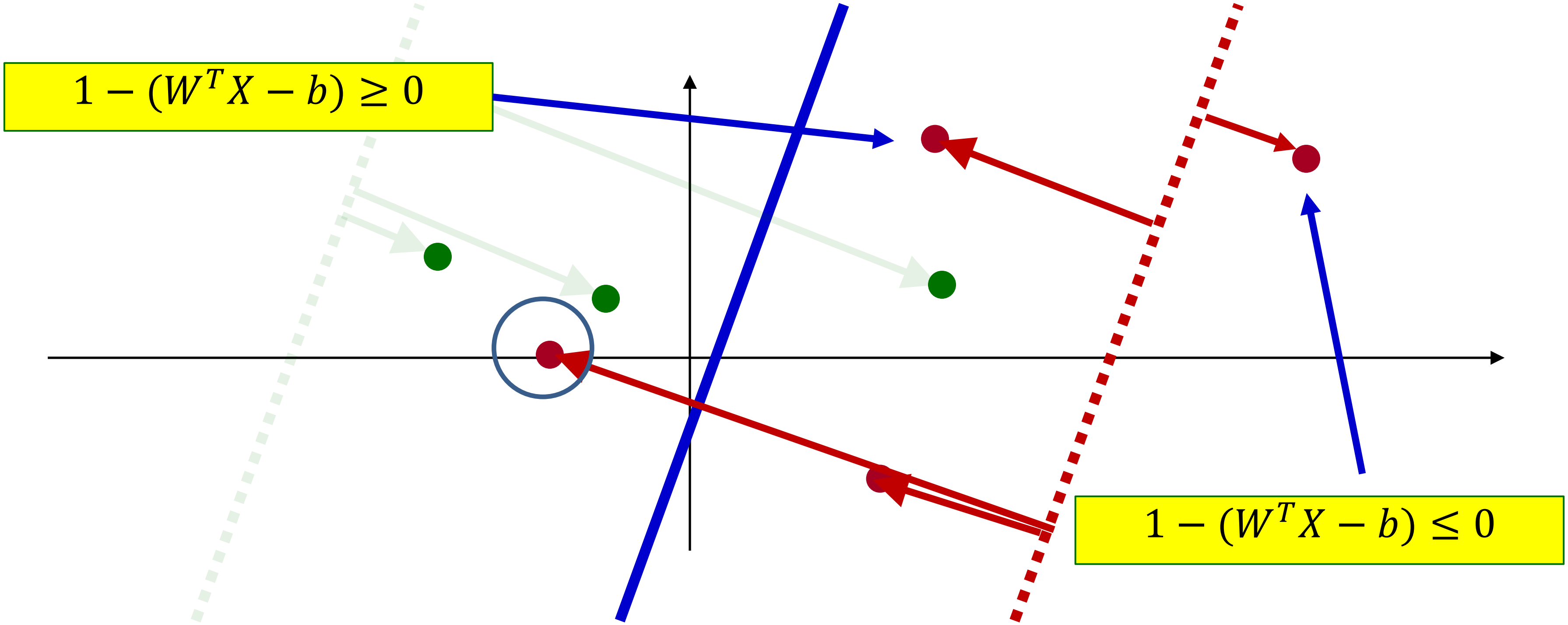


- The (unnormalized) distance of any X from this plane

$$W^TX - b - 1$$

- This will be negative for instances on the "wrong" side (in the direction away from W), but positive for those on the "right" side

