COMP 551 – Applied Machine Learning
Lecture 11: Support Vector Machines

Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca*)

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Today's quiz

- In the random forest approach proposed by Breiman, how many hyper-parameters need to be specified?
  - 1, 2, 3, 4, 5
- What is the complexity of each iteration of Adaboost, assuming your weak learner is a decision stump and you have all binary variables? Let M be the number of features and N be the number of examples.
  - O(M), O(N), O(MN), O(MN^2)
- Which of the two ensemble strategies is most effective for high variance base classifiers?
  - Bagging, Boosting

Project #2

#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲1	💫💫 **2045** 💫💫			0.81390	27	3d
2	▼1	**ZSV**			0.80887	16	1d
3	▲3	**DMT**			0.79067	8	3h
4	new	**Lazy Sloths**			0.78963	5	5d
5	▼1	**Nothing but Nets**			0.78929	15	10h
6	▼3	**Bluehorsens**			0.78684	3	7d
7	new	**vicrep**			0.78569	4	3d
8	new	**rdali**			0.78417	8	1d
9	new	**BBC News**			0.78338	5	3h
10	new	**NotoriousLanguageClassifiers**			0.78150	2	5d

Outline

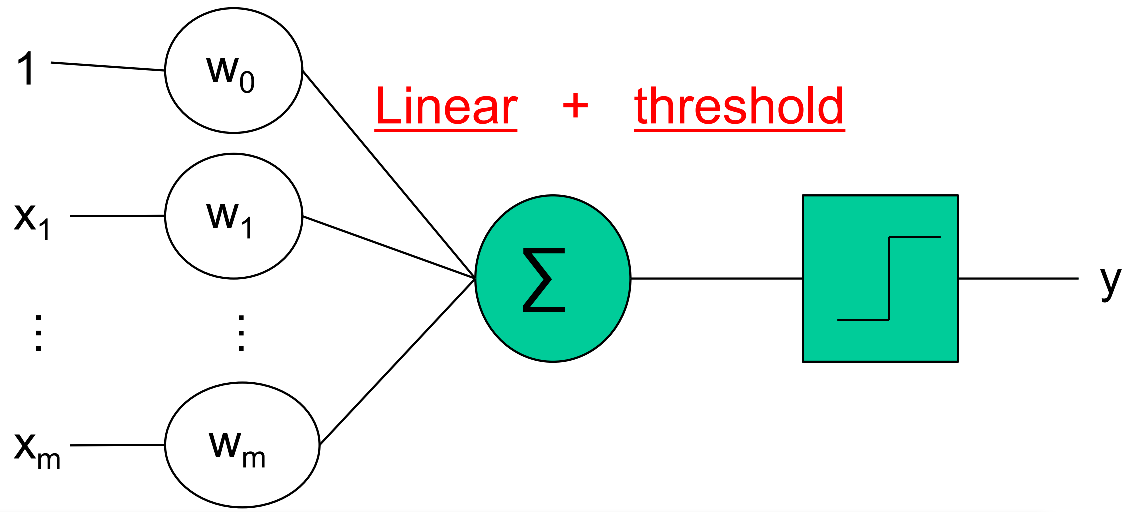
- Perceptrons
  - Definition
  - Perceptron learning rule
  - Convergence
- Margin & max margin classifiers
- Linear Support Vector Machines
  - Formulation as optimization problem
  - Generalized Lagrangian and dual
- Non-linear Support Vector Machines (next class)

A simple linear classifier

- Given a binary classification task: $\{\mathbf{x}_i, y_i\}_{i=1:n}$, $y_i=\{-1,+1\}$.
- The **perceptron** (Rosenblatt, 1957) is a classifier of the form:

$$h_{\mathbf{w}}(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x}) = \{+1 \text{ if } \mathbf{w}^T\mathbf{x} \geq 0; \; -1 \text{ otherwise}\}$$

  - The decision boundary is $\mathbf{w}^T\mathbf{x}=0$.
  - An example $<\mathbf{x}_i, y_i>$ is classified correctly if and only if: $y_i(\mathbf{w}^T\mathbf{x}_i)>0$.



Perceptron learning rule (Rosenblatt, 1957)

- Consider the following procedure:

> Initialize w_j, $j=0{:}m$ randomly,
>
> While any training examples remain incorrectly classified:
>
> Loop through all misclassified examples x_i
>
> Perform the update: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\, y_i\, \boldsymbol{x}_i$
>
> where α is the learning rate (or step size).

- **Intuition**: For misclassified positive examples, increase $w^T x$, and reduce it for negative examples.

Gradient-descent learning

- The perceptron learning rule can be interpreted as a **gradient descent procedure**, with optimization criterion:

$$Err(w) = \sum_{i=1:n} \{ 0 \text{ if } y_i \mathbf{w}^T \mathbf{x_i} \geq 0; \; -y_i \mathbf{w}^T \mathbf{x} \text{ otherwise } \}$$

Gradient-descent learning

- The perceptron learning rule can be interpreted as a **gradient descent procedure**, with optimization criterion:

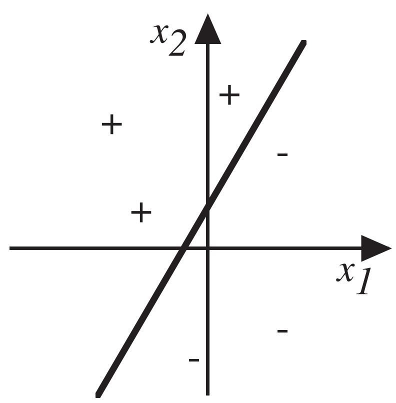
$$Err(w) = \sum_{i=1:n} \{ 0 \text{ if } y_i \mathbf{w}^T \mathbf{x_i} \geq 0; \ -y_i \mathbf{w}^T \mathbf{x} \text{ otherwise } \}$$

- For correctly classified examples, the error is zero.
- For incorrectly classified examples, the error tells by how much $w^T x$ is on the wrong side of the decision boundary.
- The error is zero when all examples are classified correctly.

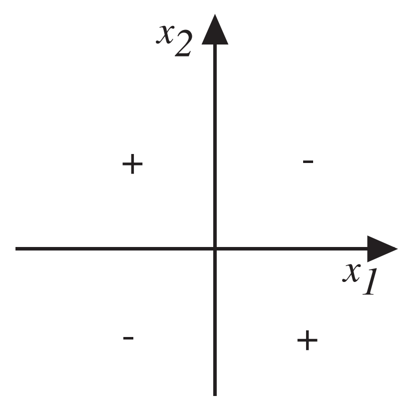
Linear separability

The data is **linearly separable** if and only if there exists a ***w*** such that:

- For all examples, $y_i \mathbf{w}^T \mathbf{x}_i > 0$
- Or equivalently, the 0-1 loss is zero for some set of parameters (***w***).



