Support Vector Machines (SVMs)

References:
Cristianini & Shawe-Taylor book; Vapnik’s book; and “A Tutorial on Support Vector Machines for Pattern Recognition” by Burges, in Data Mining and Knowledge Discovery 2, 1998 (check citeeseer)

SVMs

• Combine learning and optimization theory
• Exactly solve optimization (as opposed to ANN or Decision Trees)
• Allow use of “kernel trick” to get more features almost for free

Basic Idea: (separable case)

• Which linear threshold hypothesis to use?
• Pick one with biggest margin (gap)
  — Intuitively good, safe against small errors
  — Some supporting theory
  — Empirically works well
• Leads to a quadratic programming problem (Vapnik 95, Cortes and Vapnik 95)

Basic Idea:

• Pick one with biggest margin (gap)
• Leads to a quadratic programming problem (Vapnik 95, Cortes and Vapnik 95)
• Support Vectors are the points with smallest margin (closest to boundary, circled)
• Hypothesis stability: it only changes if support vectors change

Good generalization

• If \( f(x) = \sum w_i x_i + b \) and \( \sum w_i^2 = 1 \) and large margin, then it (expects) to generalize well

\[
\hat{\delta} \left( \frac{R^2}{N \delta^2} \frac{\log(1/\text{error})}{N} \right)
\]

\( R = \) radius of support, \( N = \) # examples, \( \delta = \) margin

\( \text{this is independent of dimension (#features)}! \)

SVM Mathematically

• Find \( w, b, \delta \) such that:
  \[ w \cdot x_i + b \geq \delta \quad \text{when } y_i = +1 \]
  \[ w \cdot x_i + b \leq -\delta \quad \text{when } y_i = -1 \]
  \( \delta \) as big as possible
• Scaling issue - fix by setting \( \delta = 1 \) and finding shortest \( w \):
  \[
  \min_{w, b} ||w||^2 \text{ subject to } y_i (w \cdot x_i + b) \geq 1 \text{ for all examples } (x_i, y_i)
  \]
• This is a quadratic programming problem, packages exist (also SMO algorithm)
More Math (Quickly)

- LaGrange multipliers, dual problem, ..., magic, ...
- The \( w \) vector is a linear combination of support vectors: \( w = \sum_{\text{sup vcts}} a_i x_i \)
- Predict on \( x \) based on \( w \cdot x \geq b \), or equivalently \( \sum_{\text{sup vcts}} a_i (x_i \cdot x) = b \)
- Key idea 1: predictions (and finding \( a_i \)'s) depend only on dot products

Kernel functions

- Don’t need to use dot product, can use any dot-product like function \( K(x,x') \)
- Example: \( K(x,x') = (x \cdot x' + 1)^2 \)
- In 1-dimension: \( K(x,x') = x^2 x'^2 + 2xx + 1 = (x^2, \sqrt{3}x, 1) \cdot (x^2, \sqrt{3}x', 1) \), get quadratic term too (extra feature)
- How can this help?

Kernel trick:

- Kernel functions embed the low dimensional original space into a higher dimensional one like the film of a soap bubble
- Although sample is not linearly separable in the original space, the higher-dimensional embedding might be
- Get embedding and “extra” features for free (almost)
- General trick for any dot-product based algorithm (e.g. Perceptron)

Common Kernels

- Polynomials: \( K(x,x') = (x \cdot x' + 1)^d \)
- Radial Basis function (Gaussian): \( K(x,x') = \exp(-||x - x'||^2 / 2\sigma^2) \)
- Sigmoid: \( K(x,x') = \tanh(a (x \cdot x') - b) \)
- Also special purpose string and graph kernels
Which to use and parameters is a black art, cross validation can help

Gaussian Kernels

- Act like weighted nearest neighbor on support vectors
- Allow very flexible hypothesis space
- Mathematically a dot product in an “infinite dimensional” space
- SVMs unify LTU (linear kernel) and nearest neighbor (Gaussian kernel)
Soft Margin for noise

- Allow margin errors
- \( \varepsilon_i \geq 0 \) measures amount of error on \( x_i \)
- Want to minimize both \( w^T x_i \) and \( \sum \varepsilon_i \)
- Need tradeoff parameter
- Still Quadratic programming, math messier

SVM applications

- Text classification - (Joachims, used outlier removal when \( a_i \) too large, Joachims also author of SVM-light)
- Object detection (e.g. Osuna et.al., for faces)
- Hand written digits - few support vectors
- Bioinformatics - protein homology, micro-array

See Cristianini&Shaw-Taylor book, and www.kernel-machines.org for more info and references

SVM Summary

- Model: very flexible, but must pick many parameters (kernel, kernel parameters, trade-off)
- Data: Numeric (depending on kernel)
- Interpretable? Yes for dot product, pretty pictures for Gaussian kernels
- Missing values? No
- Noise/outliers? Very good
- Irrelevant features? Yes, dot product with abs value penalty
- Comp. efficiency? Poor quadratic in # examples, but exact optimization rather than approximate (like ANN, decision trees)
  Chunking techniques and SMO (iteratively optimize \( a_i \)'s over pairs)