Quantifying Slack Length

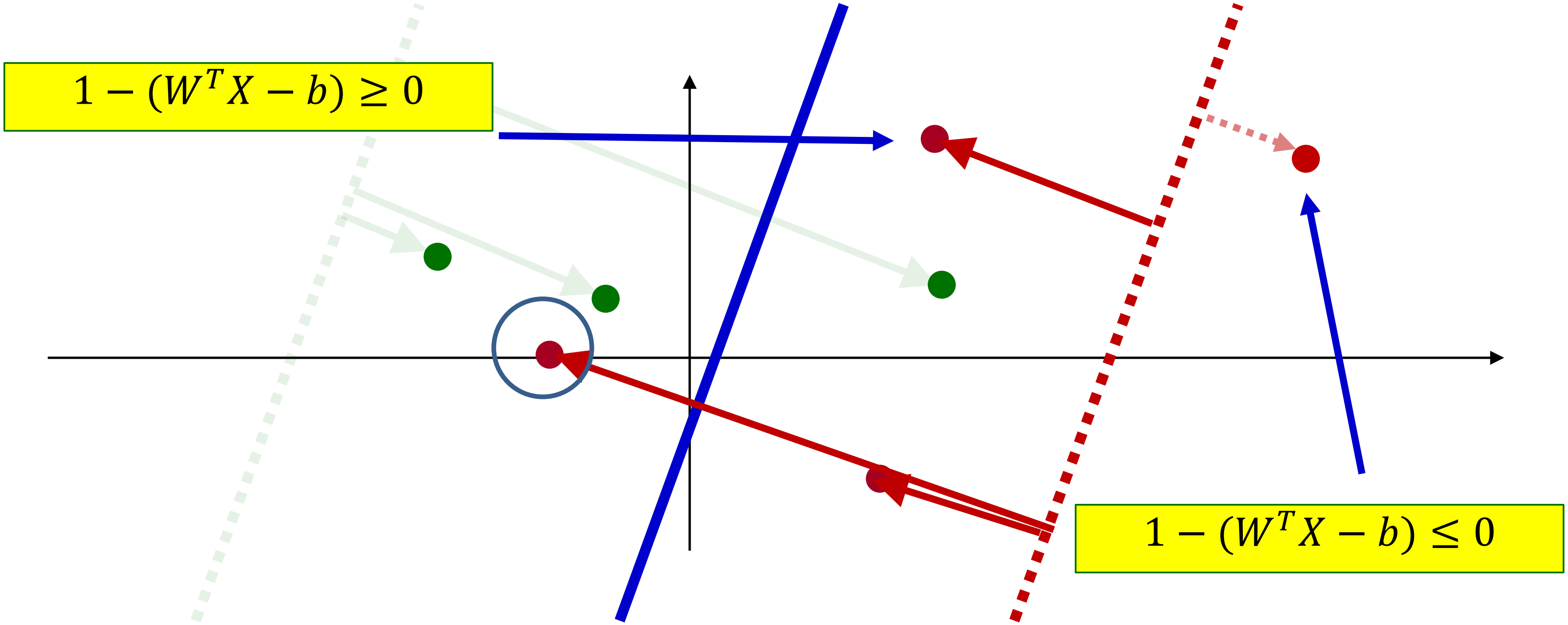


- The *negated* (unnormalized) distance of any X from this plane

$$1 - (W^T X - b)$$

- This will be positive for instances on the wrong side of the margin plane, but negative for instances on the right side of it

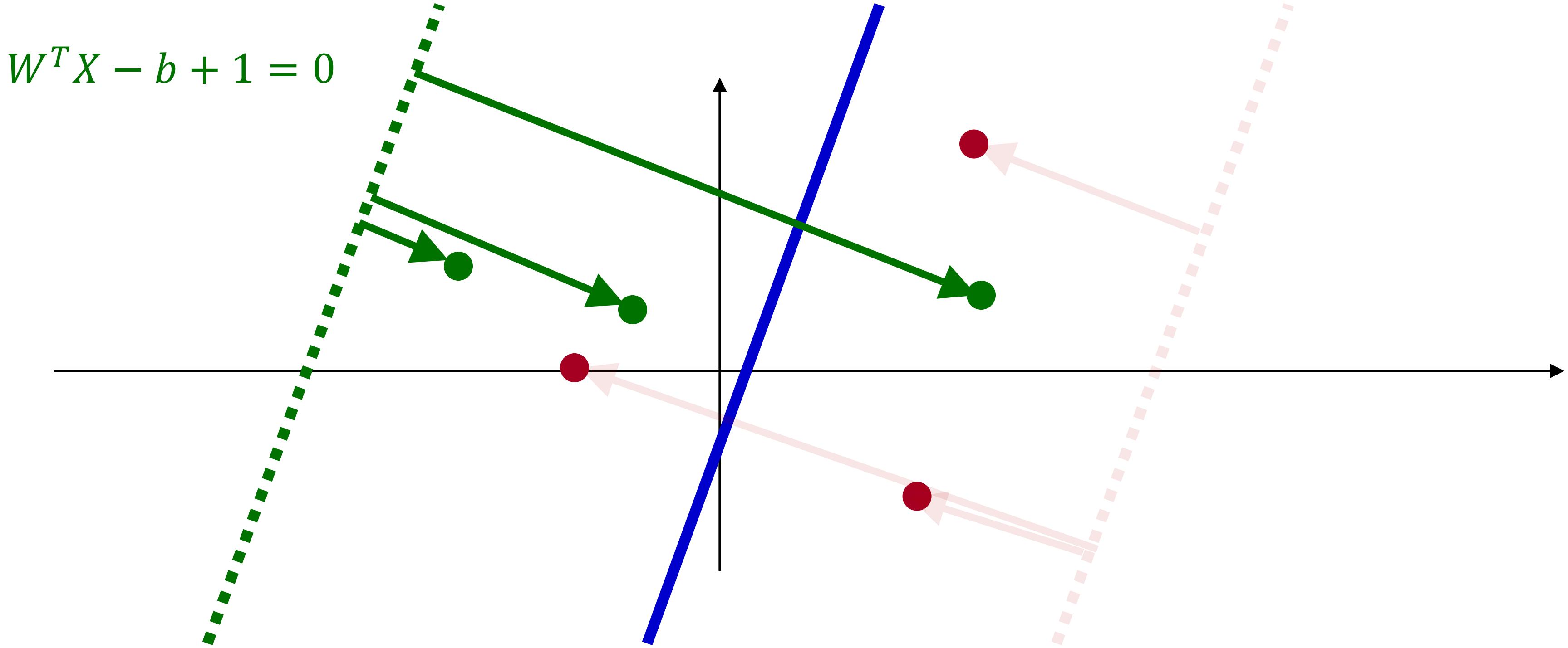
Quantifying Slack Length



- We do not care about the actual distance of instances to the *right* of the plane
- So the slack value of any point is