Linearly separable



Not linearly separable

Perceptron convergence theorem

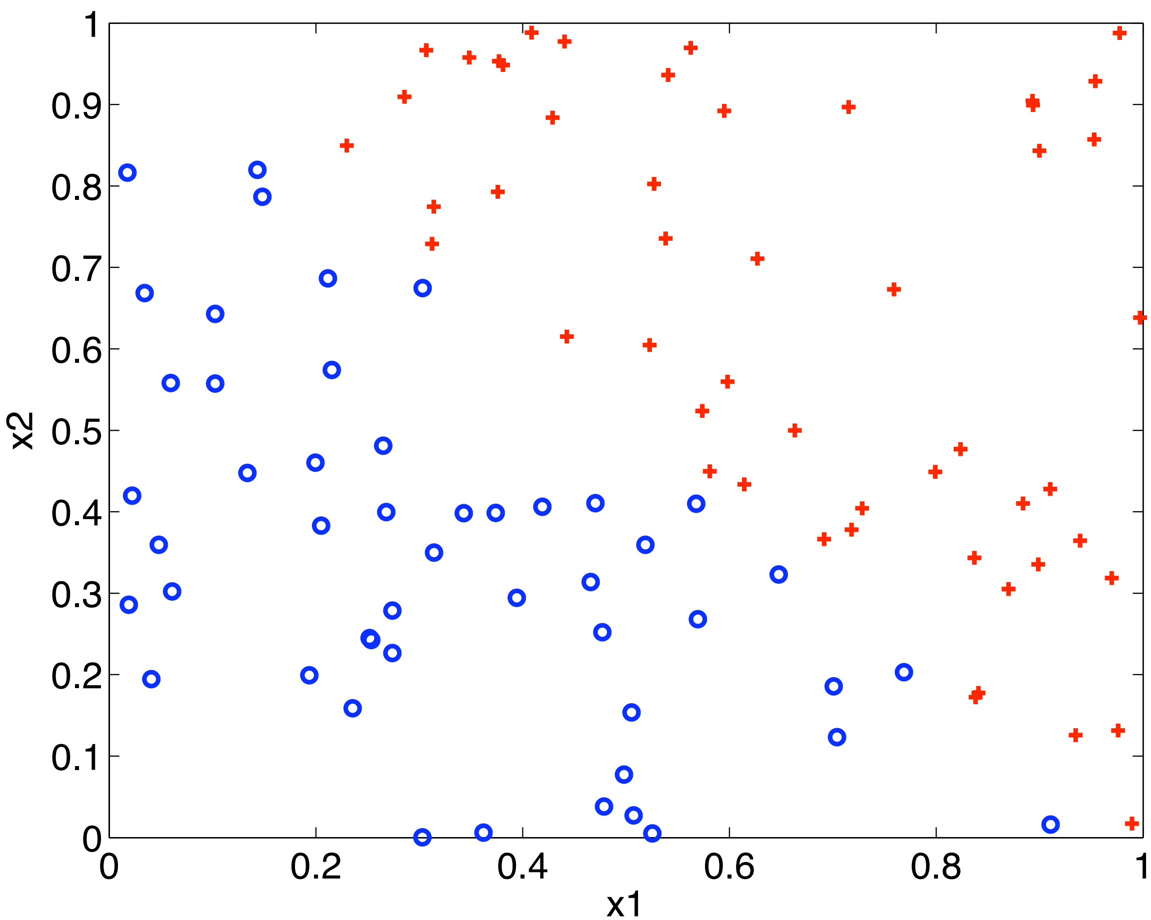
- The basic theorem:
  - If the perceptron learning rule is applied to a linearly separable dataset, a solution will be found after some finite number of updates.

Perceptron convergence theorem

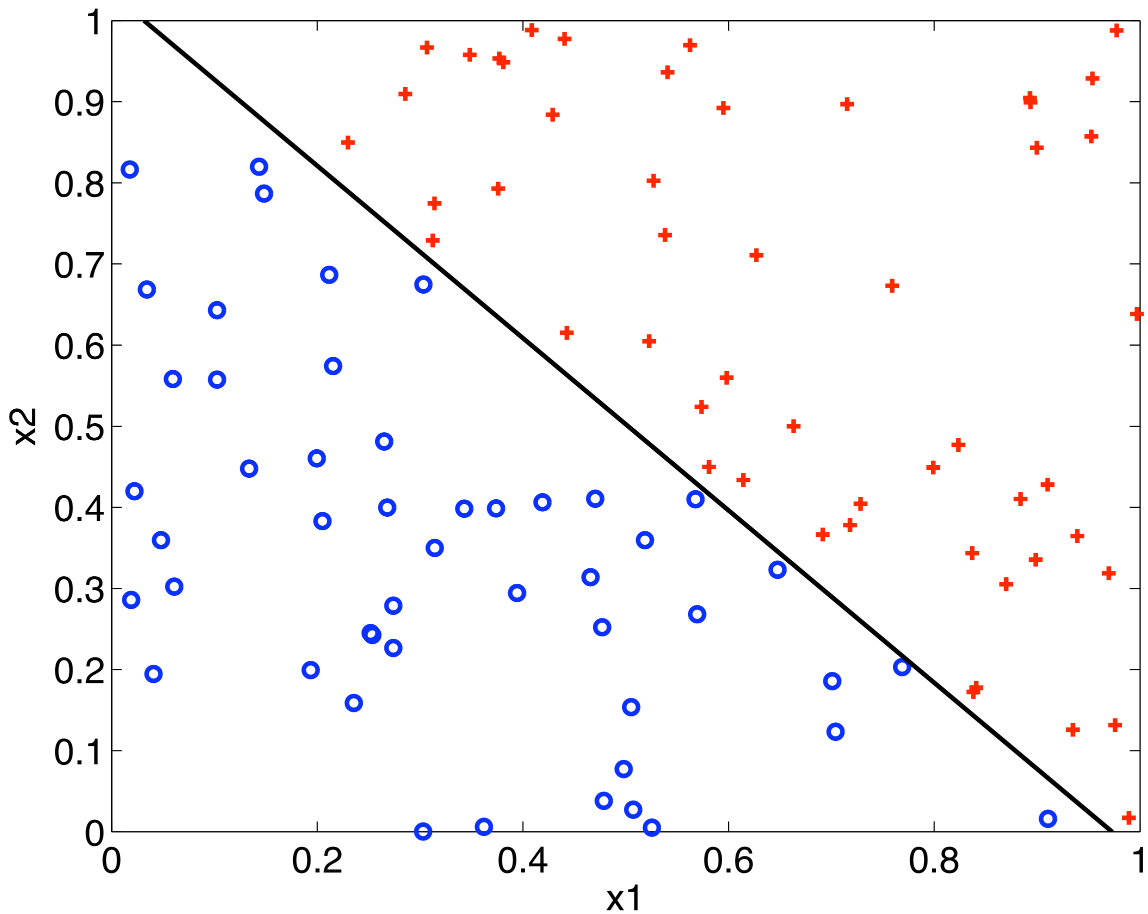
- The basic theorem:
  - If the perceptron learning rule is applied to a linearly separable dataset, a solution will be found after some finite number of updates.

- Additional comments:
  - The number of updates depends on the dataset, on the learning rate, and on the initial weights.
  - If the data is not linearly separable, there will be oscillation (which can be detected automatically).
  - Decreasing the learning rate to 0 can cause the oscillation to settle on some particular solution.

Perceptron learning example



Perceptron learning example



Weight as a combination of input vectors

- Recall perceptron learning rule:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\, y_i\, \boldsymbol{x}_i$$

- If initial weights are zero, then at any step, the weights are a linear combination of feature vectors of the examples:

$$\boldsymbol{w} = \sum_{i=1:n} \alpha_i\, y_i\, \boldsymbol{x}_i$$

where α_i is the sum of step sizes used for all updates applied to example i.

