Transductive learning

LING 572
Advanced Statistical Methods for NLP
February 20, 2020

Based on F Xia '18

Reading

- Thorsten Joachims, 1999. “Transductive Inference for Text Classification using SVMs”, in Proceedings of ICML.

Induction vs. Transduction

- Inductive learning: induce a decision function that works well for all the possible examples.

- Transductive learning: find a decision function that works well for the given test examples.

Transductive learning

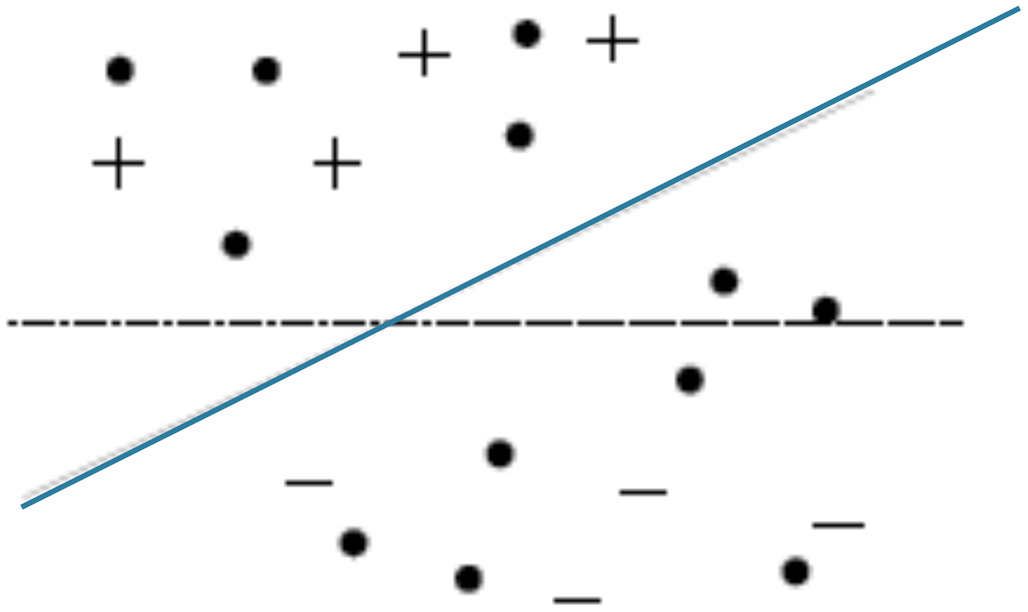
- Setting:
  - Training examples: $\{ (x_i, y_i) \mid i=1, \ldots, n\}$
  - Test examples: $\{ x_i^* \mid i=1, \ldots, k \}$
  - Quite often, n is much smaller than k.

- Training stage:
  - Use both training and test examples
  - Goal: choose a function so that the expected number of erroneous predictions on the test examples is minimized.

Advantage of transductive learning

- The training and test data split hypothesis space H into a finite number of equivalence classes.
  - Two functions belong to the same class if they classify training and test data in the same way.
  - Finding a function in the possibly infinite set H
    - ➔ finding one of finitely many equivalence classes

- We can study the location of the test examples when defining the structure.



+, -: training examples
Dot: test examples

Dashed line: the solution of inductive SVM
Solid line: the solution of transductive SVM

Training: Choose $\vec{w}$ and b

and $y_1^*, y_2^*, \ldots, y_k^*$

Mimimizes $||w||^2$ subject to the constraints

$$y_i(< \vec{w}, \vec{x_i} > +b) \geq 1 \text{ for every } (\vec{x_i}, y_i)$$

and

$$y_i^*(< \vec{w}, \vec{x_i}^* > +b) \geq 1 \text{ for every } (\vec{x_i}^*, y_i^*)$$

Find the hyperplane that separates both training and test examples with maximum margin

Soft margin

$$\textit{Minimize over } (y_1^*, ..., y_n^*, \vec{w}, b, \xi_1, ..., \xi_n, \xi_1^*, ..., \xi_k^*):$$