$$\max\left(0, 1 - (W^T X - b)\right)$$

Quantifying Slack Length

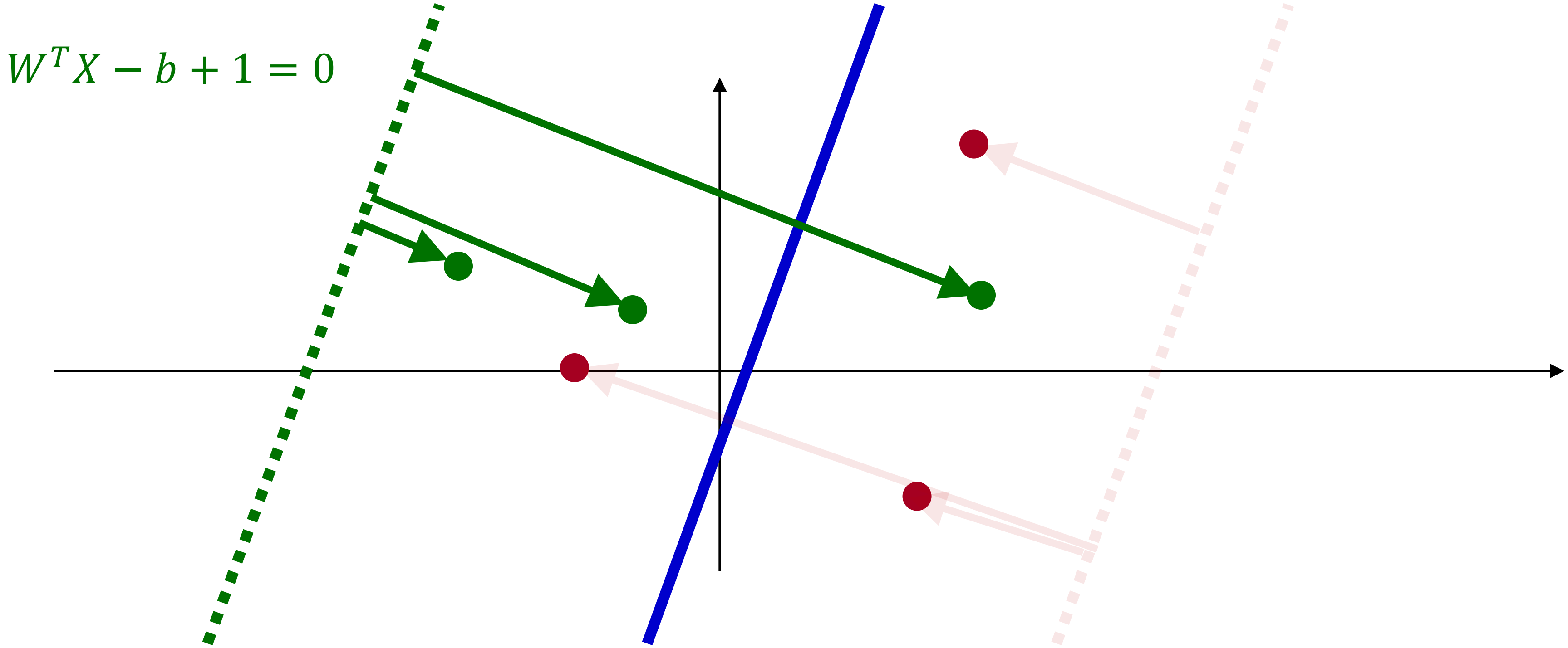


- The *negative* margin plane is given by

$$W^T X - b + 1 = 0$$

  – Ideally all negative training points would be to the left of it

Quantifying Slack Length

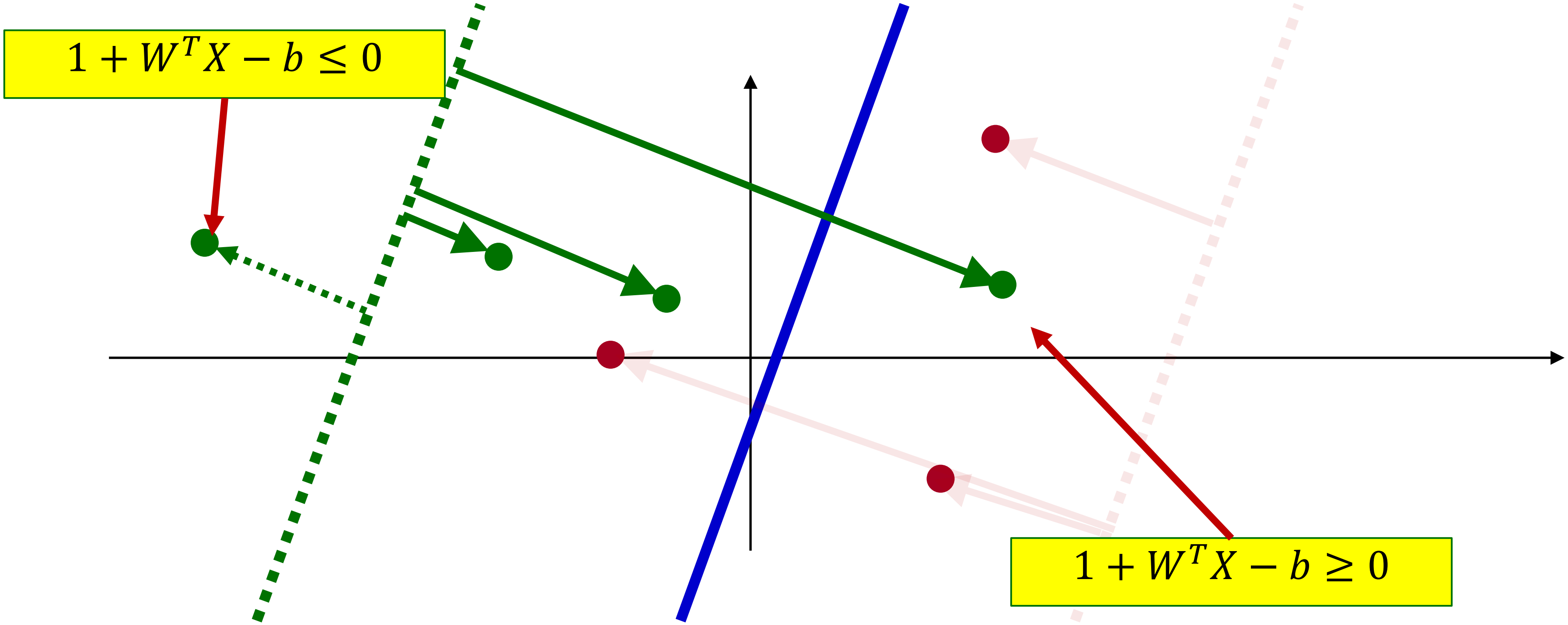


- The (unnormalized) distance of any X from this plane

$$W^T X - b + 1 = 1 + W^T X - b$$

- This will be positive for vectors on the “wrong” side, but negative for vectors on the right side