Weight as a combination of input vectors

- Recall perceptron learning rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\, y_i\, \mathbf{x}_i$$

- If initial weights are zero, then at any step, the weights are a linear combination of feature vectors of the examples:

$$\mathbf{w} = \sum_{i=1:n} \alpha_i\, y_i\, \mathbf{x}_i$$

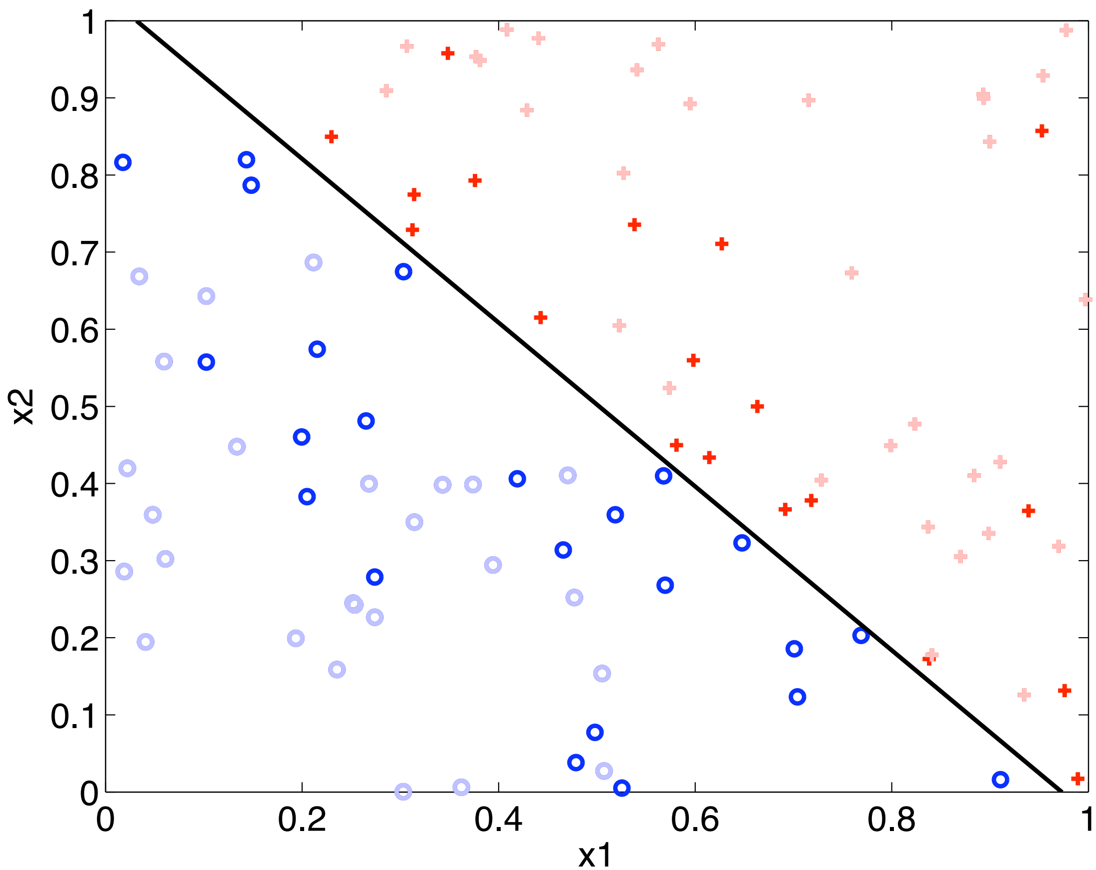
where α_i is the sum of step sizes used for all updates applied to example i.

- By the end of training, some examples may have never participated in an update, so will have $\alpha_i=0$.

- This is called the **dual representation** of the classifier.

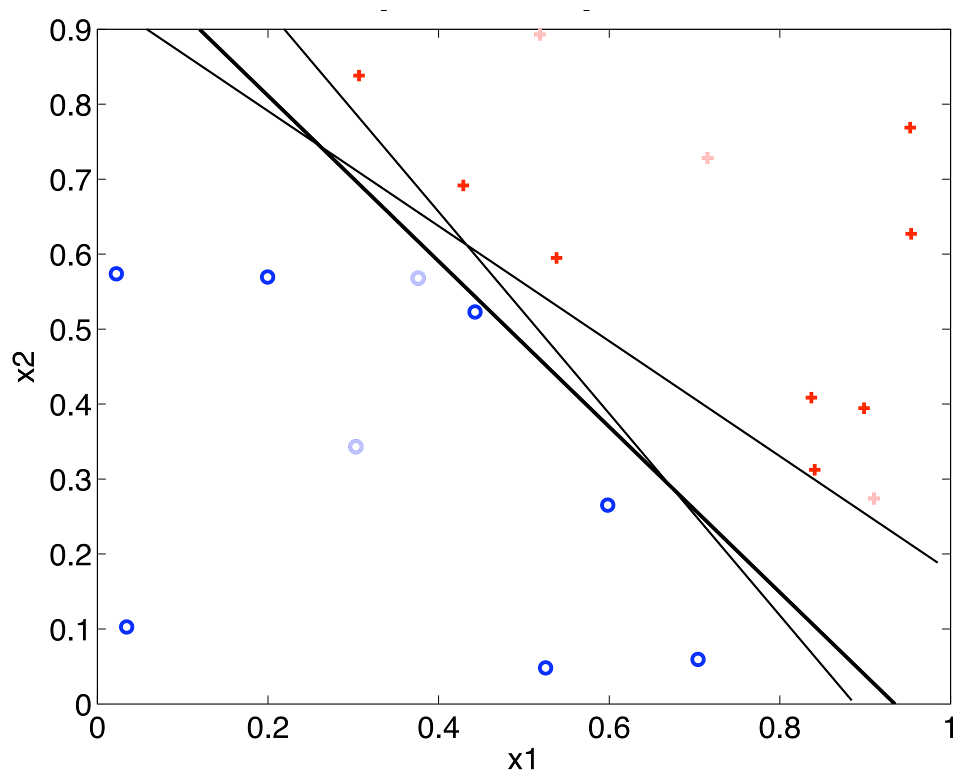
Perceptron learning example

- Examples used (bold) and not (faint). What do you notice?



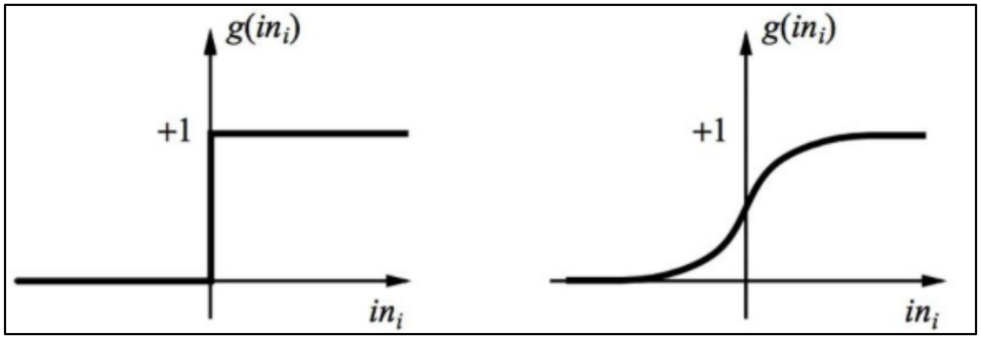
Perceptron learning example

- Solutions are often non-unique. The solution depends on the set of instances and the order of sampling in updates.



