Lecture 10. Soft-Margin SVM, Lagrangian Duality

COMP90051 Statistical Machine Learning

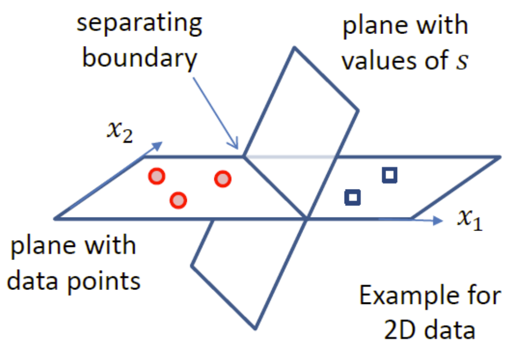
Semester 2, 2019
Lecturer: Ben Rubinstein

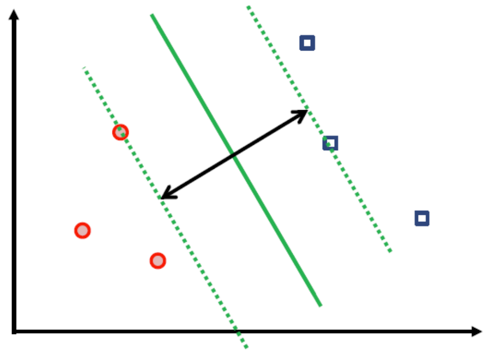
THE UNIVERSITY OF MELBOURNE

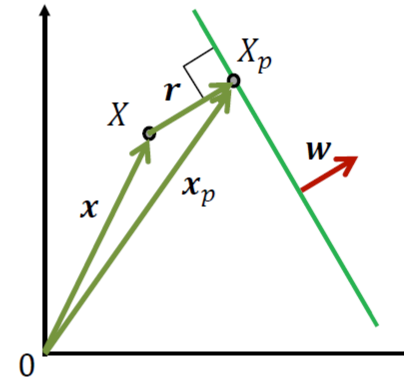
This lecture

- Soft-margin SVM
  - Intuition and problem formulation
- Lagrangian dual
  - Alternate formulation with different training complexity
  - Explains support vectors
  - Sets us up for kernels (next lectures)

Recap: hard-margin SVM







- SVM is a linear binary classifier
- Max margin: aim for boundary robust to noise
- Trick to resolve ambiguity $\frac{y_{i^*}(\boldsymbol{w}'\boldsymbol{x}_{i^*}+b)}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$
- Hard-margin program:

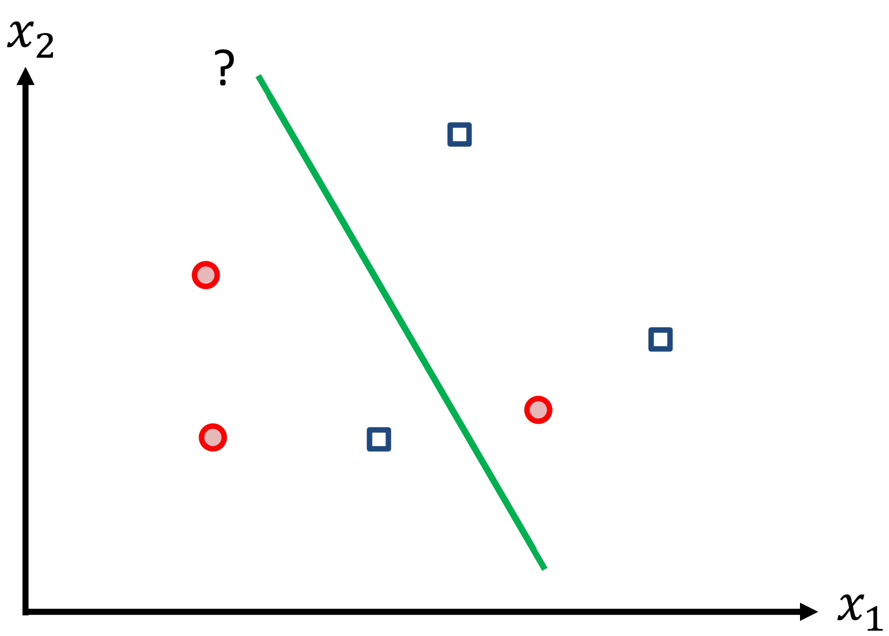
$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \|\boldsymbol{w}\| \text{ s.t. } y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 \text{ for } i = 1, \dots, n$$

Soft-Margin SVMs

Addressing linear inseparability

When data is not linearly separable

- Hard-margin loss is too stringent (*hard*!)
- Real data is unlikely to be linearly separable
- If the data is not separable, hard-margin SVMs are in trouble

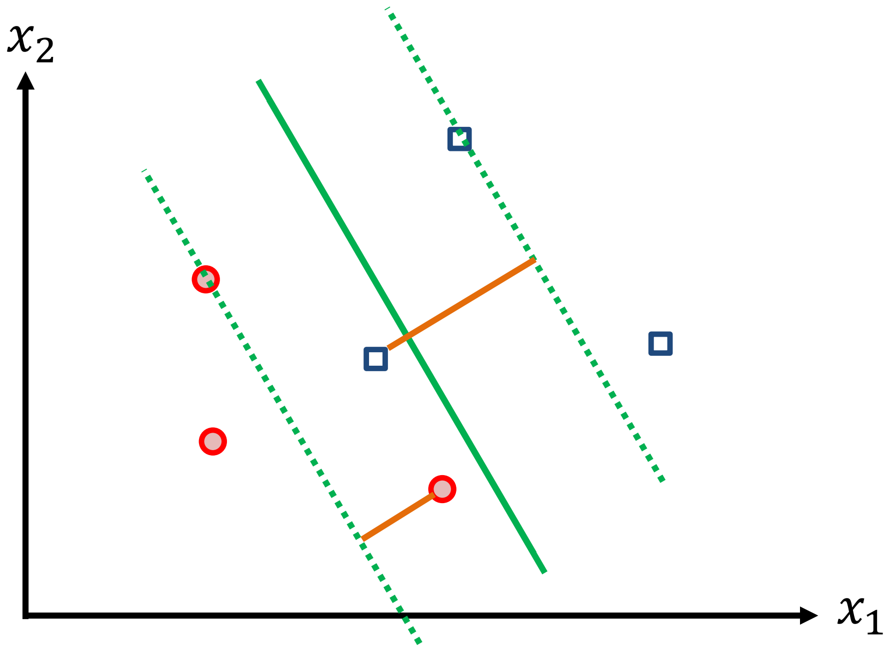


SVMs offer 3 approaches to address this problem:

1. *Still use hard-margin SVM, but transform the data (next lecture)*
2. *Relax the constraints (next slide)*
3. *The combination of 1 and 2 ☺*

Soft-margin SVM

- Relax constraints to allow points to be inside the margin or even on the wrong side of the boundary



However, we penalise boundaries by the extent of "violation"

In the figure, the objective penalty will take into account the orange distances

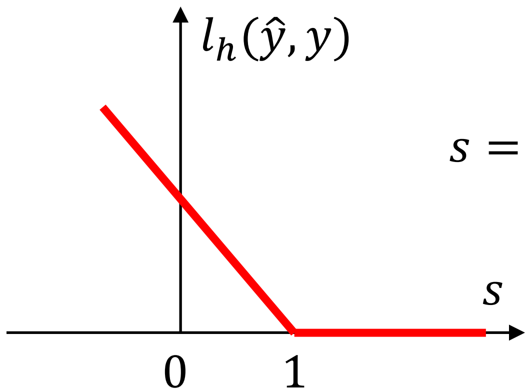
Hinge loss: soft-margin SVM loss

- Hard-margin SVM loss

$$l_\infty = \begin{cases} 0 & 1 - y(\boldsymbol{w}'\boldsymbol{x} + b) \leq 0 \\ \infty & otherwise \end{cases}$$

- Soft-margin SVM loss (hinge loss)

$$l_h = \begin{cases} 0 & 1 - y(\boldsymbol{w}'\boldsymbol{x} + b) \leq 0 \\ 1 - y\underbrace{(\boldsymbol{w}'\boldsymbol{x} + b)}_{\hat{y}} & otherwise \end{cases}$$



$s = y\hat{y}$

compare this with perceptron loss

Soft-margin SVM objective

- Soft-margin SVM objective

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}}\left(\sum_{i=1}^{n} l_h(\boldsymbol{x}_i, y_i, \boldsymbol{w}, b) + \lambda\|\boldsymbol{w}\|^2\right)$$

  * Reminiscent of ridge regression
  * Hinge loss $l_h = \max\left(0, 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)\right)$

- We are going to re-formulate this objective to make it more amenable to analysis

Re-formulating soft-margin objective

- Define slack variables as an upper bound on loss

$$\xi_i \geq l_h = \max\left(0, 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)\right)$$

or equivalently $\xi_i \geq 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)$ and $\xi_i \geq 0$

- Re-write the soft-margin SVM objective as:

$$\underset{\boldsymbol{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right)$$

s.t. $\xi_i \geq 1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b)$ for $i = 1, \dots, n$

$\xi_i \geq 0$ for $i = 1, \dots, n$

Side-by-side: Two variations of SVM

- Hard-margin SVM objective*:

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t. } y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 \text{ for } i = 1, \dots, n$$

- Soft-margin SVM objective:

$$\underset{\boldsymbol{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i\right)$$

$$\text{s.t. } y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1, \dots, n$$

$$\xi_i \geq 0 \text{ for } i = 1, \dots, n$$

- In the second case, the constraints are relaxed ("softened") by allowing violations by ξ_i. Hence the name "soft margin"

*Changed $\|\boldsymbol{w}\|$ to $0.5\|\boldsymbol{w}\|^2$ - monotonic increasing transform. Modified objective yields same solution.

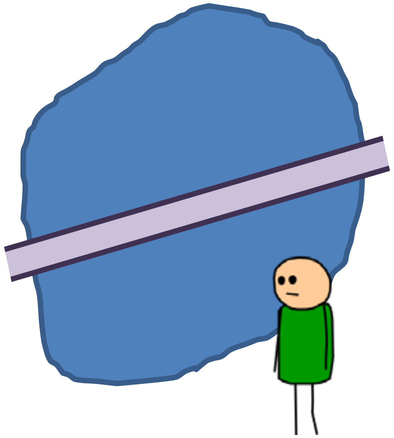
Lagrangian Duality for the SVM

An equivalent formulation, with important consequences.

Constrained optimisation

- Constrained optimisation: canonical form

$$\text{minimise } f(\boldsymbol{x})$$
$$\text{s.t. } g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, n$$
$$h_j(\boldsymbol{x}) = 0, j = 1, \dots, m$$



  * E.g., find deepest point in the lake, *south of the bridge*

- <u>Applicable but</u>: gradient descent doesn't immediately apply
- Hard-margin SVM: $\underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2$ s.t. $1 - y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) \leq 0$ for $i = 1, \dots, n$
- Method of Lagrange multipliers
  * Transform to unconstrained optimisation – not necessarily for solution
  * Transform primal program to a related dual program, alternate to primal
  * Analyse necessary & sufficient conditions for solutions of both programs

The Lagrangian and duality

- Introduce auxiliary objective function via auxiliary variables

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\nu}) = f(\boldsymbol{x}) + \sum_{i=1}^{n} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{m} \nu_j h_j(\boldsymbol{x})$$

Primal constraints became penalties

  - Called the *Lagrangian* function
  - New $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are called the *Lagrange multipliers* or *dual variables*

- (Old) primal program: $\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}\geq\mathbf{0},\boldsymbol{\nu}} \mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\nu})$
- (New) dual program: $\max_{\boldsymbol{\lambda}\geq\mathbf{0},\boldsymbol{\nu}} \min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\nu})$

May be easier to solve, advantageous

- Duality theory relates primal/dual:
  - Weak duality: dual optimum ≤ primal optimum
  - For convex programs (inc. SVM!) strong duality: optima coincide!

Karush-Kuhn-Tucker Necessary Conditions

- Lagrangian: $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \sum_{i=1}^{n} \lambda_i g_i(\boldsymbol{x}) + \sum_{j=1}^{m} \nu_j h_j(\boldsymbol{x})$
- Necessary conditions for optimality of a primal solution
- Primal feasibility:
  * $g_i(\boldsymbol{x}^*) \leq 0, i = 1, \ldots, n$
  * $h_j(\boldsymbol{x}^*) = 0, j = 1, \ldots, m$

Souped-up version of necessary condition “derivative is zero” in **unconstrained** optimisation.

- Dual feasibility: $\lambda_i^* \geq 0$ for $i = 1, \ldots, n$
- Complementary slackness: $\lambda_i^* g_i(\boldsymbol{x}^*) = 0, i = 1, \ldots, n$

Don’t penalise if constraint satisfied

- Stationarity: $\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) = \boldsymbol{0}$

KKT conditions for hard-margin SVM

The Lagrangian

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \lambda_i \left(y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) - 1\right)$$

KKT conditions:

* Feasibility: $y_i\left((\boldsymbol{w}^*)'\boldsymbol{x}_i + b^*\right) - 1 \geq 0$ for $i = 1, \dots, n$
* Feasibility: $\lambda_i^* \geq 0$ for $i = 1, \dots, n$
* Complementary slackness: $\lambda_i^*\left(y_i\left((\boldsymbol{w}^*)'\boldsymbol{x}_i + b^*\right) - 1\right) = 0$
* Stationarity: $\nabla_{\boldsymbol{w},b}\mathcal{L}(\boldsymbol{w}^*, b^*, \boldsymbol{\lambda}^*) = \boldsymbol{0}$

Let's minimise Lagrangian w.r.t primal variables

- Lagrangian:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \lambda_i (y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) - 1)$$

- Stationarity conditions give us more information:

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{n} \lambda_i y_i = 0$$

New constraint

$$\frac{\partial \mathcal{L}}{\partial w_j} = w_j^{\ *} - \sum_{i=1}^{n} \lambda_i y_i (\boldsymbol{x}_i)_j = 0$$

Eliminates primal variables

- The Lagrangian becomes (with additional constraint, above)

$$\mathcal{L}(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i' \boldsymbol{x}_j$$

Dual program for hard-margin SVM

- Having minimised the Lagrangian with respect to primal variables, now maximising w.r.t dual variables yields the dual program

$$\underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i' \boldsymbol{x}_j$$

$$\text{s.t. } \lambda_i \geq 0 \text{ and } \sum_{i=1}^{n} \lambda_i y_i = 0$$

- Strong duality: Solving dual, solves the primal!!
- Like primal: A so-called *quadratic program* - off-the-shelf software can solve – more later
- Unlike primal:
  * Complexity of solution is $O(n^3)$ instead of $O(d^3)$ – more later
  * Program depends on dot products of data only – more later on kernels!

Making predictions with dual solution

Recovering primal variables

- Recall from stationarity: ${w_j}^* - \sum_{i=1}^{n} \lambda_i y_i (\boldsymbol{x}_i)_j = 0$
- Complementary slackness: b^* can be recovered from dual solution, noting for any example j with $\lambda_i^* > 0$, we have $y_j\left(b^* + \sum_{i=1}^{n} \lambda_i^* y_i \boldsymbol{x}_i' \boldsymbol{x}_j\right) = 1$

Testing: classify new instance $\boldsymbol{x}$ based on sign of

$$s = b^* + \sum_{i=1}^{n} \lambda_i^* y_i \boldsymbol{x}_i' \boldsymbol{x}$$

Soft-margin SVM's dual

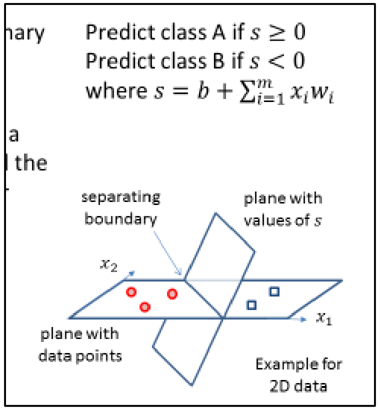
- Training: find $\boldsymbol{\lambda}$ that solves

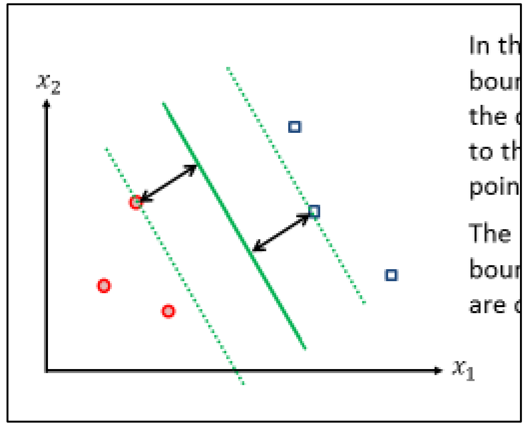
$$\underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i' \boldsymbol{x}_j$$

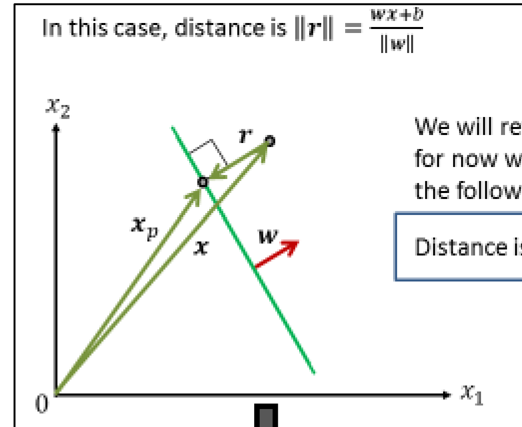
box constraints

s.t. $\boxed{C \geq \lambda_i \geq 0}$ and $\sum_{i=1}^{n} \lambda_i y_i = 0$

- Making predictions: same pattern as in as in hard-margin case







Hard margin SVM objective is a constrained optimisation problem:

$$\underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2$$

s.t. $y_i(\boldsymbol{w}\boldsymbol{x}_i + b) - 1 \geq 0$ for $i = 1, \dots, n$

- The corresponding KKT conditions a
- $y_i(\boldsymbol{w}\boldsymbol{x}_i + b) - 1 \geq 0$ for $i = 1, \dots, n$
- $\lambda_i \geq 0$ for $i = 1, \dots, n$
- $\lambda_i(y_i(\boldsymbol{w}\boldsymbol{x}_i + b) - 1) = 0$
- $\nabla_{\boldsymbol{w},b} L_{KKT}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = 0$ zero gradie to uncons

Hard margin SVM Lagrangian dual problem is

$$\underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j$$

s.t. $\lambda_i \geq 0$ and $\sum_{i=1}^{n} \lambda_i y_i = 0$

- Training: finding $\boldsymbol{\lambda}$ that solve

$$\underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j$$

s.t. $\lambda_i \geq 0$ and $\sum_{i=1}^{n} \lambda_i y_i = 0$

- Making predictions: classify new instance $\boldsymbol{x}$ base sign of

$$s = b + \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i \boldsymbol{x}$$

graphics: stux at pixabay.com (CC0)

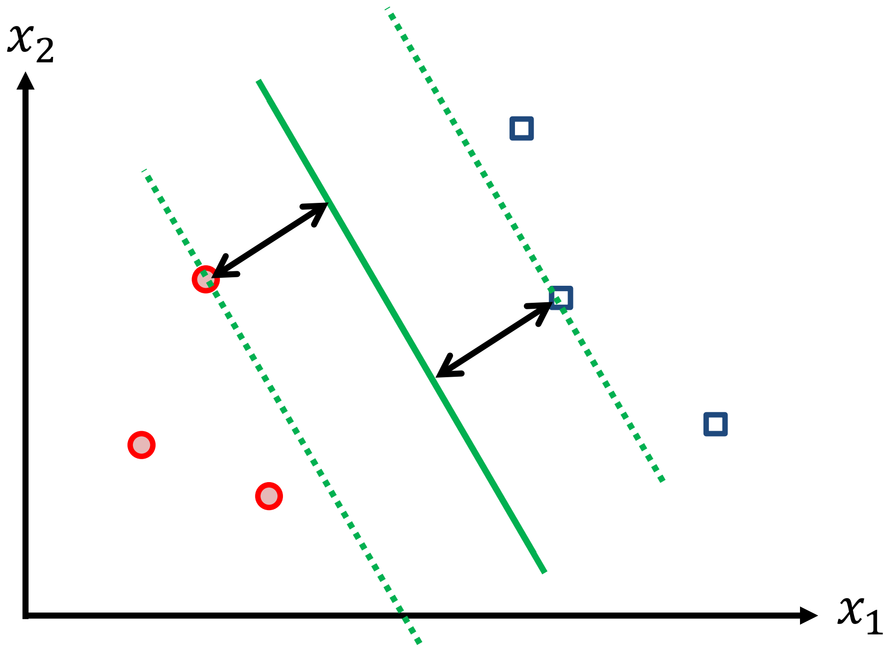
Additional Notes

Complementary slackness, Dot products

- Recall that one of the KKT conditions is *complementary slackness*

$$\lambda_i^*\left(y_i\left((\boldsymbol{w}^*)'\boldsymbol{x}_i + b^*\right) - 1\right) = 0$$

- Remember that $y_i(\boldsymbol{w}'\boldsymbol{x}_i + b) - 1 > 0$ means that $\boldsymbol{x}_i$ is outside the margin



(Likely many) points outside the margin must have $\lambda_i^* = 0$

The points with non-zero λs are *support vectors*

$$\boldsymbol{w}^* = \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i$$

Predictions made by dot products with the s.v.'s

$$s = b^* + \sum_{i=1}^{n} \lambda_i^* y_i \boldsymbol{x}_i' \boldsymbol{x}$$

Training the SVM

- The SVM dual problems are *quadratic programs*. Using standard algorithms this problem can be solved in in $O(n^3)$. Or $O(d^3)$ for the primal.

- This can inefficient; Several specialised solutions proposed

- Solutions mostly decompose training data and break down program into smaller programs that can be solved quickly

- Original SVM training algorithm *chunking* exploits fact that many λs will be zero (sparsity)

- *Sequential minimal optimisation* (SMO) another algorithm an extreme case of chunking. An iterative procedure that analytically optimises randomly chosen pairs of λs per iteration

This lecture

- Soft-margin SVM
  * Intuition and problem formulation
- Forming the dual program
  * Lagrangian multipliers, KKT conditions
  * Weak and strong duality
- Finishing touches
  * Complementary slackness
  * Notes on training
- Workshops Week #6: more neural nets
- Next lecture: kernelisation