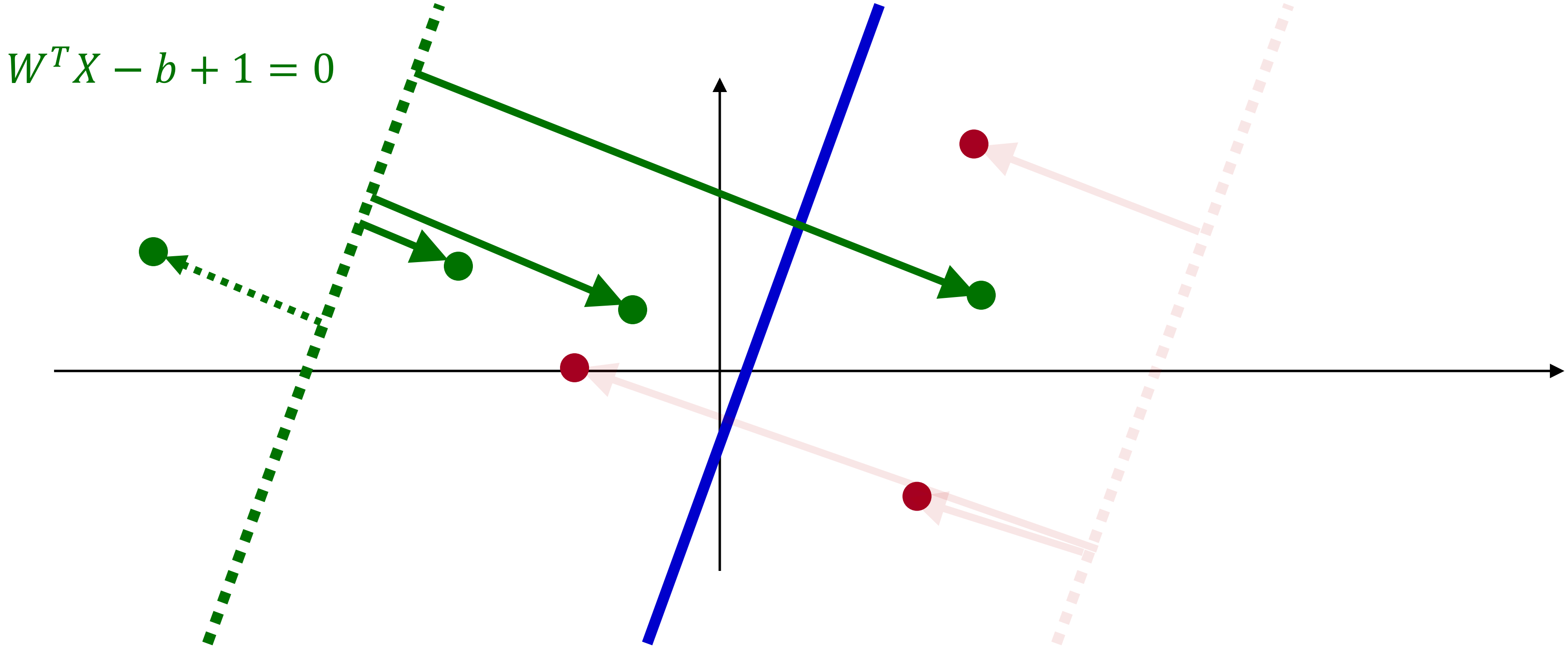
Quantifying Slack Length



- We do not care about the actual distance of instances to the *left* of the plane
- So the slack value of any point is