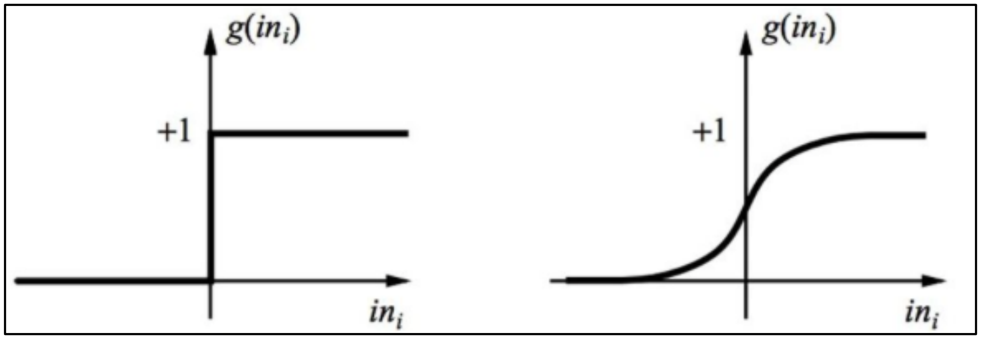
A few comments on the Perceptron

- Perceptrons can be learned to fit <u>linearly separable</u> data, using a gradient-descent rule.
  - The logistic function offers a "smooth" version of the perceptron.



A few comments on the Perceptron

- Perceptrons can be learned to fit linearly separable data, using a gradient-descent rule.
  – The logistic function offers a "smooth" version of the perceptron.

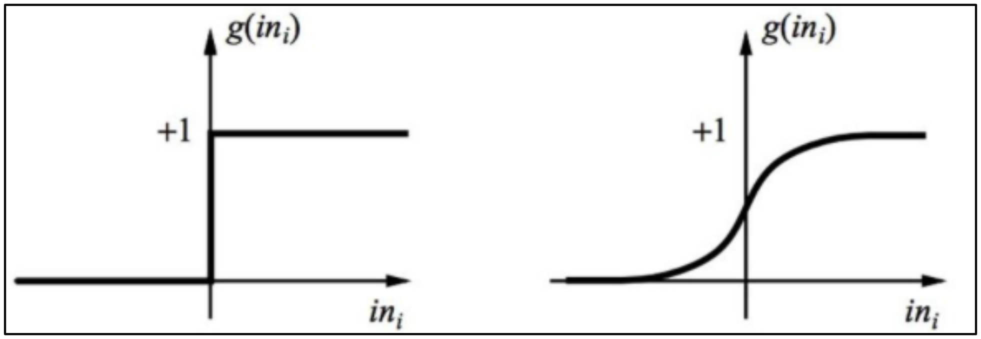


Two issues:

- Solutions are non-unique.
- What about non-linearly separable data?

A few comments on the Perceptron

- Perceptrons can be learned to fit linearly separable data, using a gradient-descent rule.
  - The logistic function offers a "smooth" version of the perceptron.

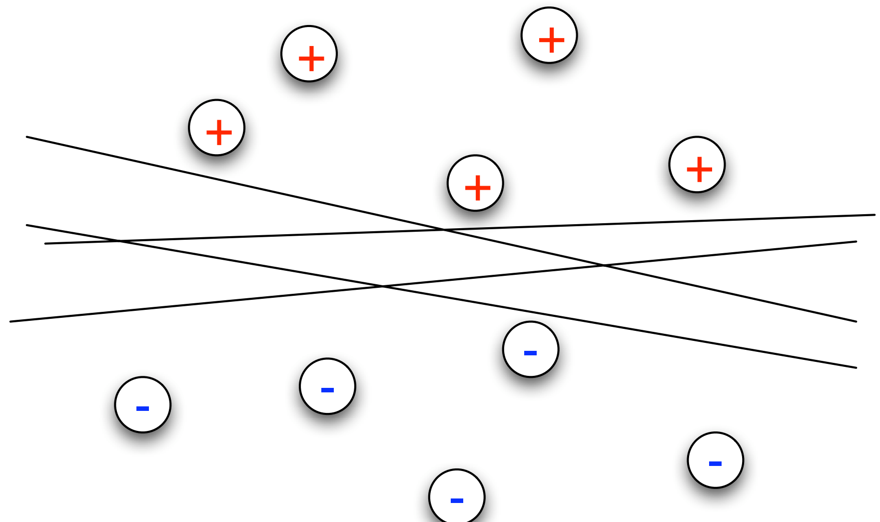


Two issues:

- Solutions are non-unique.
- What about non-linearly separable data? *(Topic for next class.)*
  - Perhaps data can be linearly separated in a different feature space?
  - Perhaps we can relax the criterion of separating all the data?

The non-uniqueness issue

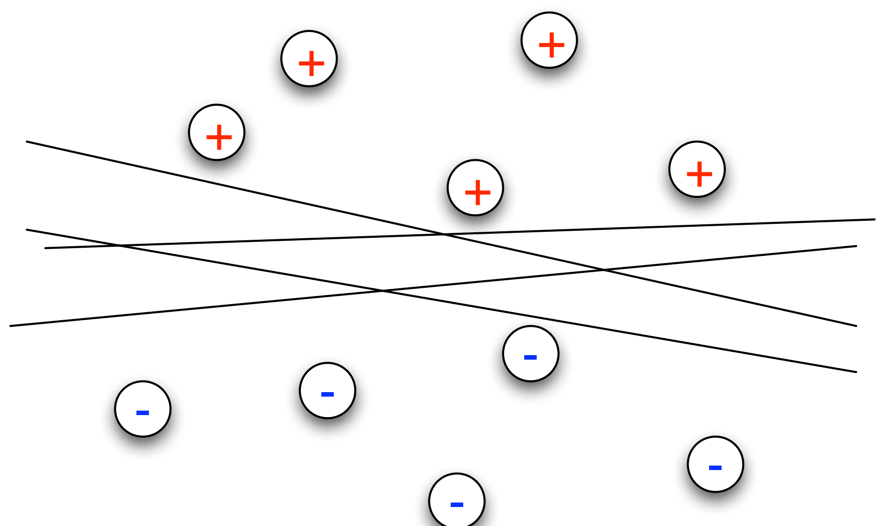
- Consider a linearly separable binary classification dataset.
- There is an infinite number of hyper-planes that separate the classes:



- **Which plane is best?**

The non-uniqueness issue

- Consider a linearly separable binary classification dataset.
- There is an infinite number of hyper-planes that separate the classes:
- **Which plane is best?**

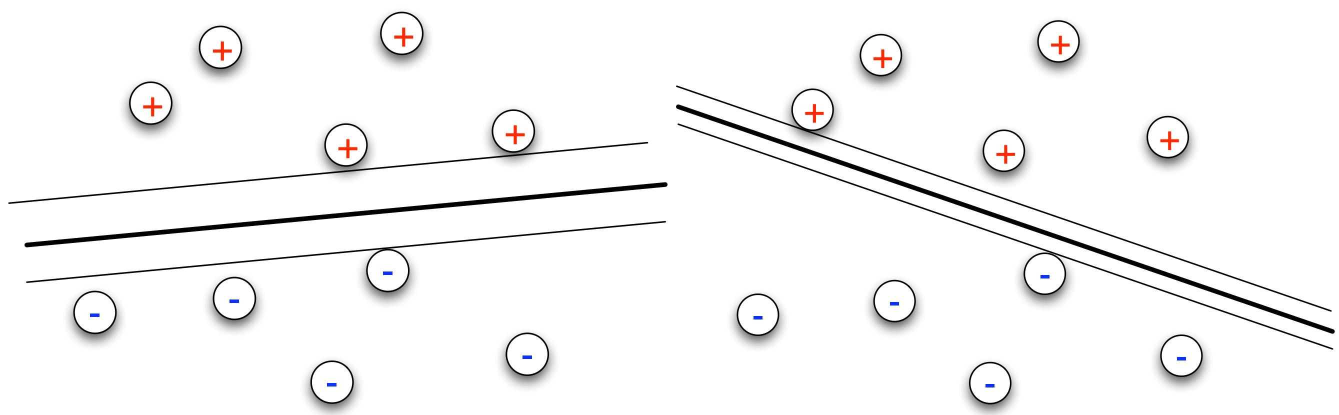


- Related question: For a given plane, for which points should we be most confident in the classification?

