In [3]:
```
import numpy as np
```

Problem 1

1a: Describe a setting of the hyperparameters gamma and C that is sure to overfit to any typical training set and achieve zero training error.

$\gamma = 10^{-10}, c = 10^{5}$

1b: Describe a setting of the hyperparameters gamma and C that is sure to underfit to any typical training set.

$\gamma = 10^{5}, c = 10^{-10}$

1c: If you were performing grid search to find the SVM that generalized best to unseen data, what range of values for gamma, C would you recommend? Specify your answer with two lines of NumPy code (e.g. using a list like [1, 2, 3] or np.linspace or np.logspace). Also provide 1-2 sentences of justification. Assume that you can't afford more than 5 distinct values for each variable.

In [10]:
```
c_grid=np.logspace(-2.0, 2.0, num=5)
gamma_grid=np.logspace(-2.0, 2.0, num=5)
```

Explanations:
c_grid goes from slightly underfitting to overfitting.
gamma_grid goes from slightly overfitting to underfitting.

1d: Answer the following True/False questions, providing one sentence of explanation for each one

i False. SVM only gives a hyperplane seperating 2 class of points. There will be only classified label (without probability).

ii False. The weight vector (especially after kernelized) will be dense.

iii False. The kernel matrix can only proved to be semidefinite. In fact, if you have more data points than features, the linear kernel matrix(which is the covarience matrix) is surely not invertible because of linear dependence on data points.

Problem 2

2a: Describe a setting of the hyperparameters n_estimators, max_features, and min_samples_leaf that is sure to overfit to any typical training set. You can assume no

BAgging (bootstrap=False).

```
n_estimator=1, max_features=F, min_samples_leaf=1
```

2b: Describe a setting of the hyperparameters n_estimators, max_features, and min_samples_leaf that is sure to underfit to any typical training set. You can assume no BAgging (bootstrap=False).

n_estimator=1, max_features=1, min_samples_leaf=N

2c: If you were performing grid search to find the RandomForest that generalized best to unseen data, what range of values for n_estimators, max_features, and min_samples_leaf would you recommend? Should we set bootstrap to True or False? Specify your answer with a few lines of NumPy code (e.g. using a list like [1, 2, 3] or np.linspace or np.logspace to set each variable). Include a sentence or two of justification. Assume that you can't afford more than 5 distinct values for each variable.

```
n_estimators_grid = [3,5,10,20,50]
max_features_grid = [F/10, F/5, F/3, F/2, 2*F/3]
min_samples_leaf_grid = [5, 10, N/10, N/5, N/3]
bootstrap_grid = [True, False]
```

Explanations:
n_estimators_grid is the number of trees we are training. The more trees we have, the more objective the voting system is as well as the longer time it will take. So we want to try a few numbers to see which gives a good accuracy while taking reasonale time to get result.

max_features suggests how distinguish the trees are. We want use less features to get distinguish trees for a good prediction. On the other hand, more features mean a better fitting on the training set. So there is a trade off.

min_samples_leaf_grid suggests how many examples on each classifier is going to have. When it's small, the model will be overfitting. When it's big, the model will be overfitting. So we go from a small number to a larger propotion of total number N of examples.

Boostrap_grid only have true or false. So we just try both of them.

2d: Answer the following True/False questions, providing one sentence of explanation for each one:

i True. There is a probabilty that the number K of trees suggest the data goes to a type out of the total number N trees, which gives a probability of $\frac{K}{N}$.

ii False. Large number of tree used is not a key factor of overfitting or underfitting. The number of feature used in trees is more important.

iii False. We want our trees to be diverged so the prediction is various. If we have similar features in each trees, due to the greedy algorithm, the prediction will be identical, which means the voting or averaging system won't work.

Iv False. We also randomly choose features randomly.

Problem 3

3a: Can we apply ROC curves to binary classifiers that cannot easily produce probabilities p (0,1), but produce some real-valued scores s R for each example? How would we select the range of thresholds to evaluate?

One thing we can do is to do a min_max transform so that we have numbers between (0,1).

Another thing is we just put a threshold(i.e quintile or decile) on S and calculate FPR and TPR directly.

3b: Suppose you fit a classifier to data, and you observe a TPR of 0.3 and an FPR of 0.7. Your friend says that this is worse than a random classifier (if plotted on an ROC curve, would fall below the TPR=FPR diagonal line), and so you should throw this result away. Is there anything better you can do? Describe how to tranform the predicted binary labels to reach better performance. What TPR and FPR would you expect to achieve?

Yes. We can switch the label(0 to 1 and 1 to 0).

Then TPR will be 0.7 and FPR will be 0.3.

In [ ]: