CS M146 - Week 6

Xinzhu Bei

xzbei@cs.ucla.edu

February 16, 2018

Overview

- Midterm Questions
- Bag of words Feature, dictionary in python
- Confusion matrix, performance estimation
- Cost parameter in SVM
- sklearn.svm.SVC and sklearn.svm.LinearSVC

How to read in text file in python

Comparison

```
with open(somefileName,'r') as somefile:
    for line in somefile:
        print line
        # ...more code
```

v.s.

```
somefile = open(somefileName,'r')
try:
    for line in somefile:
        print line
        # ...more code
finally:
    somefile.close()
```

Bag Of Words

In practice, the Bag-of-words model is mainly used as a tool of feature generation. After transforming the text into a "bag of words", we can calculate various measures to characterize the text.

- example in nlp:
  (1) John likes to watch movies. Mary likes movies too.
  (2) John also likes to watch football games.
  ["John","likes","to","watch","movies","Mary","too","also","football","
  ]
  (1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
  (2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]
- An example in computer vision (page 6-9)

Python Dictionary

```
>>> food = {"ham" : "yes", "egg" : "yes", "spam" : "no" }
>>> food
{'egg': 'yes', 'ham': 'yes', 'spam': 'no'}
>>> food["spam"] = "yes"
>>> food
{'egg': 'yes', 'ham': 'yes', 'spam': 'yes'}
```

k in d: True, if a key k exists in the dictionary d
k not in d: True, if a key k doesn't exist in the dictionary d

Before sklearn.metrics: Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. Example:

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

Let's now define the most basic terms, which are whole numbers (not rates):

- **true positives** (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives** (TN): We predicted no, and they don't have the disease.
- **false positives** (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives** (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

Before sklearn.metrics: Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- **Accuracy**: Overall, how often is the classifier correct?

$$(TP + TN)/total = (100 + 50)/165 = 0.91$$

- **Misclassification Rate**: Overall, how often is it wrong?

$$(FP + FN)/total = (10 + 5)/165 = 0.09$$

- **True Positive Rate**(sensitivity, recall): When it's actually yes, how often does it predict yes?

$$TP/(actualyes) = 100/105 = 0.95$$

- **False Positive Rate**: When it's actually no, how often does it predict yes?

$$FP/(actualno) = 10/60 = 0.17$$

Before sklearn.metrics: Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- **False Positive Rate**: When it's actually no, how often does it predict yes?

$$FP/(actualno) = 10/60 = 0.17$$

- **Specificity**: When it's actually no, how often does it predict no?(equivalent to 1 minus False Positive Rate)

$$TN/(actualno) = 50/60 = 0.83$$

- **Precision**: When it predicts yes, how often is it correct?

$$TP/(predictedyes) = 100/110 = 0.91$$

metrics - accuracy, F1-Score

- **Precision**= $TP/(predictedyes)$
- **Recall**= $TP/(actualyes)$

The F1 score can be interpreted as harmonic mean[a] of precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

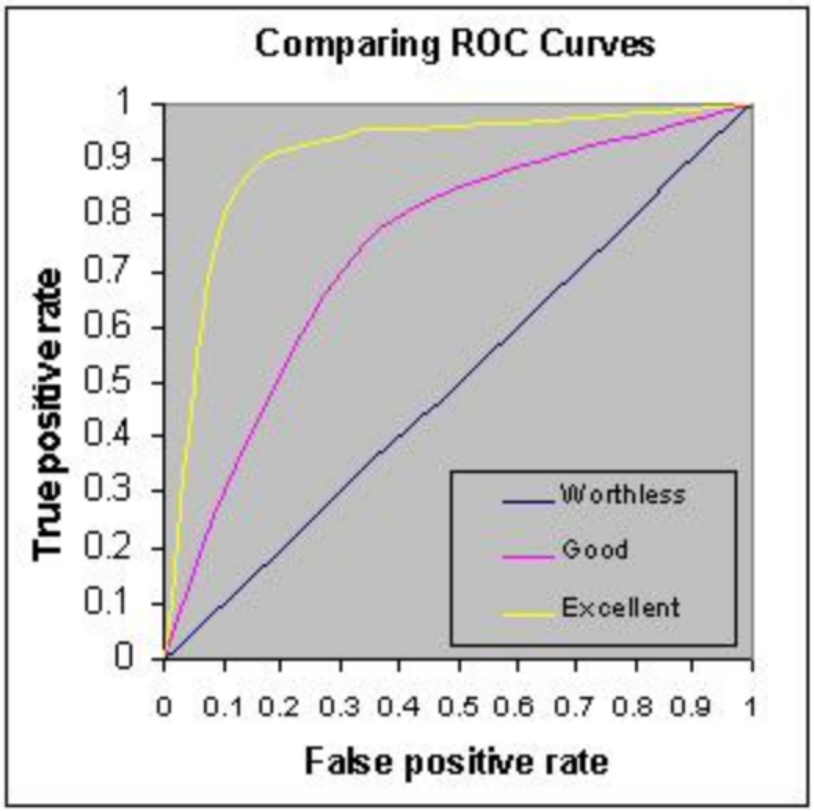
$$F_1 = 2 \cdot \frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

```
metrics.f1_score (y_true, y_label)
```