$$\max\left(0, 1 + W^T X - b\right)$$

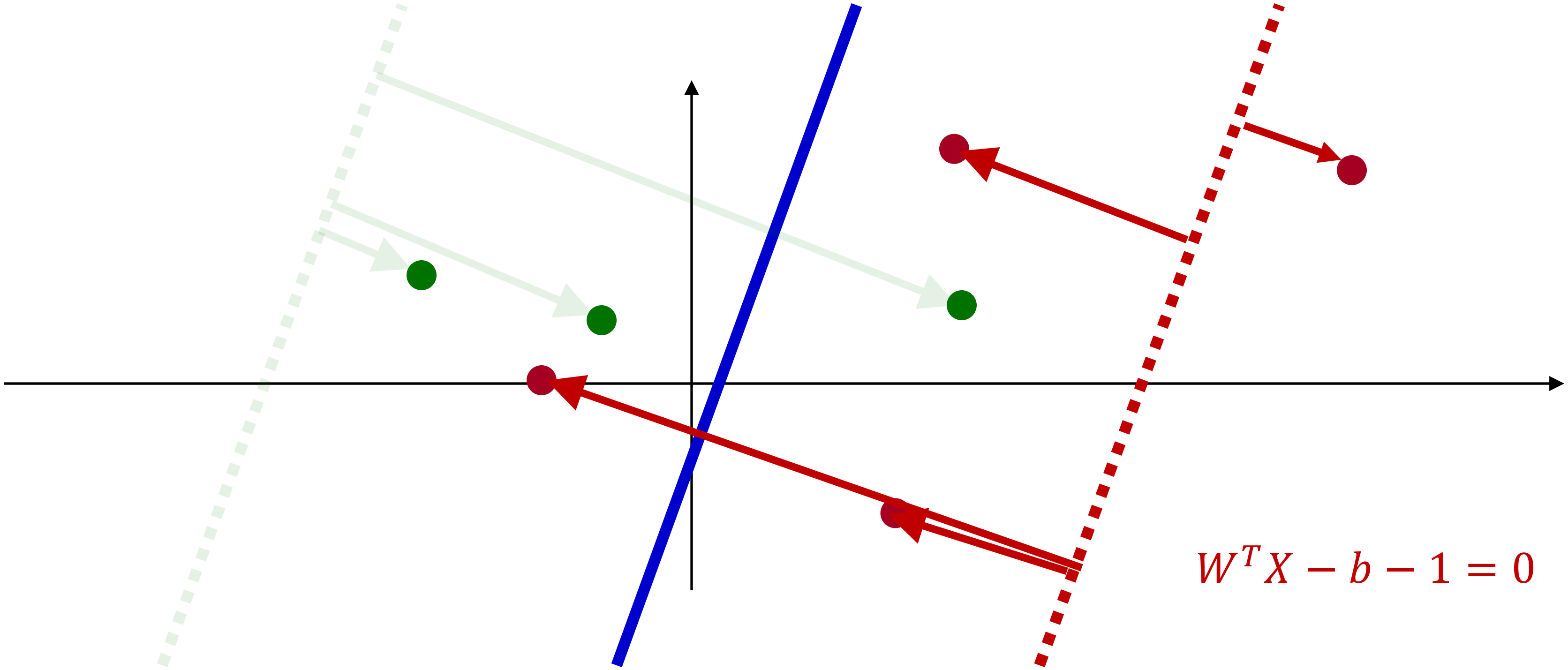
Quantifying Slack Length



- Combining the following for negative instances

$$\max\left(0, 1 + (W^T X - b)\right)$$

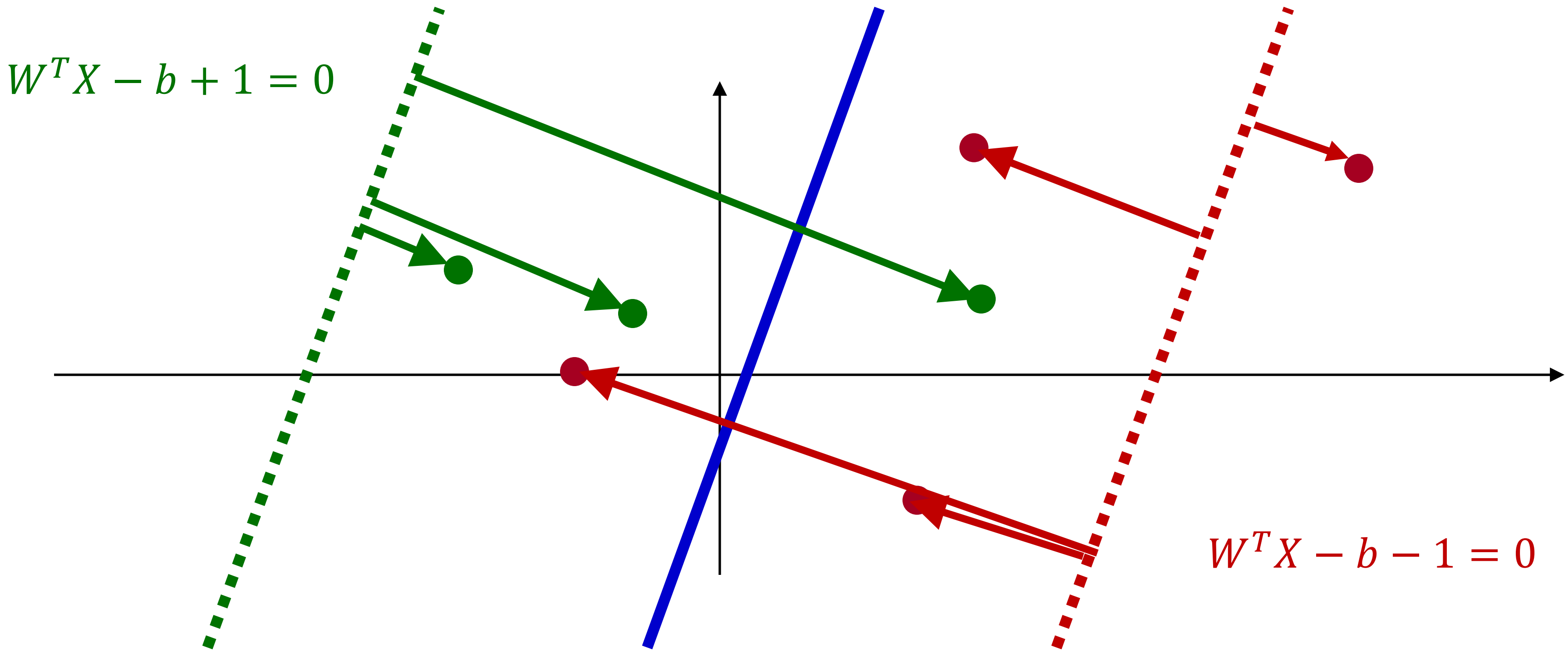
Quantifying Slack Length



- And the following for positive instances

$$\max\big(0, 1 - (W^T X - b)\big)$$

Quantifying Slack Length



- Generic Slack length for any point

$$\max\left(0, 1 - y(W^T X - b)\right)$$

- This is also called a *hinge loss*

Total Slack Length

- Total slack length for *all* training instances

$$\sum_i \max\big(0, 1 - y(W^T X - b)\big)$$

- This must be minimized

Overall Optimization

- Minimize $||W||^2$to maximize the distance between margin planes
- Minimize total slack length to minimize the distance of *misclassified* instances to margin planes

$$\sum_i \max\left(0, 1 - y(W^T X - b)\right)$$

  – This will make the margin planes *closer*
- The two objectives must be traded off..

Support Vector Machine for Inseparable data

- Minimize

$$\underset{W,b}{\operatorname{argmin}} \frac{1}{N}\sum_{i} \max\left(0, 1 - y(W^T X - b)\right) \ + \lambda \|W\|^2$$

- λ is a “regularization” parameter that decides the relative importance of the two terms

- This is just a regular optimization problem that can be solved through gradient descent

Support Vector Machine for Inseparable data

- λ is typically set using *held-out* training data
  - Train the classifier for various values of λ
  - Test each of these classifiers on some held-out portion of the training data that was not included in training the SVM
  - Pick the λ for which the classifier gave best performance
  - Retrain the SVM using the entire training data and this λ

- Frequently, instead of a single held-out set, λ is set through K-fold cross validation

