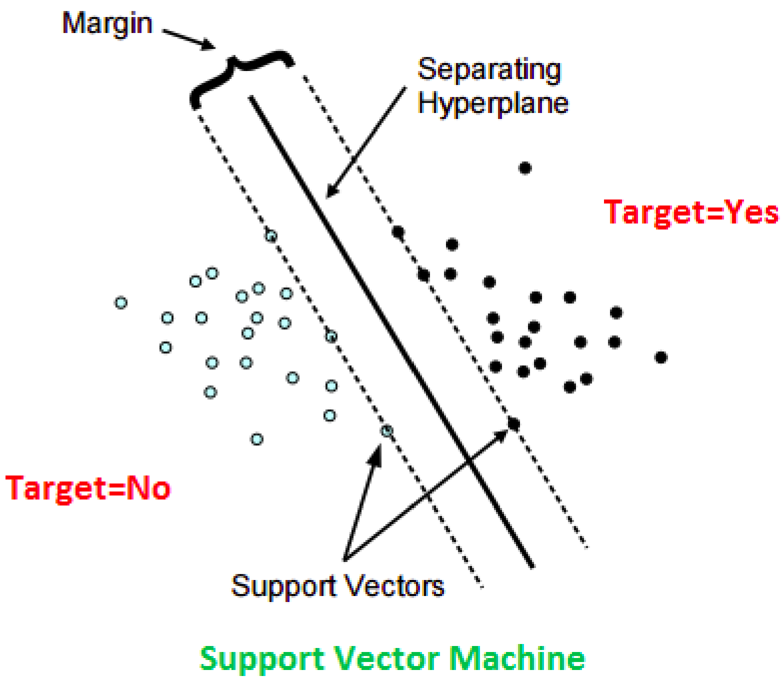
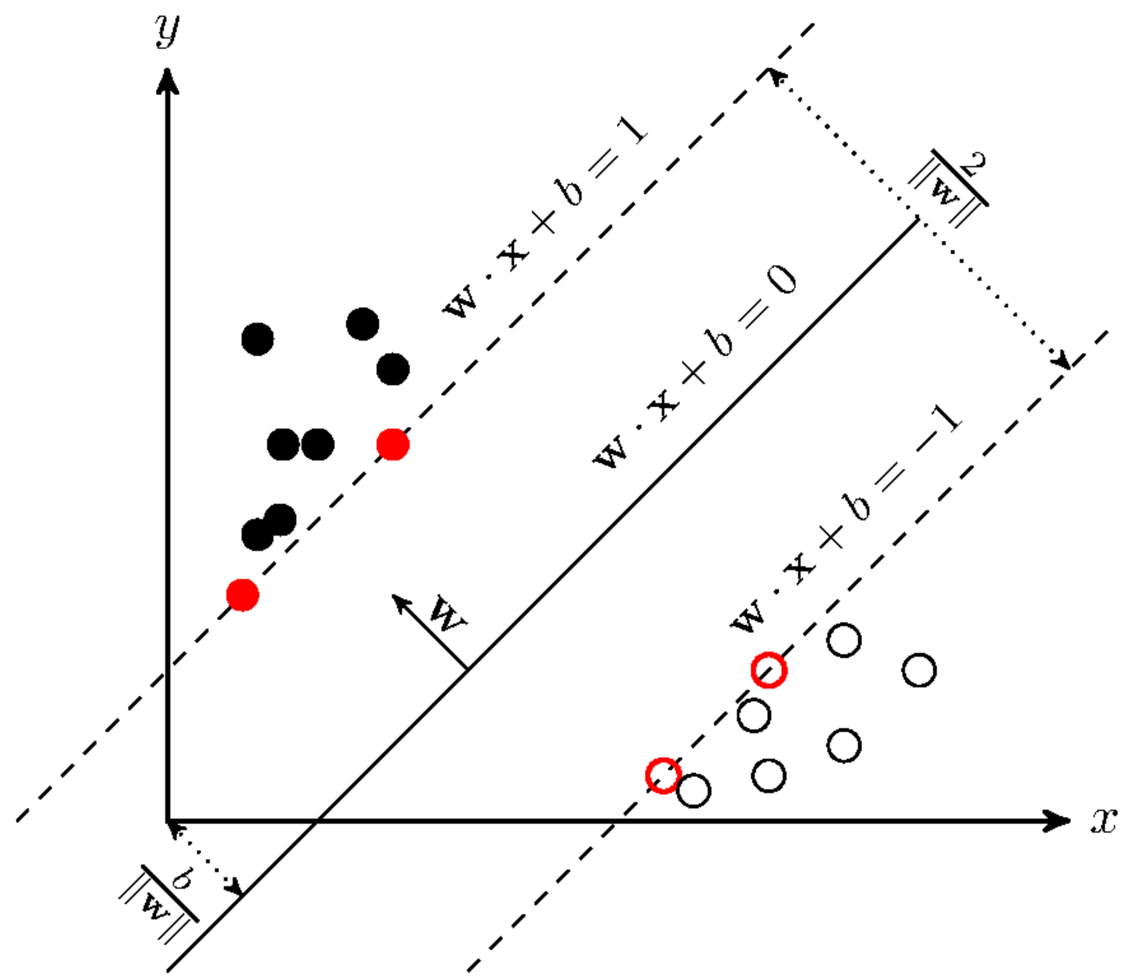
Support Vector Machines (SVM) and Support Vector Regression (SVR)

