Support Vector Machines

Rucha Joshi

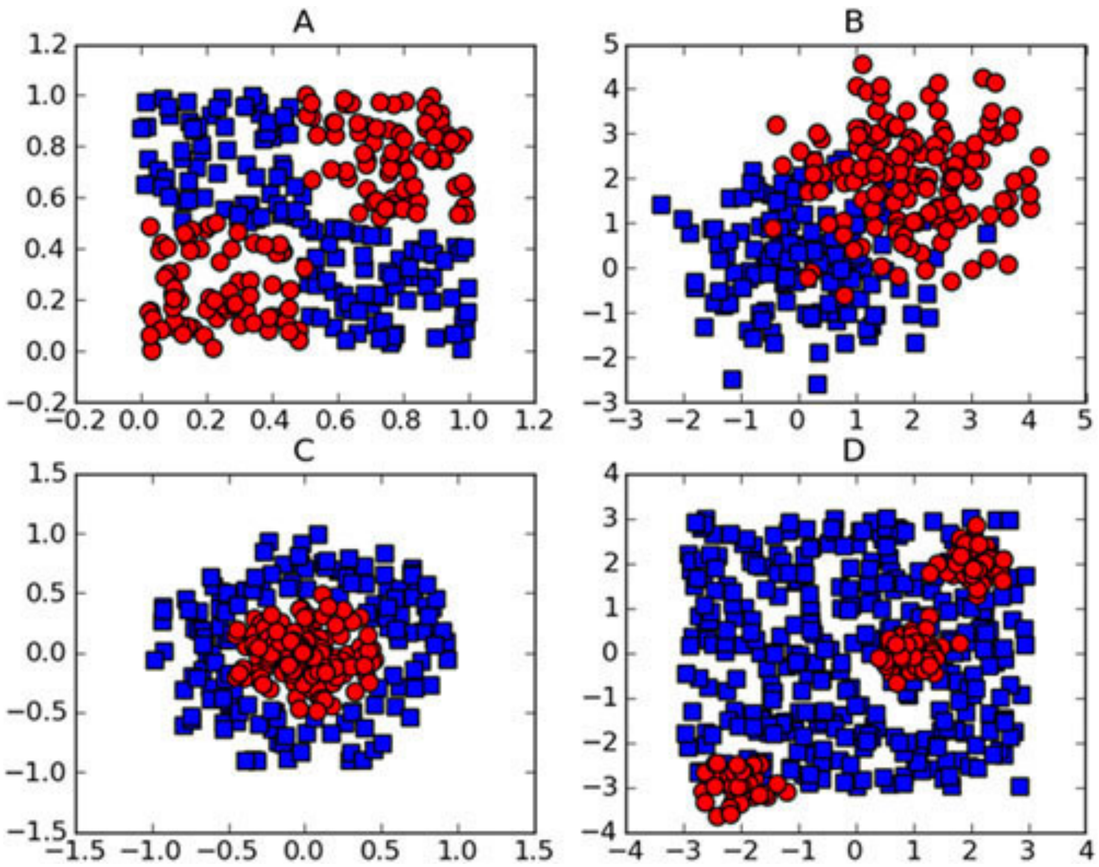
Overview

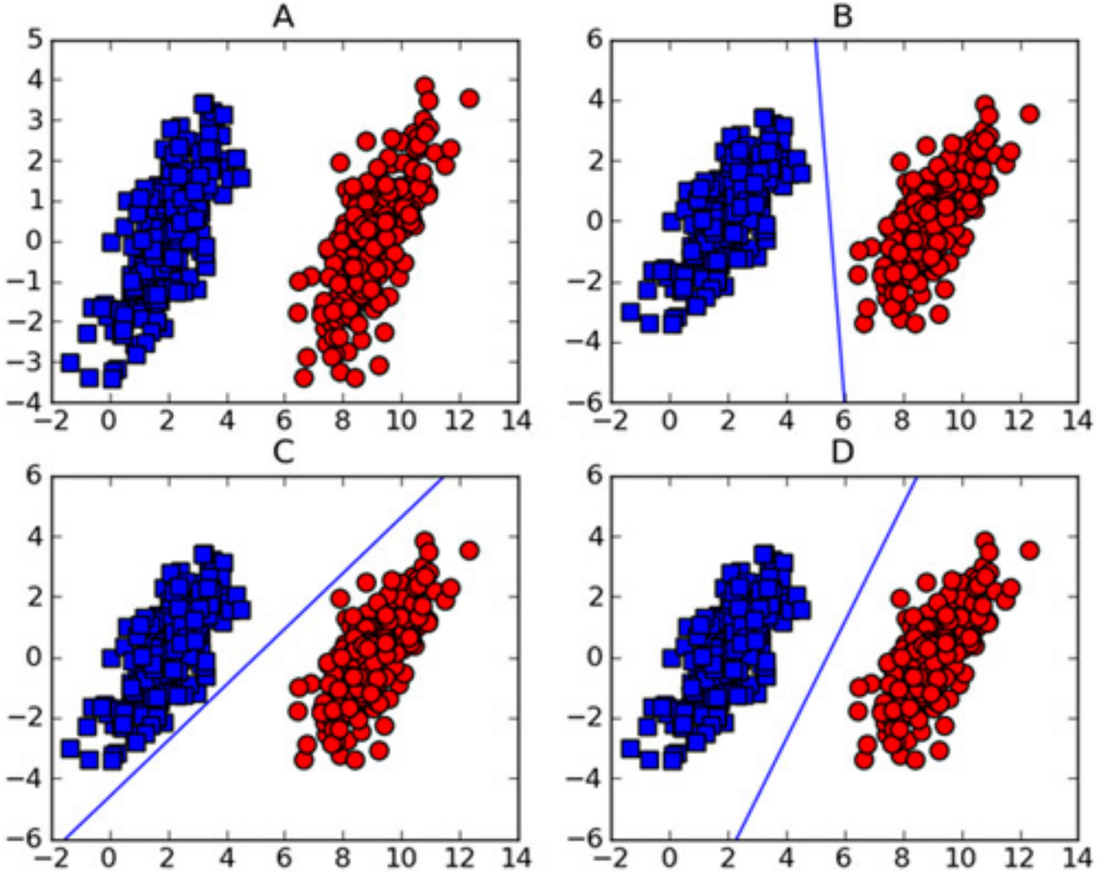
- Introduction to SVM
- Using SMO algorithm for optimization
- Using kernels to ‘transform’ data

