UNIVERSITY OF
SAN FRANCISCO
Master of Science
in Analytics

Support Vector Machines

Machine Learning 1

Classification - simple beginnings

- Assumptions
  - Simple start: two classes, two features, linearly separable
  - Goal: find a separating plane
  - Ultimately: *k* classes; *p* features, separating hyperplane
- Hyperplane
  - A "flat affine subspace" of dimension *p* - 1
  - Takes the form:

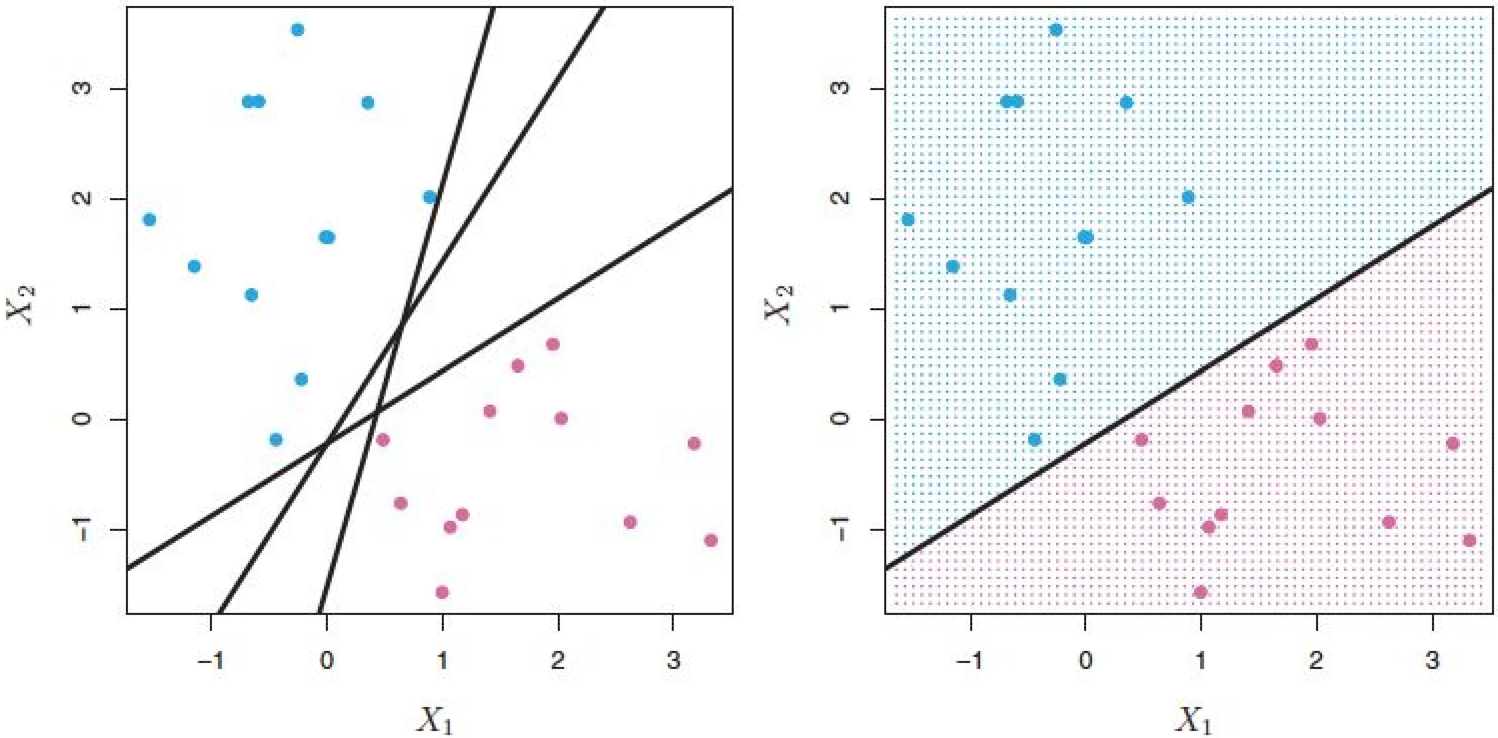
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p = 0$$