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n}\xi_i + C^*\sum_{j=0}^{k}\xi_j^*$$

$$\textit{subject to:} \quad \forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i + b] \geq 1 - \xi_i$$

$$\forall_{j=1}^{k} : y_j^*[\vec{w}\cdot\vec{x}_j^* + b] \geq 1 - \xi_j^*$$

$$\forall_{i=1}^{n} : \xi_i > 0$$

$$\forall_{j=1}^{k} : \xi_j^* > 0$$

Solving the optimization problem

- When the number of test instances is small, one can try all possible assignments of the test instances to the two classes.
- When the number of test instances is big, this approach is too expensive.
- Joachims proposed a greedy algorithm.

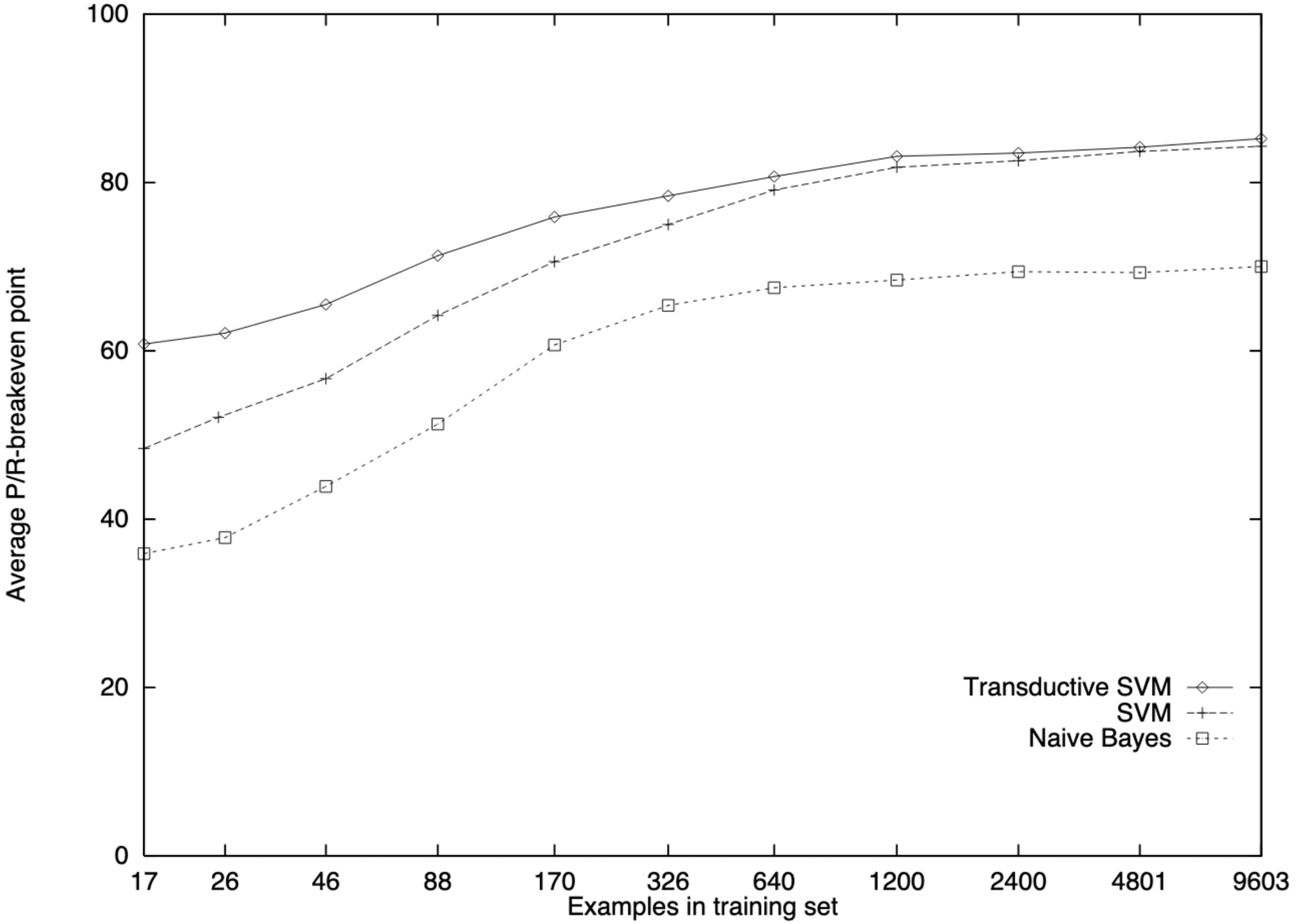
A greedy algorithm

- Train an inductive SVM on the training data
- Use the classifier to classify test instances.
- Improve the solution by switching the labels of test examples so that the objective function decreases.

Experiments

	Bayes	SVM	TSVM
earn	78.8	91.3	95.4
acq	57.4	67.8	76.6
money-fx	43.9	41.3	60.0
grain	40.1	56.2	68.5
crude	24.8	40.9	83.6
trade	22.1	29.5	34.0
interest	24.5	35.6	50.8
ship	33.2	32.5	46.3
wheat	19.5	47.9	54.4
corn	14.5	41.3	43.7
average	35.9	48.4	60.8

Precision/Recall breakeven point for 10 categories,
Using 17 training and 3299 test examples


Average P/R-breakeven point
100
80
60
40
20
0
17
26
46
88
170
326
640
1200
2400
4801
9603
Examples in training set
Transductive SVM
SVM
Naive Bayes

Summary

- Transductive learning aims to find a good decision function for a given test set.
  - Reduce the learning problem into finding one of finitely many equivalence classes.
  - Test instances can help define the structure of the data.

- Transductive SVM:
  - Goal: find a hyperplane that separates both training and test instances with maximum margin.