Support Vector Machines

- Pros
  - Low generalization error
  - Computationally inexpensive
  - Easy to interpret results
- Cons
  - Sensitive to tuning parameters and kernel choices
  - Natively handles binary classifications only
- Can handle numeric and nominal data

Data separability





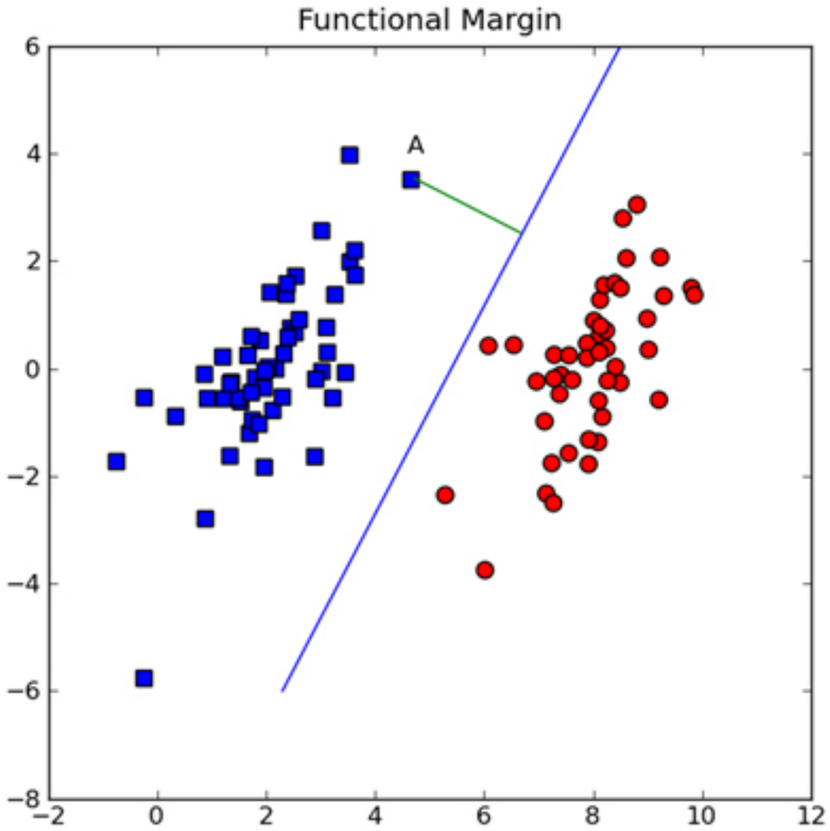
Terminology

- Separating hyperplane - decision boundary
  - e.g. for data of 1024 dimensions - hyperplane of 1023 dimensions
- Support Vectors - points closest to separating hyperplane
- Margin - Support vectors should be as far as possible
  - Goal: greatest margin
- How to optimize?