- The "[normal] vector"
  - Orthogonal to the hyperplane
  - Represented as:

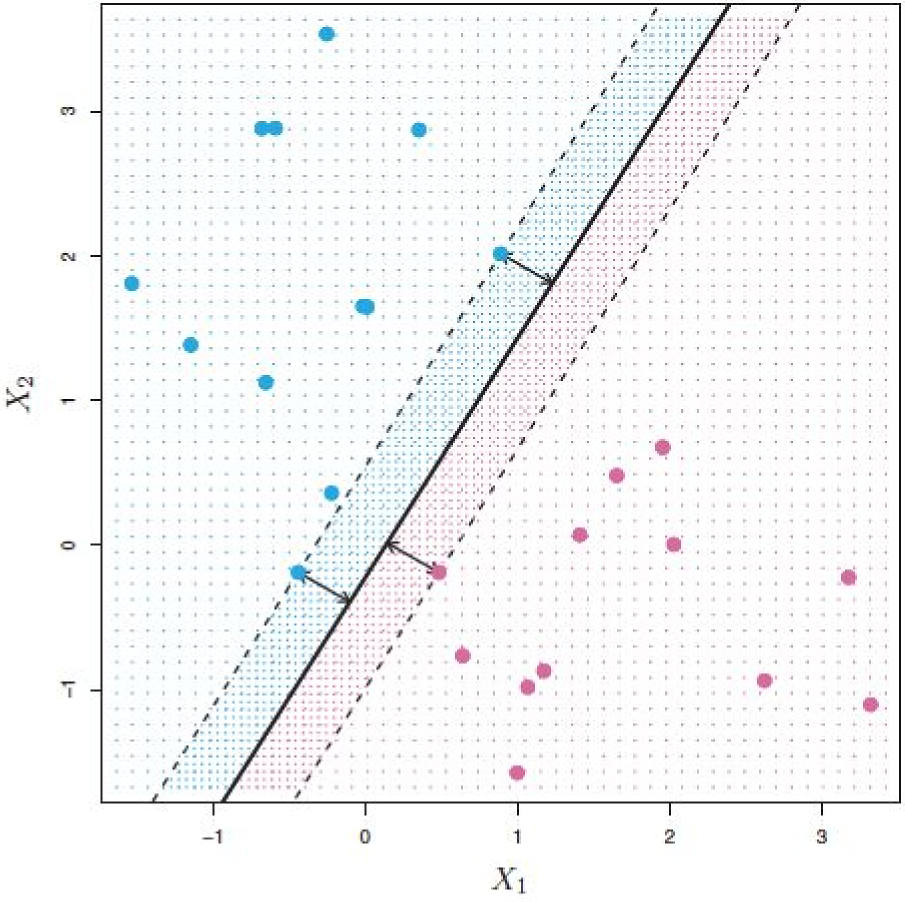
$$\beta_0 = (\beta_1, \beta_2, \ldots, \beta_p)$$

Max margin classifier

- Many (infinite?) possible hyperplanes
- Pick the one which maximizes the gap between classes



Max margin classifier



$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

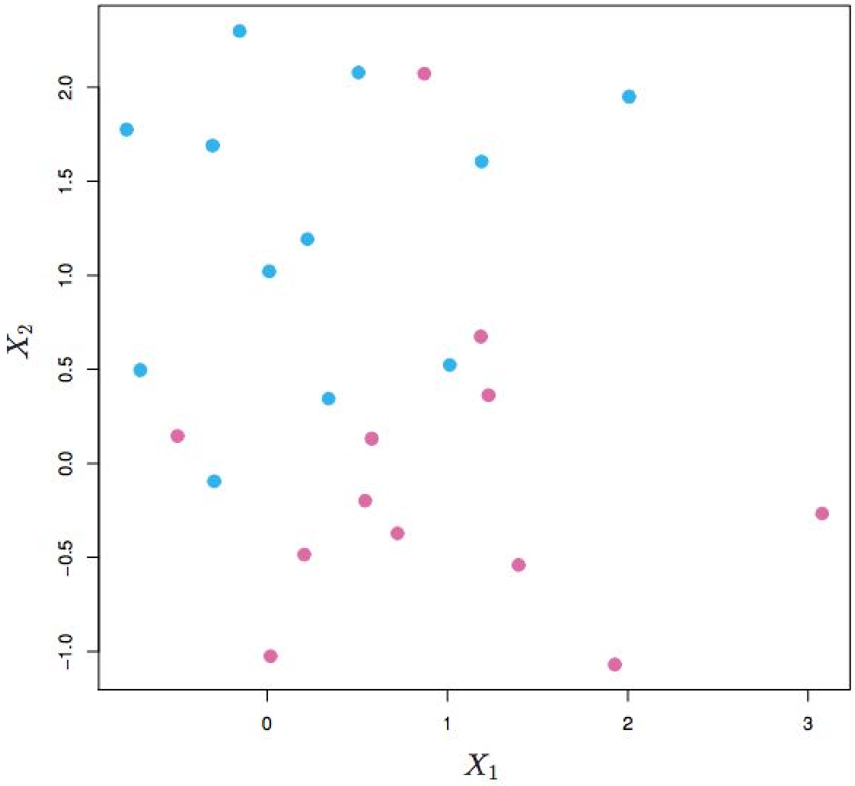
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all } i = 1, \ldots, N.$$

Margin: gap between hyperplane and observations

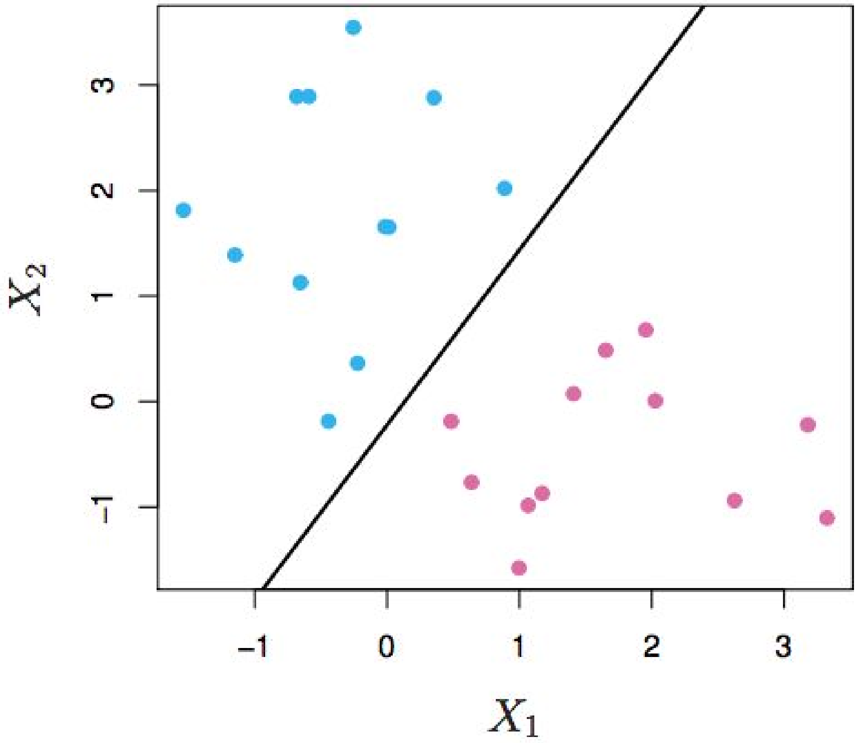
Problem 1

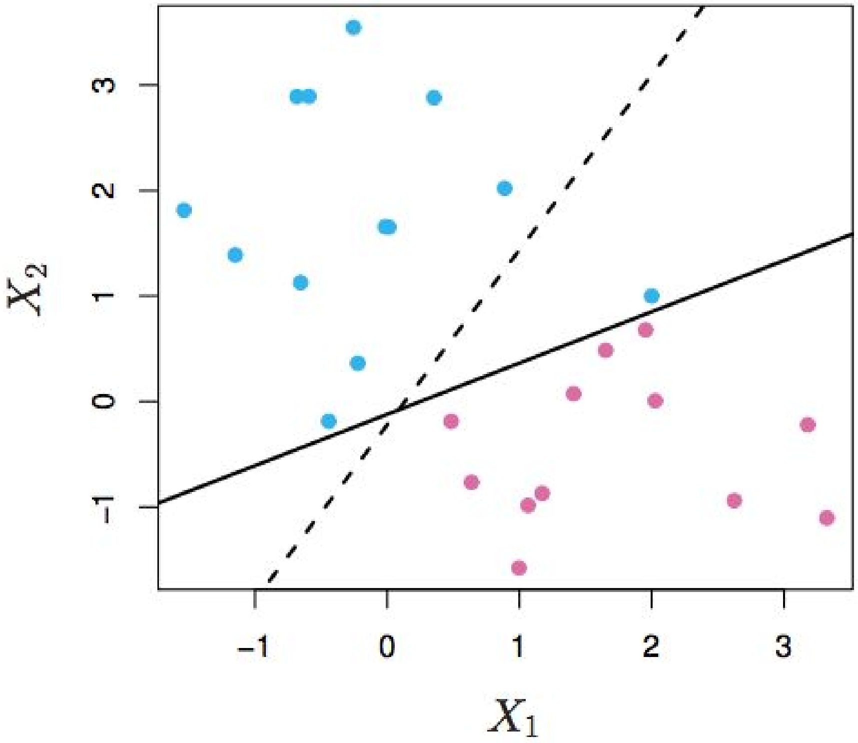


- Assume n > p
- Classes are not always linearly separable

Problem 2

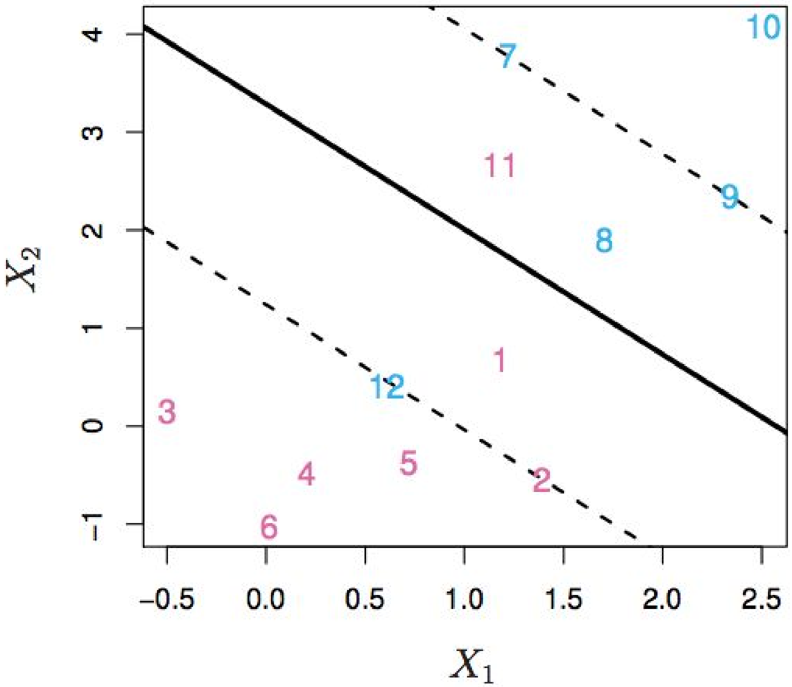
- (Not really a problem)
- Observations can dramatically change the hyperplane