Linear Support Vector Machine (SVM)

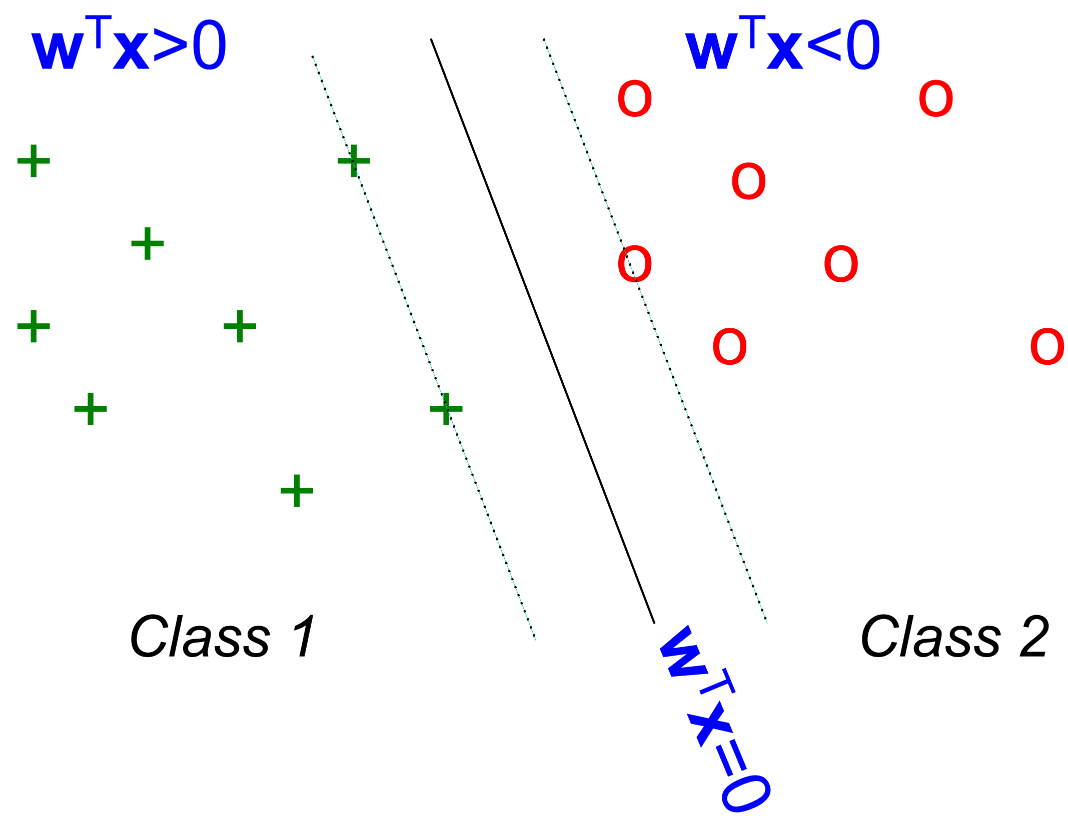
- A **linear** SVM is a perceptron for which we chose **w** such that the margin is maximized.

- For a given separating hyper-plane, the **margin** is twice the (Euclidean) distance from hyper-plane to nearest training example.
  - I.e. the width of the "strip" around the decision boundary that contains no training examples.



Distance to the decision boundary

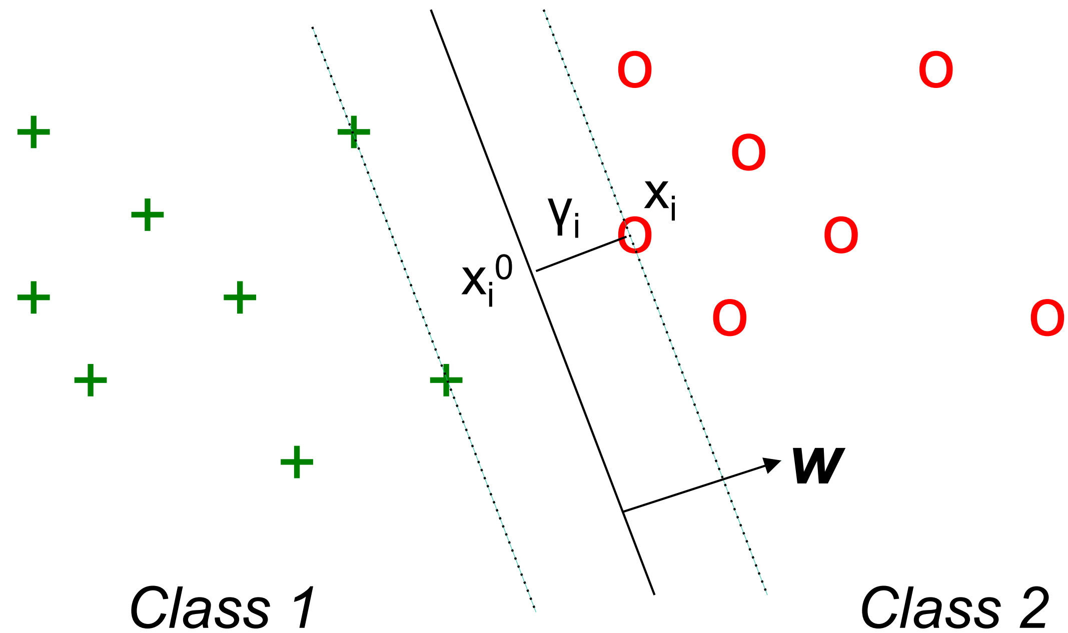
- Suppose we have a decision boundary that separates the data.



- Assuming $y_i = \{-1, +1\}$, *"confidence"* = $y_i \mathbf{w}^T \mathbf{x}_i$

Distance to the decision boundary

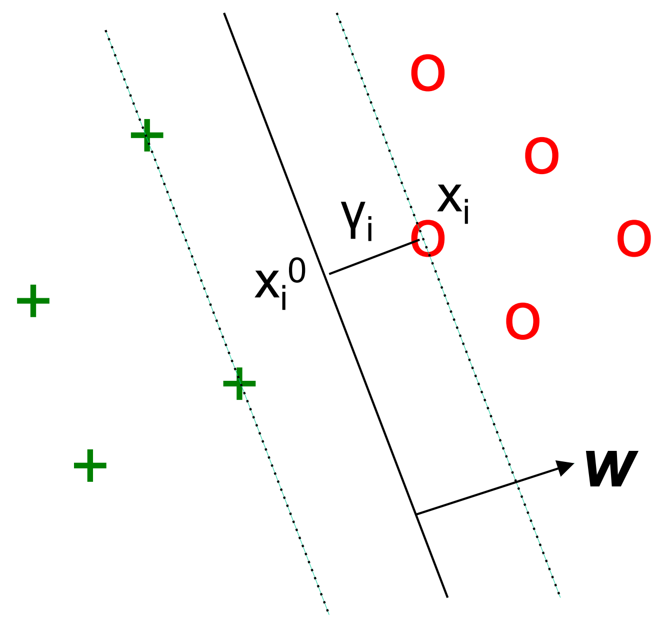
- Suppose we have a decision boundary that separates the data.