Equivalent Slack Formalism

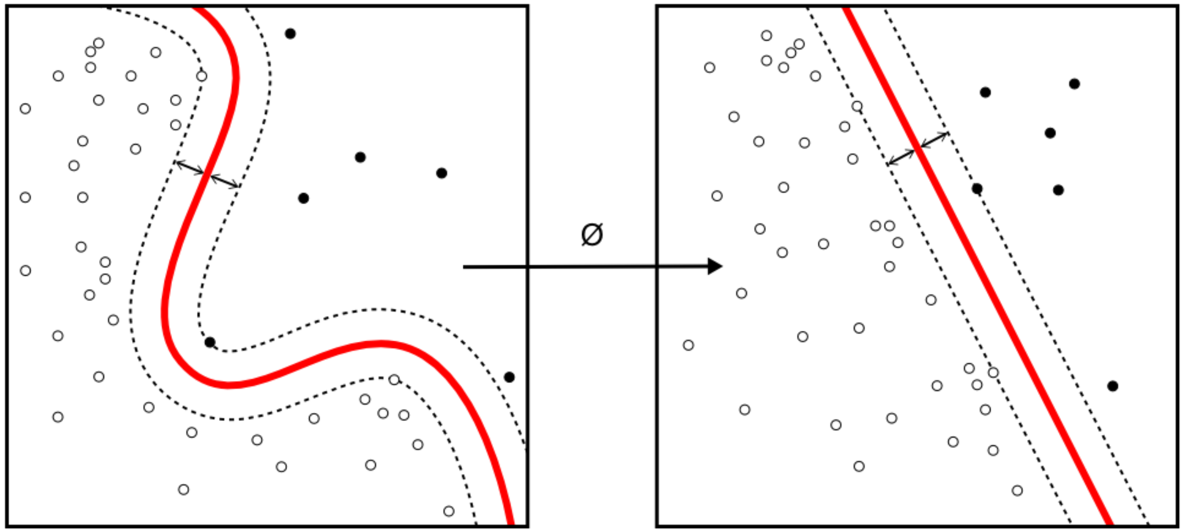
$$\underset{W,b}{\operatorname{argmin}} \; \|W\|^2 + C \sum_i \xi_i$$

- Subject to

$$Y_i\left(W^T X_i - b\right) \geq 1 - \xi_i$$

- This is a quadratic programming problem
- Slack parameter C is determined through held-out data as earlier (or through K-fold cross-validation)

How to deal with *non-linear* boundaries?



- First some math..

Recall: The Lagrange Method

- Optimize $f(x,y)$ subject to $g(x,y) = c$

$$\boxed{L(x,y,\lambda) = f(x,y) - \lambda(g(x,y) - c)}$$

to maximize $f(x,y)$: $\max_{x,y}\left(\min_{\lambda} L(x,y,\lambda)\right)$

to minimize $f(x,y)$: $\min_{x,y}\left(\max_{\lambda} L(x,y,\lambda)\right)$

Optimization with inequality constraints

- Optimization problem with constraints

$$
\begin{aligned}
&\min_x f(x) \\
&s.t.\ \ g_i(x) \le 0, \ \ i = \{1, \ldots, k\} \\
&\qquad h_j(x) = 0, \ \ j = \{1, \ldots, l\}
\end{aligned}
$$

- Lagrange multipliers $/_i$ ³ $0, n$ Î Â

$$L(x, /, n) = f(x) + \text{å}_{i=1}^{k} /_i g_i(x) + \text{å}_{j=1}^{l} n_j h_j(x)$$

- The optimization problem

$$\operatorname*{argmin}_x \max_{\lambda, v} L(x, \lambda, v)$$

Revisiting the *linearly separable* case

- This is a quadratic programming problem!

$$\widehat{W} = \underset{W}{\operatorname{argmin}} \|W\|^2$$
$$s.t. \;\forall i \quad Y_i\left(W^T X_i - b\right) \geq 1$$

- Can be stated using Lagrangians as

$$\underset{W,b}{\operatorname{argmin}} \; \underset{\alpha>0}{\max} \|W\|^2 + \sum_i \alpha_i \left(Y_i\left(W^T X_i - b\right) - 1\right)$$

Linearly separable case: Lagrangian formalism

- Can be stated using Lagrangians as

$$\underset{W,b}{\operatorname{argmin}} \ \max_{\alpha>0} ||W||^2 + \sum_i \alpha_i \left(Y_i \left(W^T X_i - b\right) - 1\right)$$

- The optimum satisfies the *Karush Kuhn-Tucker* conditions, hence we can rewrite it as

$$\underset{\alpha>0}{\operatorname{argmax}} \ \min_{W,b} ||W||^2 + \sum_i \alpha_i \left(Y_i \left(W^T X_i - b\right) - 1\right)$$

Linearly separable case: Lagrangian formalism

- Under the KKT conditions

$$\underset{\alpha>0}{\operatorname{argmax}} \ \min_{W,b} \|W\|^2 + \sum_i \alpha_i \left(Y_i\left(W^T X_i - b\right) - 1\right)$$

- Taking the deriviative w.r.t W and setting to 0, we get

$$2W = -\sum_i \alpha_i Y_i X_i$$

Linearly separable case: Lagrangian formalism

- Under the KKT conditions

$$\underset{\alpha>0}{\operatorname{argmax}}\ \min_{W,b} \|W\|^2 + \sum_i \alpha_i \left(Y_i\left(W^T X_i - b\right) - 1\right)$$

- Taking the deriviative w.r.t b and setting to 0, we get

$$0 = \sum_i \alpha_i Y_i$$

Linearly separable case:

- Restating (and ignoring the factor of 2)

$$\underset{\alpha>0}{\operatorname{argmax}} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j Y_i Y_j X_i^T X_j - b \sum_i \alpha_i Y_i$$

