Lecture 9: Support Vector Machines

William Webber (william@williamwebber.com)

COMP90042, 2014, Semester 1, Lecture 8

What we'll learn in this lecture

Support Vector Machines (SVMs)

- a highly robust and effective classifier
- theory of maximum-margin hyperplane
- transforming data into higher dimensional space
- soft-margin for classifier errors
- practicalities of use with text classification

Support Vector Machines (SVM)

Basic concepts of (binary) SVMs:

- Project training data into feature space
- Find the *maximum-margin hyperplane* (MMH) between classes
  - Hyperplane is generalization of line to > 3 dimensions
- MMH completely separates training data into positive and negative classes
- ... and maximizes distance of nearest examples from hyperplane
- These nearest examples are called the *support vectors*

Support vectors and separating hyperplane

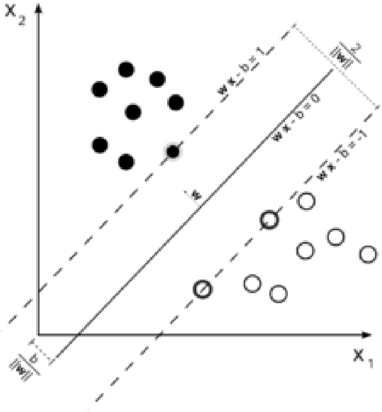

Figure : Maximum margin hyperplane and support vectors (Wikipedia)

Calculating MMH: math

Labelled training examples

$$(y_1, \mathbf{x}_1), \ldots, (y_\ell, \mathbf{x}_\ell), \quad y_i \in \{-1, 1\} \tag{1}$$

separable if exists vector **w** and scalar b such that:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, \ldots, \ell \tag{2}$$

($\cdot$ is dot product). **w** describes angle of hyperplane, being vector perpendicular to it; b ("bias") locates it from origin, relative to **w**. Optimal hyperplane:

$$\mathbf{w}_o \cdot \mathbf{x} + b_0 = 0 \tag{3}$$

separates with maximal margin. Maths[1] shows this is one that minimizes $|\mathbf{w}|$ under constraint (2).

[1] Cortes and Vapnik (1995)

Calculating MMH: implementation

$$\min(|\mathbf{w}|), \quad \text{s.t. } y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, \ldots, \ell \tag{4}$$

- A quadratic programming problem
- Requires $O(n^2)$ space in standard QP implementations
- But all that matters are (candidate) support vectors
- This allows efficient decomposition methods, giving linear space and time
  - E.g. note that SV of full set must be SV in any subset it occurs in
  - So calculate SV for subsets, merge[2]

[2] Cortes and Vapnik (1995). See also Joachims (1998).

Classifying new examples

- Model is $(\mathbf{w}_0, b_0)$
- More maths shows that $\mathbf{w}_0$ expressible as linear combination of support vectors $\mathcal{Z}$:

$$\mathbf{w}_0 = \sum_{\mathbf{z}_i \in \mathcal{Z}} \alpha_i \mathbf{z}_i, \quad \alpha_i > 0 \tag{5}$$

- For unlabelled example $\mathbf{x}$, calculate:

$$\hat{y} = \mathbf{w}_0 \cdot \mathbf{x} + b_0 = \sum_{\mathbf{z}_i \in \mathcal{Z}} \alpha_i \mathbf{z}_i \cdot \mathbf{z} + b_0 \tag{6}$$

- Predict class of $\mathbf{x}$ from sign of y
- $|y|$ gives strength of prediction

Linear separability

- Most problems not linearly separable
- Two (not mutually exclusive) solutions:
  - Project data into higher-dimensional space (more chance of being separable)
  - Allow some training points to fall on wrong side of hyperplane (with penalty)

Mapping to higher dimensional space

- Data points mapped to higher dimensional (feature) space
- E.g. for polynomial space, extend / replace raw features:

$$x_1, \ldots, x_n \quad (n \text{ dimensions}) \tag{7}$$

with:

$$x_1^2, \ldots, x_n^2 \quad (n \text{ dimensions}) \tag{8}$$

plus:

$$x_1 x_2, x_1 x_3, \ldots, x_n x_{n-1} \quad \left(\frac{n(n-1)}{2} \text{ dimensions} \right) \tag{9}$$

- Calculate separating hyperplane in higher-dimensional space

Higher-dimensional space: why?