- Let γ_i be the distance from instance $\boldsymbol{x_i}$ to the decision boundary.
- Define vector $\boldsymbol{w}$ to be the normal to the decision boundary.

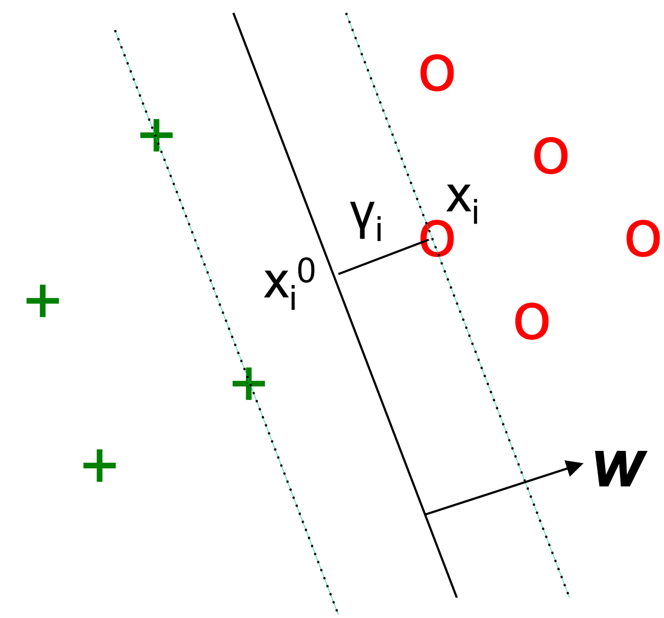
Distance to the decision boundary

- How can we write γ_i in terms of $\boldsymbol{x}_i$, y_i, $\boldsymbol{w}$?
- Let $\boldsymbol{x}_i^0$ be the point on the decision boundary nearest $\boldsymbol{x}_i$
- The vector from $\boldsymbol{x}_i^0$ to $\boldsymbol{x}_i$ is $\gamma_i \boldsymbol{w} / ||\boldsymbol{w}||$.
  - γ_i is a scalar (distance from $\boldsymbol{x}_i$ to $\boldsymbol{x}_i^0$)
  - $\boldsymbol{w}/||\boldsymbol{w}||$ is the unit normal.
- So we can define $\boldsymbol{x}_i^0 = \boldsymbol{x}_i - \gamma_i \boldsymbol{w} / ||\boldsymbol{w}||$.



Distance to the decision boundary

- How can we write γ_i in terms of $\mathbf{x}_i$, y_i, $\mathbf{w}$?
- Let $\mathbf{x}_i^0$ be the point on the decision boundary nearest $\mathbf{x}_i$
- The vector from $\mathbf{x}_i^0$ to $\mathbf{x}_i$ is $\gamma_i \mathbf{w} / ||\mathbf{w}||$.
  - γ_i is a scalar (distance from $\mathbf{x}_i$ to $\mathbf{x}_i^0$)
  - $\mathbf{w}/||\mathbf{w}||$ is the unit normal.



- So we can define $\mathbf{x}_i^0 = \mathbf{x}_i - \gamma_i \mathbf{w} / ||\mathbf{w}||$.
- As $\mathbf{x}_i^0$ is <u>on</u> the decision boundary, we have

$$\mathbf{w}^T(\mathbf{x}_i - \gamma_i \mathbf{w} / ||\mathbf{w}||) = 0$$

- Solving for γ_i yields, for a positive example: $\gamma_i = \mathbf{w}^T\mathbf{x}_i / ||\mathbf{w}||$

  or for examples of both classes: $\gamma_i = y_i\mathbf{w}^T\mathbf{x}_i / ||\mathbf{w}||$

Optimization

- First suggestion:

  Maximize M

  with respect to $\mathbf{w}$

  subject to $y_i \mathbf{w}^T \mathbf{x}_i / ||\mathbf{w}|| \geq M, \ \forall i$

- This is not very convenient for optimization:
  - $\mathbf{w}$ appears nonlinearly in the constraints.
  - Problem is underconstrained. If $(\mathbf{w}, M)$ is optimal, so is $(\beta\mathbf{w}, M)$, for any $\beta>0$. **Add a constraint**: $||\mathbf{w}||M = 1$

- Instead try:

  Minimize $||w||$

  with respect to $\mathbf{w}$

  subject to $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$

Optimization

- First suggestion:

  Maximize M

  with respect to $\mathbf{w}$

  subject to $y_i \mathbf{w}^T \mathbf{x}_i \,/\, ||\mathbf{w}|| \geq M, \;\; \forall i$

- This is not very convenient for optimization:
  - $\mathbf{w}$ appears nonlinearly in the constraints.
  - Problem is underconstrained. If $(\mathbf{w}, M)$ is optimal, so is $(\beta\mathbf{w}, M)$, for any $\beta>0.$ **Add a constraint**: $||\mathbf{w}||M = 1$

- Instead try:

  Minimize $||w||$

  with respect to $\mathbf{w}$

  subject to $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$

Final formulation

- Let's minimize $\frac{1}{2}||\mathbf{w}||^2$ instead of $||\mathbf{w}||$

  (Taking the square is a monotone transform, as $||w||$ is positive, so it doesn't change the optimal solution. The ½ is for mathematical convenience.)

- This gets us to:

  Min $\frac{1}{2}||\mathbf{w}||^2$

  w.r.t. w

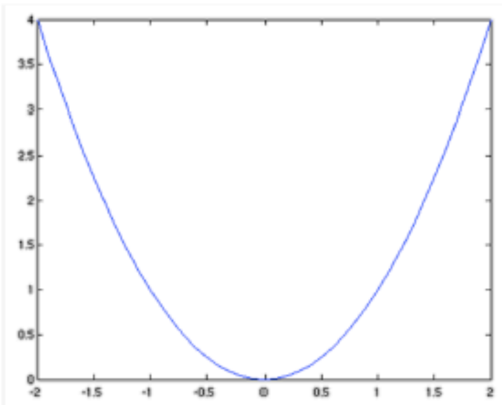
  s.t. $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$

- This can be solved! How?
  - It is a quadratic programming (QP) problem – a standard type of optimization problem for which many efficient packages are available. Better yet, it's a convex (positive semidefinite) QP.