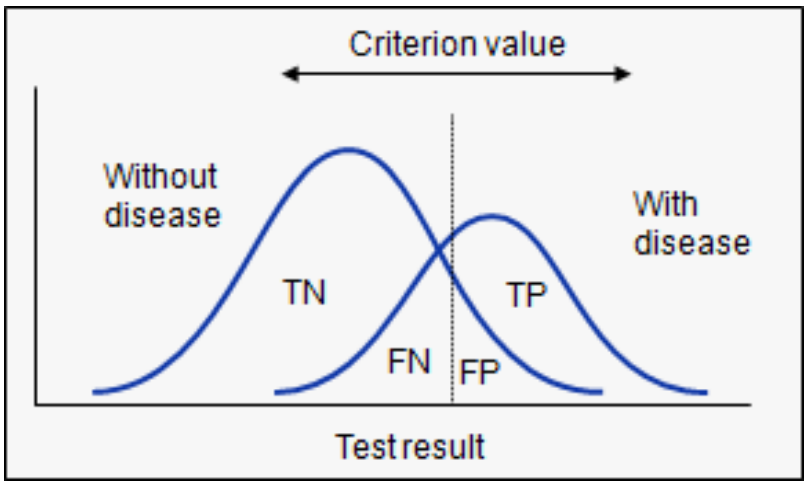
[a] harmonic mean: $H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}}$

sklearn.metrics.roc_auc_score

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings
Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.



The false positive rate = $\frac{FP}{ActualNo} = \frac{FP}{FP+TN}$
(The less the better)
The true positive rate (recall, sensitivity) = $\frac{TP}{ActualYes} = \frac{TP}{TP+FN}$
(The more the better)



The Influence of C in SVM

- **hard-margin SVM**

$$\min_{w,b} \frac{1}{2}||\mathbf{w}||_2^2 \text{ s.t. } \mathbf{y}_i(\mathbf{w}^T x_i + b) \geq 1$$

- Try to find a separating hyper-plane that has maximum distance from either classes' data points, such that all the data points in each side of the hyper-plane should be of the same class

- **"soft-margin" SVM**

$$\min_{\mathbf{w},\xi,b} \left\{ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i \right\} \text{ s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

- C is a hyper-parameter that controls how much we penalize our use of slack variables.

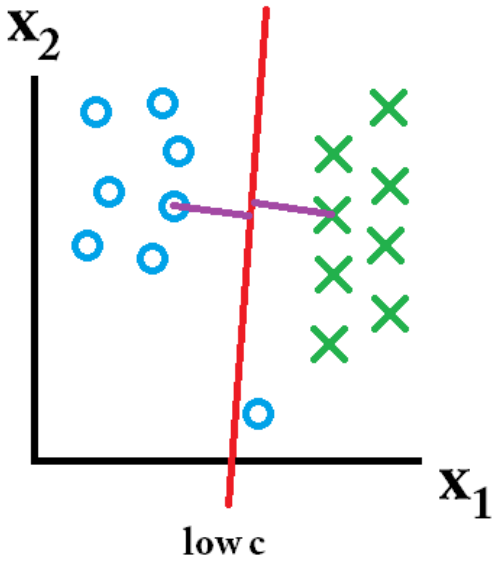
The Influence of C in SVM

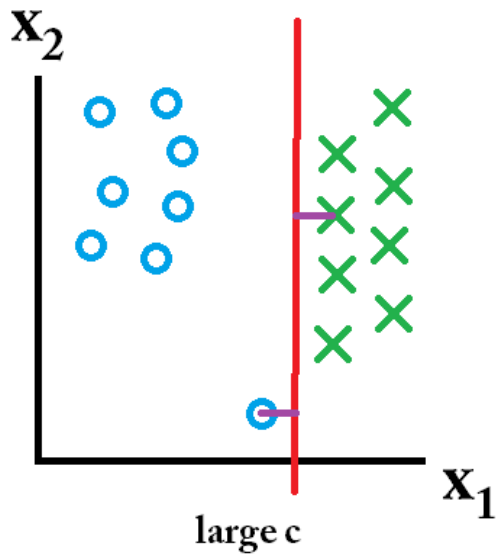
$$\min_{w,b} \frac{1}{2}||\mathbf{w}||_2^2 \text{ s.t. } \mathbf{y}_i(\mathbf{w}^T x_i + b) \geq 1$$

$$\min_{\mathbf{w},\xi,b} \left\{ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \xi_i \right\} \text{ s.t. } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