Mapping to higher-dimensional space

- Makes linearly non-separable problem separable (perhaps)
- Finds important relationships between features
- Remove monotonic assumptions from features
  - In linear space, features must be monotonically related to class (e.g., the greater the score of feature x, the greater evidence for class y)
  - Some features not like this (e.g., weight as predictor of health)
  - (Some) mappings to higher-dimensional space allow for discovery of more complex relations

But how is this even possible? Don't we get an explosion of features?

The kernel trick

- Calculation of SVMs uses dot-products throughout
- For certain classes of projections $\varphi(\mathbf{x})$, there exist a *kernel function* $K(\mathbf{x}, \mathbf{y})$ such that:

$$K(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y}) \tag{10}$$

- For example the kernel function:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d \tag{11}$$

  is equivalent to a mapping into degree d polynomial space.
- Simply replace dot product with $K()$ throughout in computation of SVM
- Then linear hyperplane effectively (and cheaply) calculated on higher-d space

Soft-margin classifiers

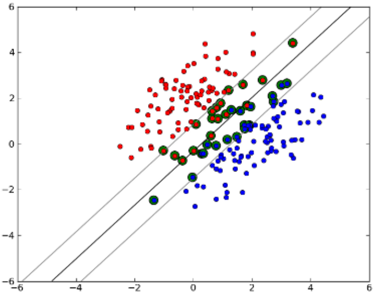

Figure : Soft-margin SVM (from StackOverflow)

- 2nd solution to linear non-separability: allow errors
- i.e. training examples within margin, or on wrong side of hyperplane
- Penalize errors by how "wrong" they are
- Solve minimization problem with error penalty added

Soft-margin: maths

Change hard-margin:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, \ldots, \ell \tag{12}$$

to soft-margin:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \tag{13}$$

where ξ_i is "slack variable" for x_i. Then solve:

$$\underset{\mathbf{w},\xi,b}{\operatorname{argmin}} \left\{ \frac{1}{2}|\mathbf{w}|^2 + C \sum_{i=1}^{n} \xi_i \right\} \tag{14}$$

(where C is our constant *slack parameter*, related to the number of training examples) subject to the constraint in (13)

SVM: practical considerations

- Choice of kernel function is trial-and-error
  - but some insight into data can help (e.g. are features monotonic?)
- Soft-margin classifiers generally used now
- SVM reputedly "robust":
  - Doesn't get confused by correlated features
  - Doesn't overfit
  - Few or no parameters to tune
- So we can "throw features at it" (at least as first pass)

SVM for text classification

- Linear SVM (i.e. no kernel transformations), with soft margins, typically most effective
  - Large feature space
  - Feature monotonicity
- SVM consistently best or near-best text classification effectiveness (over Rocchio, kNN, linear-least square fit, Naive Bayes, MaxEnt, decision trees, etc.)
  - Maxent / logistic regression (see 2nd half of course) and kNN come closest
- Drawback: model is difficult to interpret
  - Based on Support Vectors (marginal documents)
  - Hard to say what features (terms) are strong evidence
  - Non-interpretability common with geometric methods

Looking back and forward

Back

- SVMs (like kNN and Rocchio) based upon geometric model (but partitioning, not similarity)
- Finds maximally separating hyperplane between training data classes; represented by marginal support vectors (training examples)
- Can transform space into higher dimensions and efficiently calculate using kernel trick
- Soft-margin version allows classifier errors, with penalty
- SVM a robust classifier, performs well for text classification

Looking back and forward

Forward

- Later in course, will look at probabilistic classifiers, and further topics in classification
- Next lecture: start on probabilistic models of document similarity

Further reading

- Cortes and Vapnik, "Support-Vector Network", *Machine Learning*, 1995 (Vapnik is the inventor of SVMs; this paper gives a readable introduction to the theory, and then describe soft-margin hyperplane).
- Joachims, "Making Large-Scale SVM Learning Practical", 1998 (describes implementation of decomposition method to optimize calculation of SVMs).
- Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", 1998 (compares SVM with Naive Bayes, Rocchio, C4.5, and kNN)
- Lewis, Yang, Rose, and Li, "RCV1: A New Benchmark Collection for Text Categorization Research", 2004 (compares SVM with kNN and Rocchio)