Constrained optimization

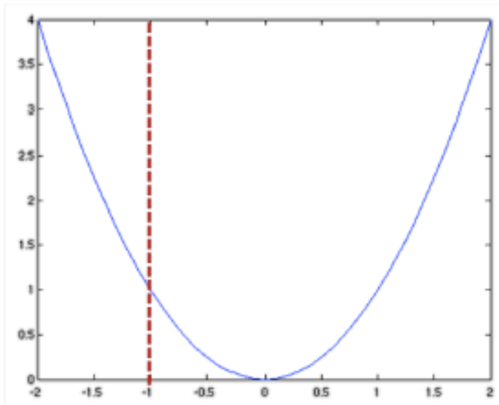
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

$$\min_x \; x^2$$



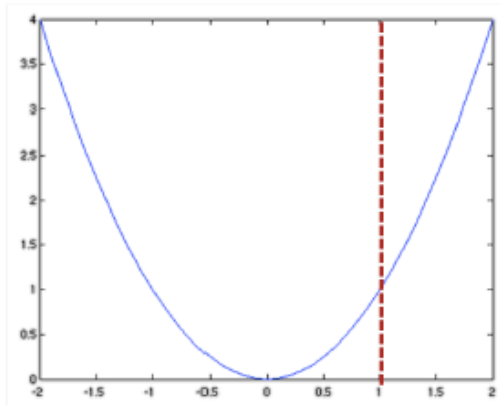
$$x^* = 0$$

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq -1$$



$$x^* = 0$$

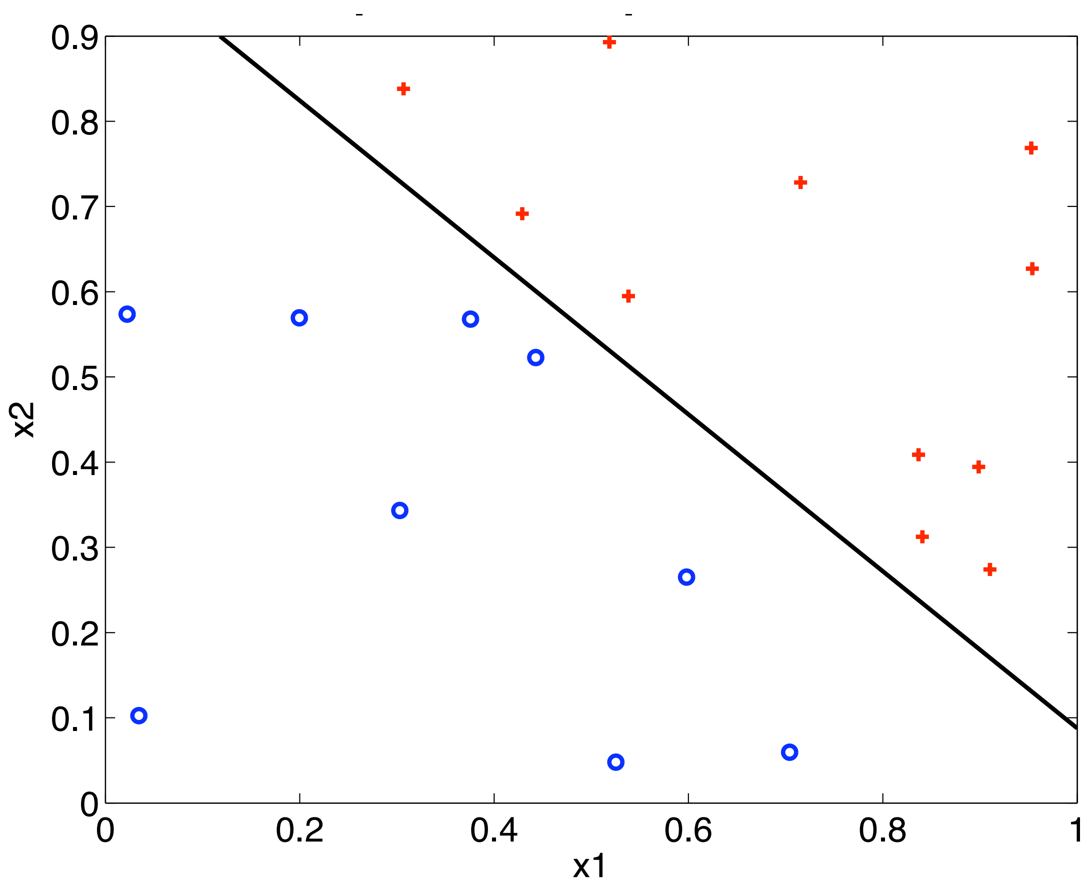
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq 1$$



$$x^* = 1$$

Picture from: http://www.cs.cmu.edu/~aarti/Class/10701_Spring14/

Example



We have a unique solution, but no support vectors yet.
Recall the dual solution for the Perceptron: Extend for the margin case.

Lagrange multipliers

- Consider the following optimization problem, called primal:

$$\min_{\mathbf{w}} \quad f(\mathbf{w})$$
$$\text{s.t.} \quad g_i(\mathbf{w}) \leq 0, \; i=1\ldots k$$

- We define the generalized Lagrangian:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = f(\mathbf{w}) + \sum_{i=1:k} \alpha_i \, g_i(\mathbf{w})$$

where α_i, $i=1\ldots k$ are the Lagrange multipliers.

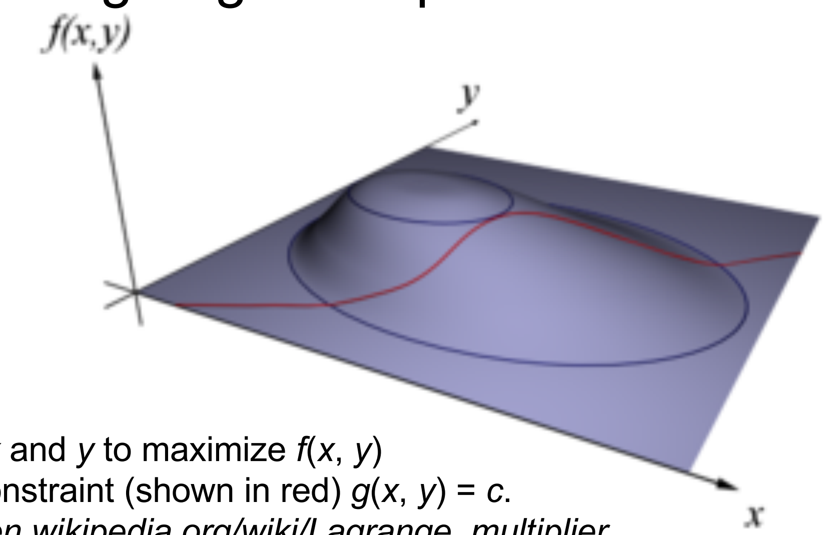


Figure : Find x and y to maximize $f(x, y)$ subject to a constraint (shown in red) $g(x, y) = c$.
From: *https://en.wikipedia.org/wiki/Lagrange_multiplier*

Lagrangian optimization

- Consider $P(\mathbf{w}) = max_{\boldsymbol{\alpha}:\alpha i \geq 0}\, L(\mathbf{w},\boldsymbol{\alpha})$ (P stands for "primal")
- Observe that the following is true:

$$P(\mathbf{w}) = \begin{cases} f(\mathbf{w}), & \text{if all constraints are satisfied,} \\ +\infty, & \text{otherwise} \end{cases}$$

- Hence, instead of computing $min_{\mathbf{w}}\, f(\mathbf{w})$ subject to the original constraints, we can compute:

$$p^* = min_{\mathbf{w}}\, P(\mathbf{w}) = min_{\mathbf{w}}\, max_{\boldsymbol{\alpha}:\alpha i \geq 0}\, L(\mathbf{w},\boldsymbol{\alpha})$$ **Primal**