- Since the last term is 0

$$\underset{\alpha}{\operatorname{argmax}} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j Y_i Y_j X_i^T X_j$$

$$s.t. \alpha_i \geq 0$$

$$\sum_i \alpha_i Y_i = 0$$

Large Margin Linear Classifier with Slack

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \ \sum_{i=1}^{n} \alpha_i - \ \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

such that

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

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

The usual simple SVM can also be solved through the ugly form

$$\underset{\alpha}{\operatorname{argmax}} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j Y_i Y_j X_i^T X_j$$

$$s.t.\ C \geq \alpha_i \geq 0$$

$$\sum_i \alpha_i Y_i = 0$$

- This is for the linear case. Note that the optimization is in terms of $X_i^T X_j$
- Also $W = -\sum_i \alpha_i Y_i X_i$
- So the classifier on any test instance has the form:

$$sign\left(-\sum_i \alpha_i Y_i X_{test}^T X_i - b\right)$$

The Kernel Trick

$$\underset{\alpha}{\operatorname{argmax}} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j Y_i Y_j X_i^T X_j$$

$$s.t. \, C \geq \alpha_i \geq 0$$

$$\sum_i \alpha_i Y_i = 0$$

- This is for the linear case. Note that the optimization is in terms of $X_i^T X_j$
- Also $W = -\sum_i \alpha_i Y_i X_i$
- So the classifier on any test instance has the form:

$$sign\left(-\sum_i \alpha_i Y_i X_{test}^T X_i - b\right)$$

The Kernel Trick

$$\underset{\alpha}{\text{argmax}} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j Y_i Y_j K(X_i, X_j)$$

$$s.t. \; C \geq \alpha_i \geq 0$$

$$\sum_i \alpha_i Y_i = 0$$

- For classification:

$$sign\left(-\sum_i \alpha_i Y_i K(X_i, X_{test}) - b\right)$$

The Kernel Trick

$$\underset{\alpha}{\operatorname{argmax}} \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j Y_i Y_j K(X_i, X_j)$$

$$s.t.\, C \geq \alpha_i \geq 0$$

$$\sum_i \alpha_i Y_i = 0$$

This is a quadratic programming problem

- For classification:

$$sign\left(-\sum_i \alpha_i Y_i K(X_i, X_{test}) - b\right)$$

Nonlinear SVMs: The Kernel Trick

- Examples of commonly-used kernel functions:
  - Linear kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
  - Polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
  - Gaussian (Radial-Basis Function (RBF)) kernel:

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

  - Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$$

- In general, functions that satisfy *Mercer's condition* can be kernel functions.

Nonlinear SVM: Optimization

- Formulation: (Lagrangian Dual Problem)

$$\text{maximize} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

such that

$$0 \le \alpha_i \le C$$

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

- The solution of the discriminant function is

$$g(\mathbf{x}) = \sum_{i \in \mathrm{SV}} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- The optimization technique is the same.

Support Vector Machine: Algorithm

- 1. Choose a kernel function
- 2. Choose a value for *C*
- 3. Solve the quadratic programming problem (many software packages available)
- 4. Construct the discriminant function from the support vectors

Some Issues

- Choice of kernel
  - Gaussian or polynomial kernel is default
  - if ineffective, more elaborate kernels are needed
  - domain experts can give assistance in formulating appropriate similarity measures

- Choice of kernel parameters
  - e.g. σ in Gaussian kernel
  - σ is the distance between closest points with different classifications
  - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.

- Optimization criterion – Hard margin v.s. Soft margin
  - a lengthy series of experiments in which various parameters are tested

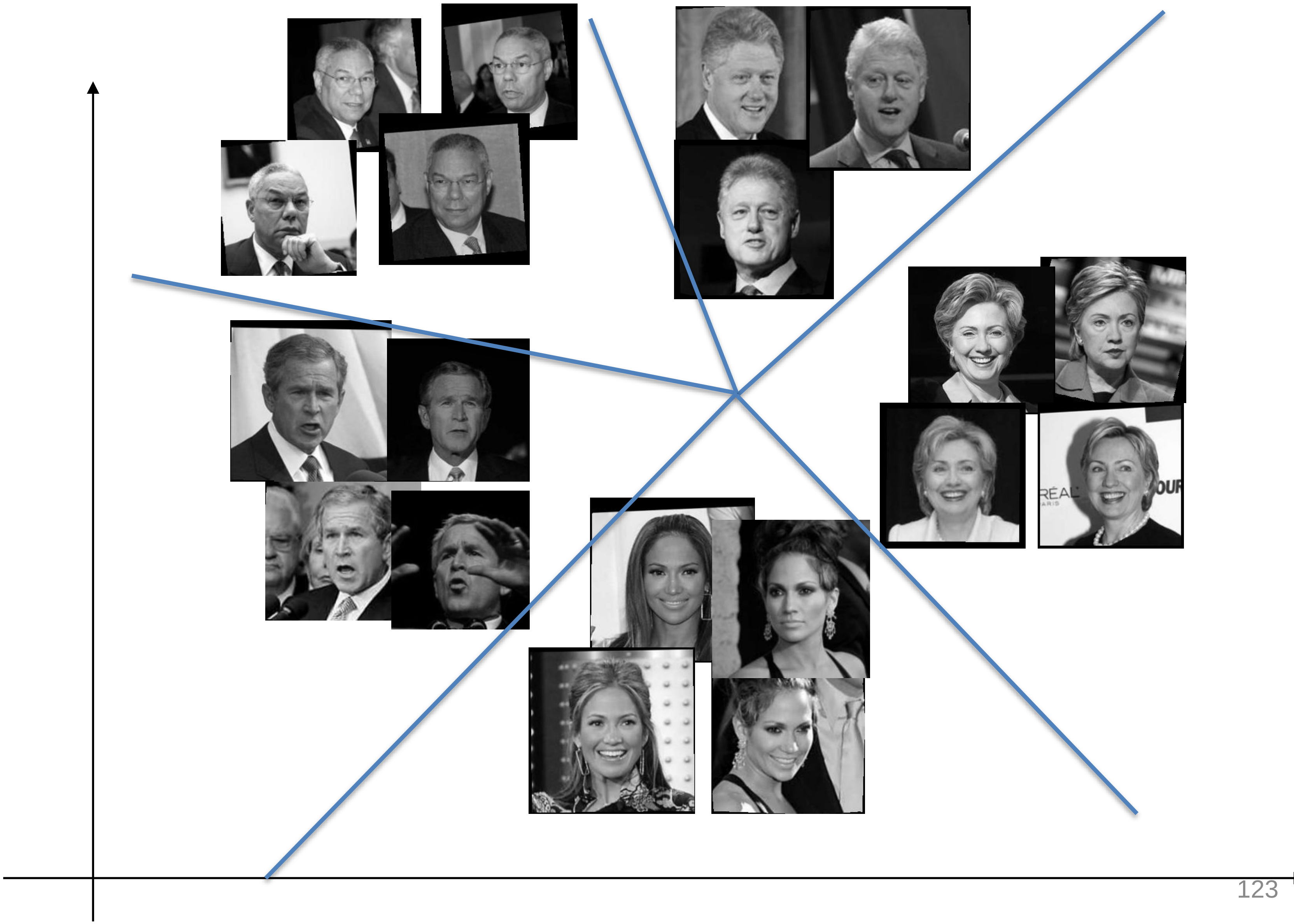
This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Summary: Support Vector Machine