Solution: support vector classifier

- Create a *soft* margin
  - Allows observations to “violate” the margin
  - Allows observations to “violate” the hyperplane
- Set max budget for violations, “C”

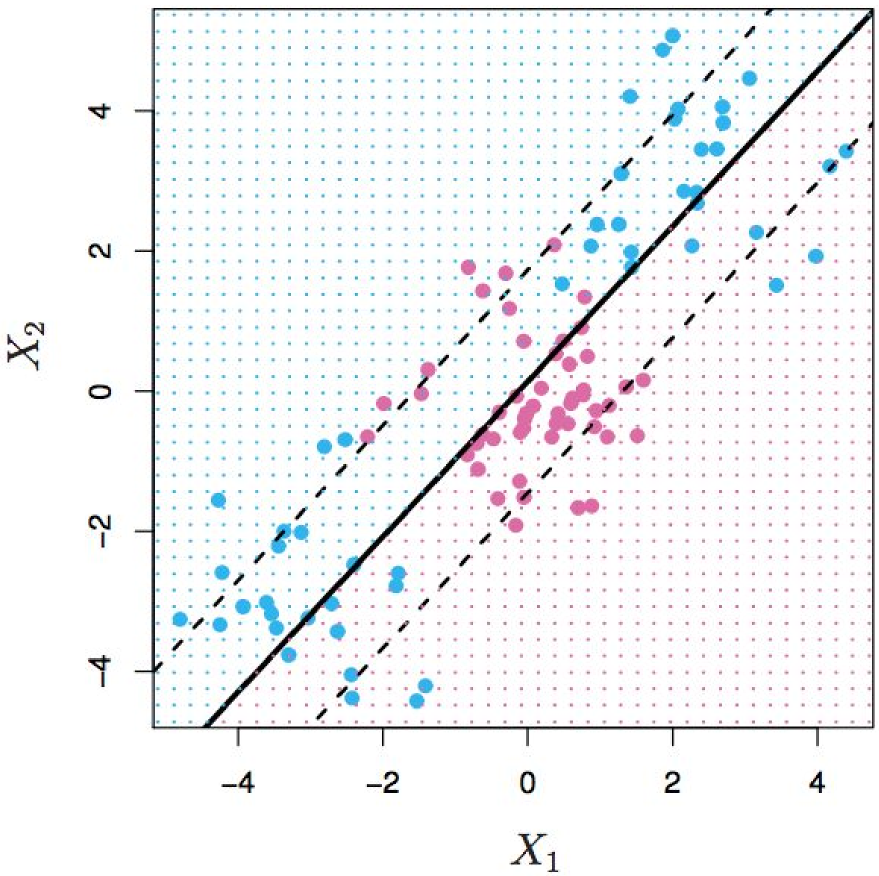


$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M \quad \text{subject to} \ \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C,$$

Problems with SVCs

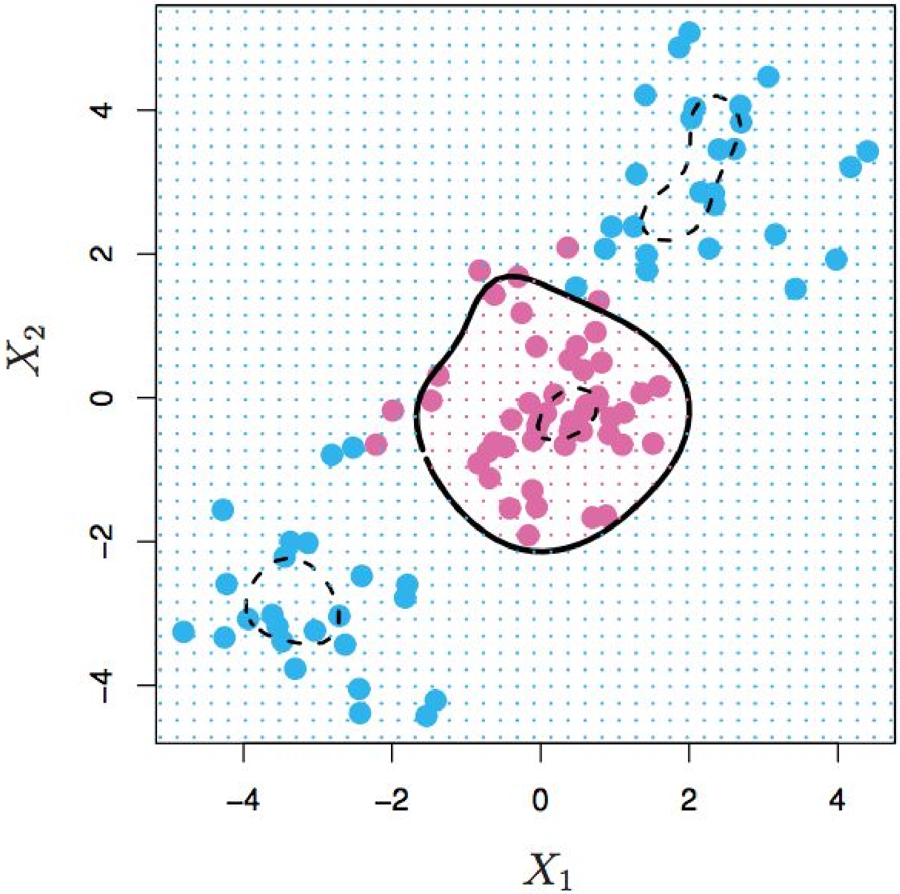


- Sometimes fails
  - In example, data is not linearly separable
  - Does not depend on “C”
- What could work
  - Powers of features X^2, X^3, etc.
  - Goes from *p*-dimensional to *M* > *p* dimensional space
  - SVC in larger space solves the problem in lower-dimensional space

Nonlinearities and kernels

- Polynomials get wild fast
  - Especially true for high-dimensional polynomials
- Inner products
  - Scalar from two (feature) vectors <x_i, $x_{i'}$>
  - Product of the vector magnitudes and the cosine of the angle between them
- Linear support vector classifier —-> $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$
  - Need to estimate parameters $\alpha_1, \ldots, \alpha_n$ and β_0
  - Need *n* choose 2 inner products between all pairs of observations
- Fitting a Support Vector classifier
  - Can be done if by computing inner products between observations
  - Different types of kernels
  - Example: d-dimensional polynomials $K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$

Radial kernel (& example)



$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

Controls variance by squashing down most dimensions

For multiple classes

- Options for $K > 2$ classes
  - OVA (one vs. all): fit K different 2-class SVM classifiers; classify x^* to the class which generates the max score
  - OVO: (one vs. one): Fit all K choose 2 pairwise classifiers; classify x^* to the class which wins the most pairwise competitions
- Which to choose?
  - If K is not large, we use OVO
  - If K is large...

SVM vs. logistic regression

- When classes are (nearly) separable, SVM does better
- When not separable, both behave similarly
  - LR must have *ridge penalty*
  - Need to estimate probabilities? LR is the choice
- SVMs are popular for non-linear boundaries
  - Implemented as kernels
  - Can also apply kernels ot LR or LDA… but don’t

Implementation in python

1) Import data
    a) If necessary, split data into train, test sets
2) Coerce data into: X (data) and Y (targets); test_x

```
# 3) See the documentation for full set of options

from sklearn.svm import SVC

clf = SVC    # Defaults to C=1.0 & kernel='rbf'
clf.fit(X, Y)
test_y = clf.predict(test_x))
```