  - Getting the optimal solution is too expensive; one can use a greedy algorithm.
  - Experiments show that transductive SVM works well when there is a small number of training instances and a large number of test instances.

Additional slides

Algorithm TSVM:

Input: – training examples $(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$
– test examples $\vec{x}_1^*, ..., \vec{x}_k^*$
Parameters: – C,C^*: parameters from OP(2)
– num_+: number of test examples to be assigned to class +
Output: – predicted labels of the test examples $y_1^*, ..., y_k^*$

$(\vec{w}, b, \vec{\xi}, \_) := solve\_svm\_qp([(\vec{x}_1, y_1)...(\vec{x}_n, y_n)], [], C, 0, 0)$;

Classify the test examples using $< \vec{w}, b >$. The num_+ test examples with the highest value of $\vec{w} * \vec{x}_j^* + b$ are assigned to the class + $(y_j^* := 1)$; the remaining test examples are assigned to class − $(y_j^* := -1)$.

$C_-^* := 10^{-5}$; // some small number
$C_+^* := 10^{-5} * \frac{num_+}{k - num_+}$;

```
while((C*_- < C*) || (C*_+ < C*)){                                          // Loop 1
    (w⃗, b, ξ⃗, ξ⃗*) := solve_svm_qp([(x⃗_1, y_1)...(x⃗_n, y_n)], [(x⃗*_1, y*_1)...(x⃗*_k, y*_k)], C, C*_-, C*_+);
    while(∃m,l : (y*_m * y*_l < 0)&(ξ*_m > 0)&(ξ*_l > 0)&(ξ*_m + ξ*_l > 2)) {   // Loop 2
        y*_m := −y*_m;                    // take a positive and a negative test
        y*_l := −y*_l;                    // example, switch their labels, and retrain
        (w⃗, b, ξ⃗, ξ⃗*) := solve_svm_qp([(x⃗_1, y_1)...(x⃗_n, y_n)], [(x⃗*_1, y*_1)...(x⃗*_k, y*_k)], C, C*_-, C*_+);
    }
    C*_- := min(C*_- * 2, C*);
    C*_+ := min(C*_+ * 2, C*);
}
return(y*_1, ..., y*_k);
```