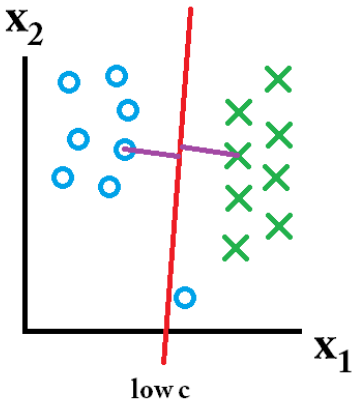
- If we let $C \to 0$ then we don't penalize slack variables at all.
- By increasing C we penalize our slack variables more and more.
- If $C \to \infty$ the soft margin svm. turns out to be hard margin svm

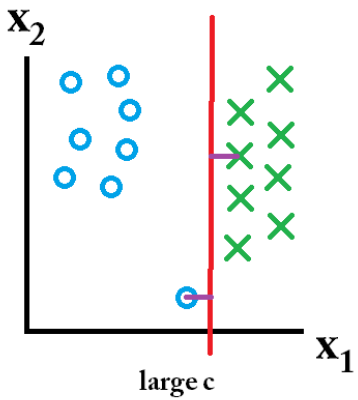
The Influence of C in SVM

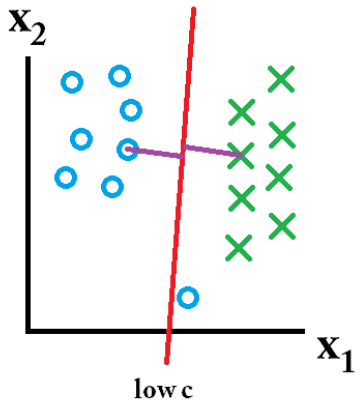


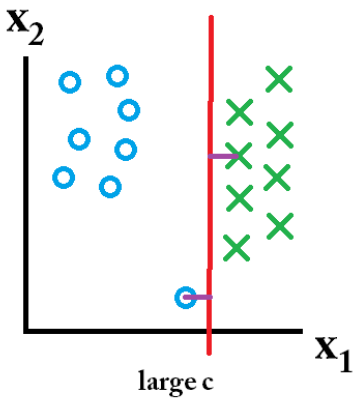


Training









Testing: which one is better? Depends on dataset.

sklearn.svm.SVC & sklearn.svm.LinearSVC

SVC
LinearSVC

Kernel Tricks

A symmetric function $K(x, y)$ can be expressed as an inner product

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

for some ϕ if and only if $K(x, y)$ is positive semidefinite, i.e.

$$\begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) & \cdots \\ K(x_2, x_1) & K(x_2, x_2) & \cdots \\ & \cdots & \end{bmatrix} \tag{1}$$

is psd for any collection $\{x_1, \cdots, x_n\}$

Computational Learning Theory

- Set up:
  - Instance Space: X, the set of examples
  - Concept Space: C, the set of possible target functions: $f \in \mathcal{C}$ is the hidden target function
  - Hypothesis Space: H, the set of possible hypotheses. This is the set that the learning algorithm explores
  - What we want: A hypothesis $h \in H$ such that $h(x) = f(x)$
- Given a distribution D over examples, the **error of a hypothesis** h with respect to a target concept f is

$$err_D(h) = Pr_{x \sim D}[h(x) \neq h(f)]$$

- For a target concept f, the **empirical error** of a hypothesis h is defined for a training set S as the fraction of examples x in S for which the functions f and h disagree, denoted by $err_S(h)$.
- Overfitting: When the empirical error on the training set $err_S(h)$ is substantially lower than $err_D(h)$.

Probably Approximately Correct (PAC) Learning

- $$m > \frac{n}{\epsilon}\left(\log(n) + \log\left(\frac{1}{\delta}\right)\right)$$
- The concept class C is **PAC learnable** by L using H if for $f \in C$, for all distribution D over X, and fixed $\epsilon > 0, \delta < 1$, given m examples sampled i.i.d. according to D, the algorithm L produces, with probability at least $(1 - \delta)$, a hypothesis $h \in H$ that has error at most ϵ, where m is polynomial in $1/\epsilon$, $1/\delta$, n and size H.
- The concept class C is **efficiently learnable** if L can produce the hypothesis in time that is polynomial in $1/\delta$,$1/\delta$ and size(H).
- A bound on how much the true error will deviate from the training error. If we have more than m examples, then with high probability $1 - \delta$,

$$err_D(h) - err_S(h) \leq \sqrt{\frac{\ln|H| + \ln(1/\delta)}{2m}}$$

Shattering

- Definition: A set S of examples is **shattered** by a set of functions H if for every partition of the examples in S into positive and negative examples there is a function in H that gives exactly these labels to the examples.
- The **VC dimension** of hypothesis space H over instance space X is the size of the largest finite subset of X that is shattered by H
- $$m > \frac{1}{\epsilon}\left(8VC(H)\log\frac{13}{\epsilon} + 4\log\frac{2}{\delta}\right)$$

  Then with probability at least $(1-\delta)$, h has error less than ϵ.
- If we have m examples, then with probability $1-\delta$,

$$err_D(h) \leq err_S(h) + \sqrt{\frac{VC(H)(\ln\frac{2m}{VC(H)} + 1) + \ln\frac{4}{\delta}}{m}}$$

The End