- Alternately, invert max and min to get:

$$d^* = max_{\boldsymbol{\alpha}:\alpha i \geq 0}\, min_{\mathbf{w}}\, L(\mathbf{w},\boldsymbol{\alpha})$$ **Dual**

Maximum Margin Perceptron

- We wanted to solve: *Min* $\frac{1}{2}||\mathbf{w}||^2$

  w.r.t. $\mathbf{w}$

  s.t. $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$

- The Lagrangian is:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 + \sum_i \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i))$$

- The primal problem is: $\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}:\alpha_i \geq 0} L(\mathbf{w}, \boldsymbol{\alpha})$

- The dual problem is: $\max_{\boldsymbol{\alpha}:\alpha_i \geq 0} \min_{w} L(\mathbf{w}, \boldsymbol{\alpha})$

Dual optimization problem

- Consider both solutions:

  $p^* = \min_{\mathbf{w}} \max_{\alpha:\alpha i \geq 0} L(\mathbf{w},\alpha)$ ***Primal***

  $d^* = \max_{\alpha:\alpha i \geq 0} \min_{\mathbf{w}} L(\mathbf{w},\alpha)$ ***Dual***

- If f and g_i are convex and the g_i can all be satisfied simultaneously for some $\mathbf{w}$, then we have equality: $d^* = p^* = L(\mathbf{w}^*, \boldsymbol{\alpha}^*)$.
  - $\mathbf{w}^*$ is the optimal weight vector (= primal solution)
  - $\boldsymbol{\alpha}^*$ is the optimal set of support vectors (=dual solution)
  - For SVMs, we have a quadratic objective and linear constraints so both f and g_i are convex.
  - For linearly separable data, all g_i can be satisfied simultaneously.
  - Note: $\mathbf{w}^*$, $\boldsymbol{\alpha}^*$ solve the primal and dual if and only if they satisfy the Karush-Kunh-Tucker conditions (*see suggested readings*).

Solving the dual

- Taking derivatives of $L(\mathbf{w}, \boldsymbol{\alpha})$ wrt $\mathbf{w}$, setting to 0, and solving for $\mathbf{w}$:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \tfrac{1}{2} ||\mathbf{w}||^2 + \sum_i \alpha_i (1 - y_i (\mathbf{w}^T \mathbf{x}_i))$$

$$\delta L / \delta \mathbf{w} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

$$\mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

- Just like for the perceptron with zero initial weights, the optimal solution $\mathbf{w}^*$ is a linear combination of the $\mathbf{x}_i$.

- Plugging this back into L we get the dual: $\max_\alpha \sum_i \alpha_i - \tfrac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i \cdot \mathbf{x})$

  with constraints $\alpha_i \geq 0$ and $\sum_i \alpha_i y_i = 0$. Quadratic programming problem.

- Complexity of solving quadratic program? Polynomial time, $O(|v|^3)$ (where $|v|$=# variables in optimization; here $|v|=n$). Fast approximations exist.

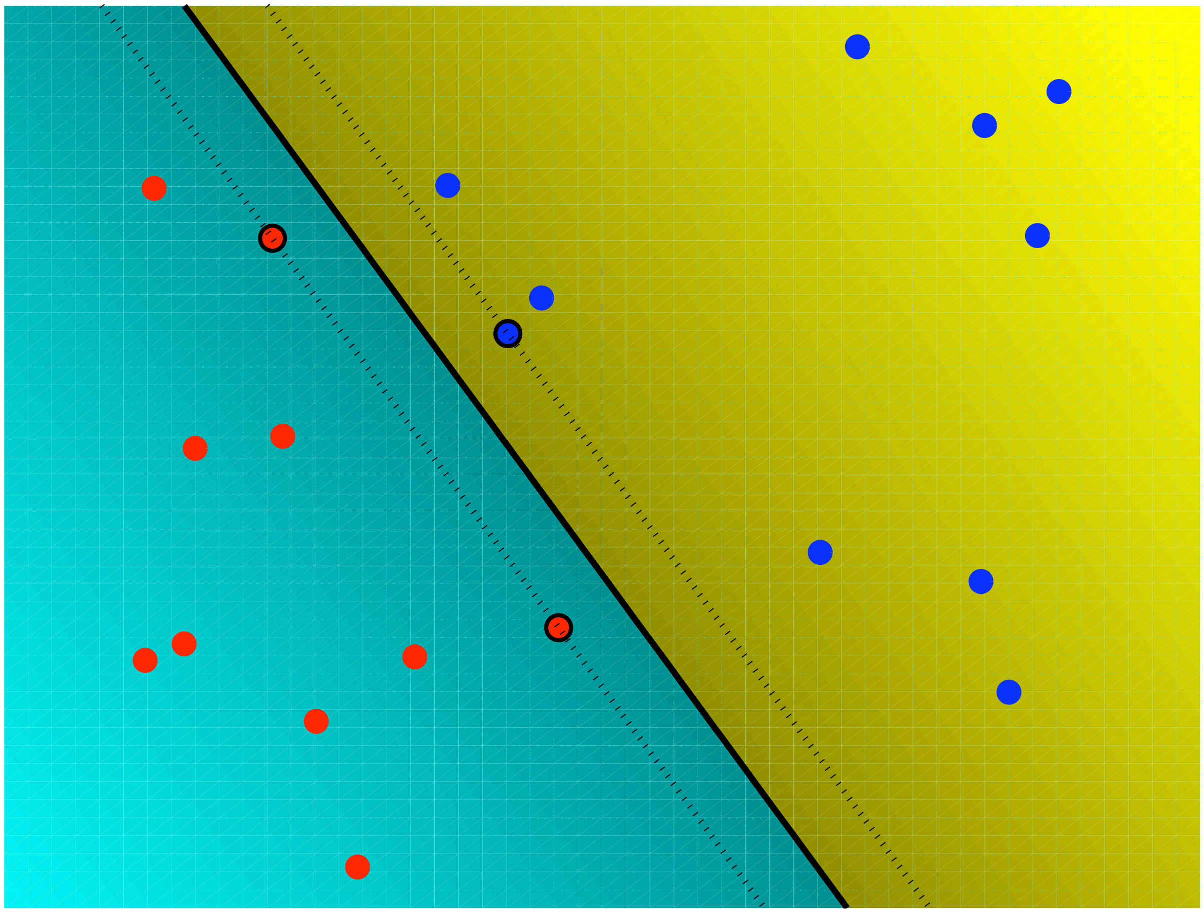
The support vectors

- Suppose we find the optimal α 's (e.g. using a QP package.)
- Constraint i is active when $\alpha_i > 0$. This corresponds for the points for which $(1-y_i \mathbf{w}^T \mathbf{x}_i)=0$.
- These are the points lying on the edge of the margin. We call them support vectors. They define the decision boundary.
- The output of the classifier for query point $\mathbf{x}$ is computed as:

$$h_{\mathbf{w}}(\mathbf{x}) = sign(\sum_{i=1:n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}))$$

It is determined by computing the dot product of the query point with the support vectors.

Example



Support vectors are in bold

What you should know

From today:

- The perceptron algorithm.
- The margin definition for linear SVMs.
- The use of Lagrange multipliers to transform optimization problems.
- The primal and dual optimization problems for SVMs.

After the next class:

- Non-linearly separable case.
- Feature space version of SVMs.
- The kernel trick and examples of common kernels.