Finding maximum margin

- Separating hyperplane: $w^T x + b$
- Distance from point A is given by $\frac{|w^T x + b|}{\|w\|}$
- Labels: +1 and -1
- Maximize the margin while finding points with smallest margin

$$arg \max_{w,b} \left\{ \min_{n} (label \,.\, (w^T x + b)) \,.\, \frac{1}{\|w\|} \right\}$$



Finding maximum margin

- Set $(label\,.\,(w^T x + b))$ to be 1 for support vectors
  - Minimize $\|w\|^{-1}$
- Using Lagrange multipliers,

$$\max_{\alpha} \left[\sum_{i=1}^{m} \alpha - \frac{1}{2} \sum_{i,j=1}^{m} label^{(i)} \,.\, label^{(j)} \,.\, \alpha_i \,.\, \alpha_j \langle x^{(i)} \,.\, x^{(j)} \rangle \right]$$, subject to

$\alpha \geq 0$, and $\sum_{i-1}^{m} \alpha_j \,.\, label^{(i)} = 0$

- Assumption: 100% linearly separable data

Finding maximum margin

- Assumption: 100% linearly separable data
- Introduce *slack variables*: allow examples to be on wrong side of decision boundary
- $c \geq \alpha \geq 0$, and $\sum_{i-1}^{m} \alpha_i \,.\, label^{(i)} = 0$
- c: parameter for optimization
- Solve for α to get separating hyperplane

Sequential Minimal Optimization

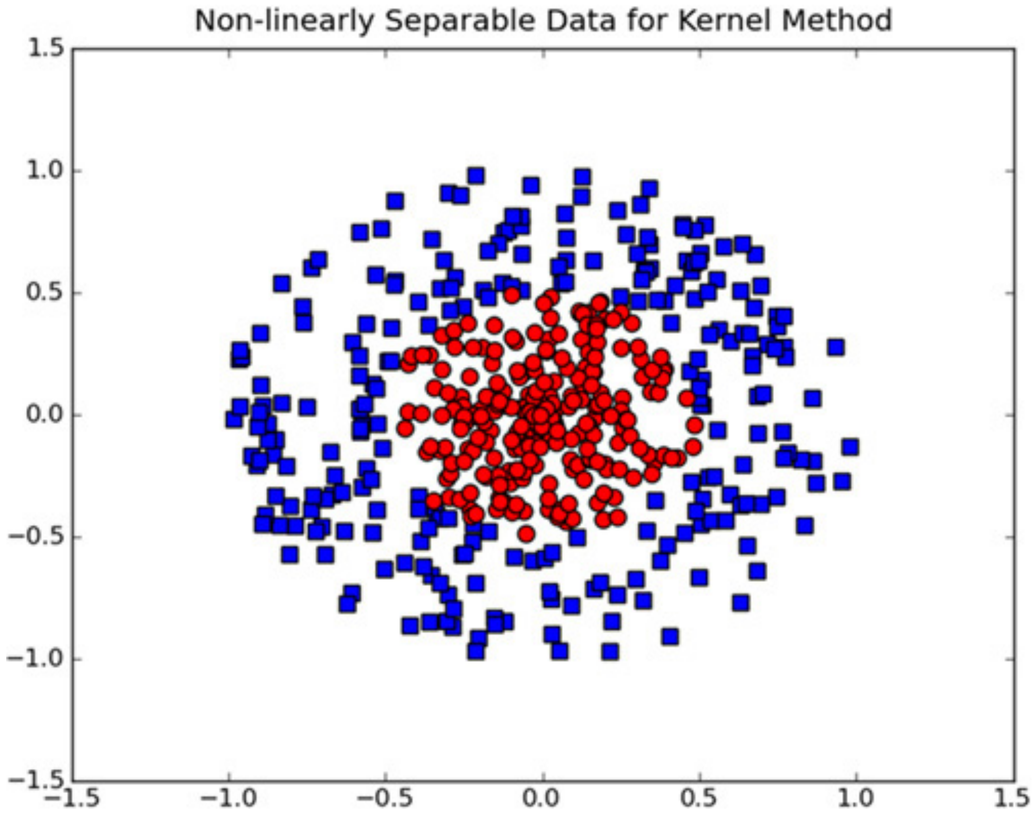
- SMO, instead of quadratic solver (QS optimizes functions subject to linear constraints on the variables)
- Platt’s SMO algorithm, 1996
  - Large optimization problem, broken into small ones
  - Solved sequentially
  - Same answer, reduced time

Sequential Minimal Optimization

- Create an alphas vector filled with 0s
- While the number of iterations < maxIterations:
  - For every data vector in dataset:
    - If data vector can be optimized:
      - Select another data vector at random
      - Optimize the two vectors together
      - If the vectors cannot be optimized, then break
    - If no vectors were optimized, then increment the iteration count

Kernels

- Using kernels for mapping from one feature space to another feature space
- Kernel trick/substation
- In SVM, we need inner products - replaced by kernel functions



Kernels

- Radial bias function: takes a vector and gives a scalar based on the vector's distance, either from 0,0 or other vector
- Gaussian version:
  $k(x, y) = exp\left(\frac{-\|x - y\|^2}{2\sigma^2}\right)$, where σ is a parameter that determines how quickly this falls to 0
- $k(x, y) = (x \,.\, y + 1)^n$

Thank you!