SVM: Linear method with feature engineering vs kernel method

Hung-Hsuan Chen

Many are taken from Prof. C.-J. Lin's slides

Support vector classification

- Data point i: $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$
- Class label of i: y_i
  - Two classes
  - Class 1: $y_i = 1$
  - Class 2: $y_i = -1$
- Find a hyperplane to separate the data points with maximal margin

Linear SVM vs kernel SVM

- Linear SVM

$$\min_{\mathbf{w},b}\left(\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i\right)$$

Subject to

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i$$

$$\text{and } \xi_i > 0 \ \forall i$$

  - Linear classifier

- Kernel SVM

$$\min_{\mathbf{w},b}\left(\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i\right)$$

Subject to

$$y_i\left(\mathbf{w}^T\phi(\mathbf{x}_i) + b\right) \geq 1 - \xi_i$$

$$\text{and } \xi_i > 0 \ \forall i$$

  - Non-linear classifier

Pros and cons of linear and kernel SVM

- Linear SVM
  - Faster in training and testing
  - If the dataset is non-linearly separable, linear SVM may not perform well
- Kernel SVM
  - Could be much slower in training and testing
  - Can fit any curve
    - RBF kernel
- Test accuracy: kernel > linear
- Time: kernel ≫ linear
- Speed is the reason to choose linear (especially when the data is large)

Solving non-linear SVM : poly2 vs kernel classification

- Non-linear SVM can be solved in two ways
  - Kernel method
    - $\mathbf{x}_i$? $\phi(\mathbf{x}_i)$
      - Little control on the mapping function ϕ (and hence the high dimensional features $\phi(\mathbf{x}_i)$
  - Poly2: linear method + feature engineering
    - Explicitly generating $\phi(\mathbf{x}_i)$
      - Full control on the training features

Popular kernels

- Linear kernel (i.e., linear SVM)

  $K(\mathbf{x}_i, \mathbf{x}_t) = \mathbf{x}_i^T \mathbf{x}_t = \langle \mathbf{x}_i, \mathbf{x}_t \rangle$
- Polynomial kernel

  $K(\mathbf{x}_i, \mathbf{x}_t) = \left(\langle \mathbf{x}_i, \mathbf{x}_t \rangle + r \right)^d, \; r > 0$
- Gaussian (RBF) kernel

  $K(\mathbf{x}_i, \mathbf{x}_t) = \exp\left(-\gamma \left\| \mathbf{x}_i - \mathbf{x}_t \right\|^2 \right)$
- The dimension of $K(\mathbf{x}_i, \mathbf{x}_t)$ could be ***infinity*** (e.g., RBF kernel), but the dimensions of $\mathbf{x}_i$ and $\mathbf{x}_t$ are finite

Poly-2 vs kernel, when data is sparse

- When data is large and sparse
  - Accuarcy by linear SVM + poly2 is as good as kernel
  - Training and testing time of linear SVM + poly2 is much faster

Example: document classification

- Using bag-of-words model to predict the labels of documents
  - Examples of labels: emotion (happy, angry, sad, etc.), author (J. K. Rowling, Dan Brown, etc.), genre (e.g., journal, poetry, ghost story, etc.), …
- Bag-of-words
  - Every English word corresponds to a feature
  - Example
    - Original sentences
      1. John likes to watch movies. Mary likes movies too.
      2. John also likes to watch football games.
    - Bag-of-words representation of the two sentences
      - ["John", "likes", "to", "watch", "movies", "Mary", "too", "also", "football", "games"]
      1. [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
      2. [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]
- Features created by the bag-of-words are usually **large** and **sparse**
  - Large: kernel SVM is much slower than linear SVM
  - Sparse: high dimensional features are likely to be zero (i.e., unhelpful), sotest accuracy of linear SVM + poly2 is as good as kernel

Linear SVM + poly2 vs kernel (training time & test accuracy)

Data set	Linear Time	Linear Accuracy	RBF Kernel Time	RBF Kernel Accuracy
MNIST38	0.1	96.82	38.1	99.70
ijcnn1	1.6	91.81	26.8	98.69
covtype	1.4	76.37	46,695.8	96.11
news20	1.1	96.95	383.2	96.90
real-sim	0.3	97.44	938.3	97.82
yahoo-japan	3.1	92.63	20,955.2	93.31
webspam	25.7	93.35	15,681.8	99.26

Large and sparse datasets

Prediction cost

Assuming **x** has *d* elements (features)

- Kernel method (polynomial kernel of degree-2)
  - Prediction:
    - $\sum_{i=1}^{\ell} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$
  - If $K(\mathbf{x}_i, \mathbf{x})$ takes $O(d)$
    - Predicting one test instance takes $O(\ell d)$
- Poly2: linear method + feature engineering (degree-2 polynomial mapping)
  - Prediction
    - $\mathbf{w}^T \phi(\mathbf{x}) + b$
  - $\phi(\mathbf{x})$ contains $O(d^2)$ elements
    - Predicting one test instance takes $O(d^2)$
- Prediction cost: $O(\ell d)$ vs $O(d^2)$
  - A similar difference occurs for training

Cases where Linear SVM + poly-2 feature engineering is probably faster

- If # training instances >> # features
  - ℓ (# of support vectors) is likely to larger than d (# features)
- If # features is large, but data is sparse
  - $O(d^2) \approx O(d)$
- Roughly
  - Test accuracy: kernel $\geq$ poly2
  - Cost: kernel $\gg$ poly2
- Speed is the reason to choose poly2

Test accuracy and training time

Data set	Degree-2 Polynomial Training time (s) LIBLINEAR	LIBSVM	Accuracy	Accuracy diff. Linear	RBF
a9a	1.6	89.8	85.06	0.07	0.02
real-sim	59.8	1,220.5	98.00	0.49	0.10
ijcnn1	10.7	64.2	97.84	5.63	−0.85
MNIST38	8.6	18.4	99.29	2.47	−0.40
covtype	5,211.9	NA	80.09	3.74	−15.98
webspam	3,228.1	NA	98.44	5.29	−0.76

Generate $\phi(\boldsymbol{x})$ explicitly — Kernel trick

Generate $\phi(\boldsymbol{x})$ explicitly: faster than kernel, better accuracy than linear (no poly-2 mapping), sometimes competitive to RBF kernel

Dependency parsing

	Kernel		Linear	
	RBF	Poly-2	Linear	Poly-2
Training time	3h34m53s	3h21m51s	3m36s	3m43s
Parsing speed	0.7x	1x	1652x	103x
UAS	89.92	91.67	89.11	91.71
LAS	88.55	90.60	88.07	90.71

- Linear SVM with feature engineering is fast in both training and testing, while maintain good accuracy
- Dependency parsing is an NLP task
  - UAS: unlabeled attachment score
  - LAS: labeled attachment score

Discussion

- Linear versus kernel is an important issue in practice.
  - It's takes enormous computation resource to apply kernel SVM on large dataset (e.g., web-scale dataset)
  - You must decide when to use which

Quiz

- If number of instances is much larger than number of features
  - Using polynomial kernel is much more efficient than using linear SVM + poly2 kernel (true or false)