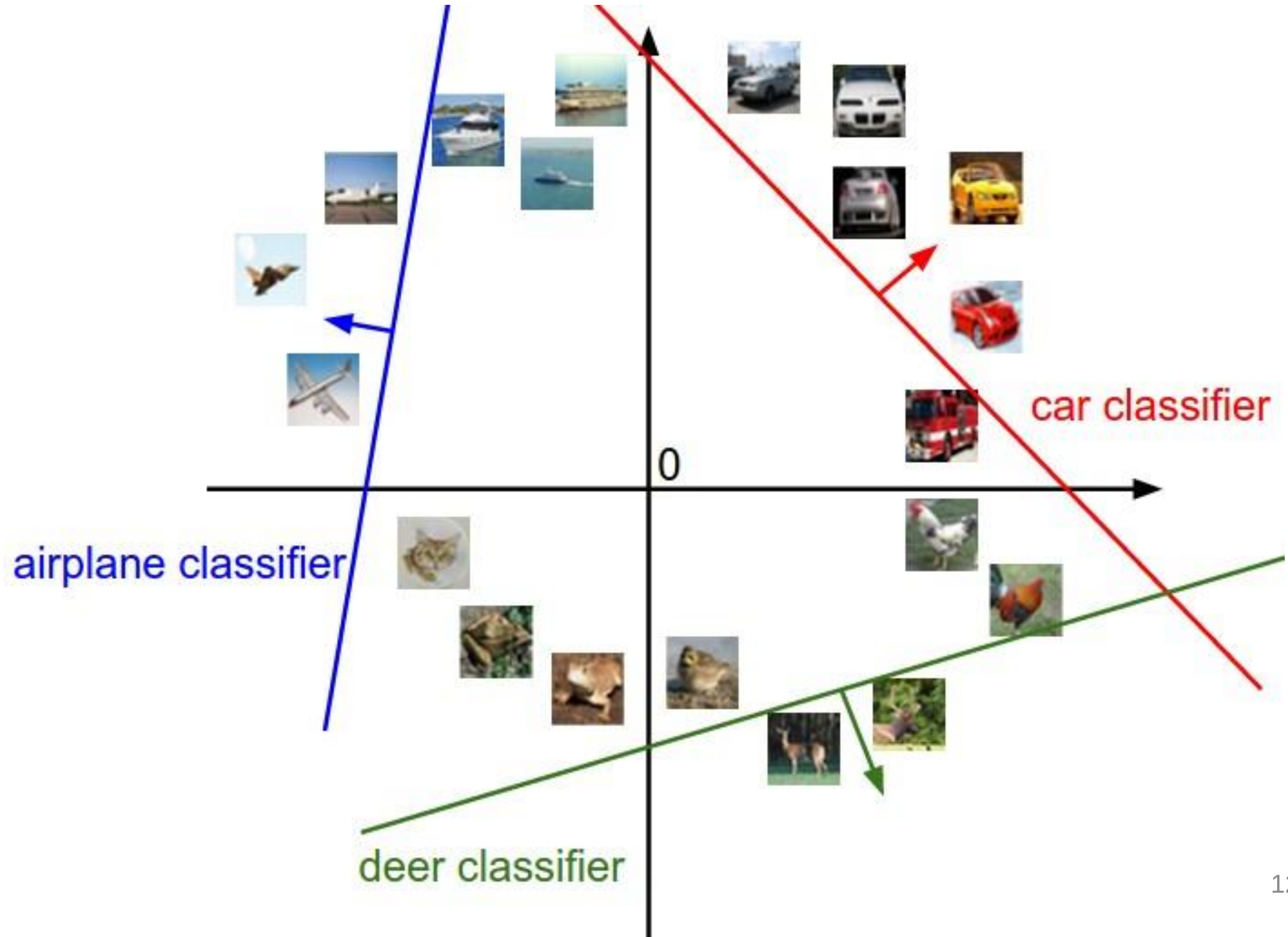
- 1. Large Margin Classifier
  - Better generalization ability & less over-fitting

- 2. The Kernel Trick
  - Map data points to higher dimensional space in order to make them linearly separable.
  - Since only dot product is used, we do not need to represent the mapping explicitly.

Multi-class generalization Pairwise



Multi-class generalization One-vs-all



Linear Classifiers: Conclusion

- Simple linear classifiers can be surprisingly effective
  - Particularly when trained to maximize a margin
    - Whereupon the "simple" arithmetic magically becomes complicated

- Kernel trick enables classification of even non-linear problems
- Most commonly used classifier, still

In the next lecture

- Digital multimedia: Recording and devices
  - Audio
  - Images
  - Video
  - Text
- Digital multimedia: Processing
  - Audio processing
  - Two generic processing techniques