Dustin McAfee

02/12/19

Support Vector Machine



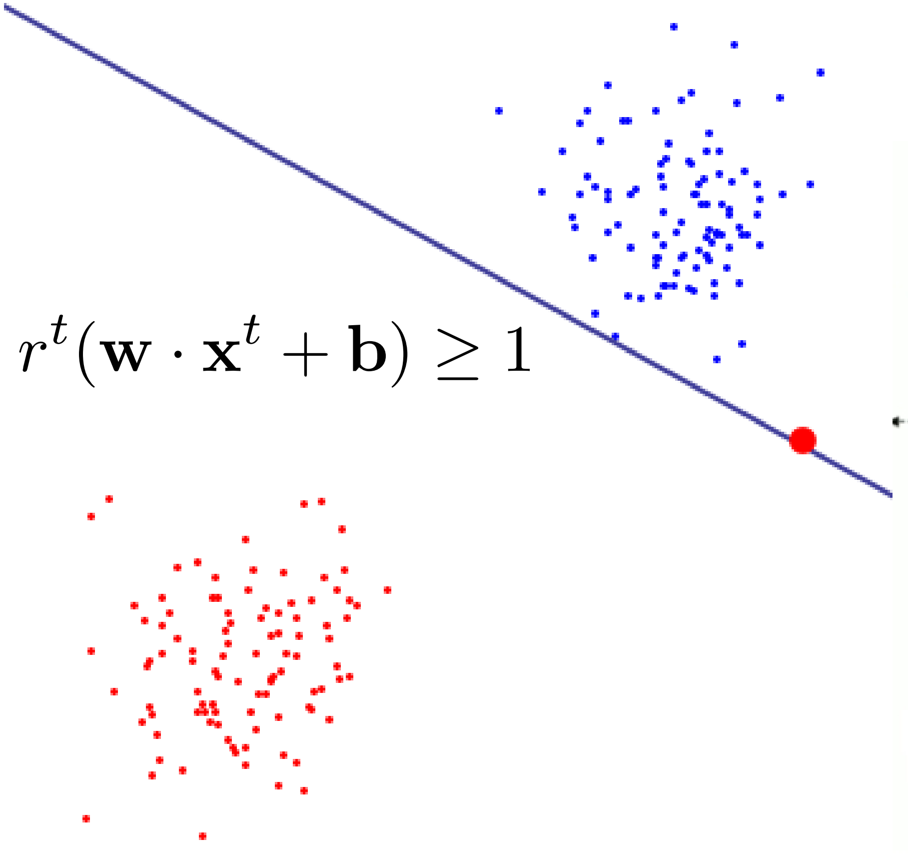


Maximize Margins = Maximizing $\frac{2}{||\mathbf{w}||}$

= Minimizing $||\mathbf{w}||$

https://github.com/EvEng/papers/blob/master/SVMandSVR.md

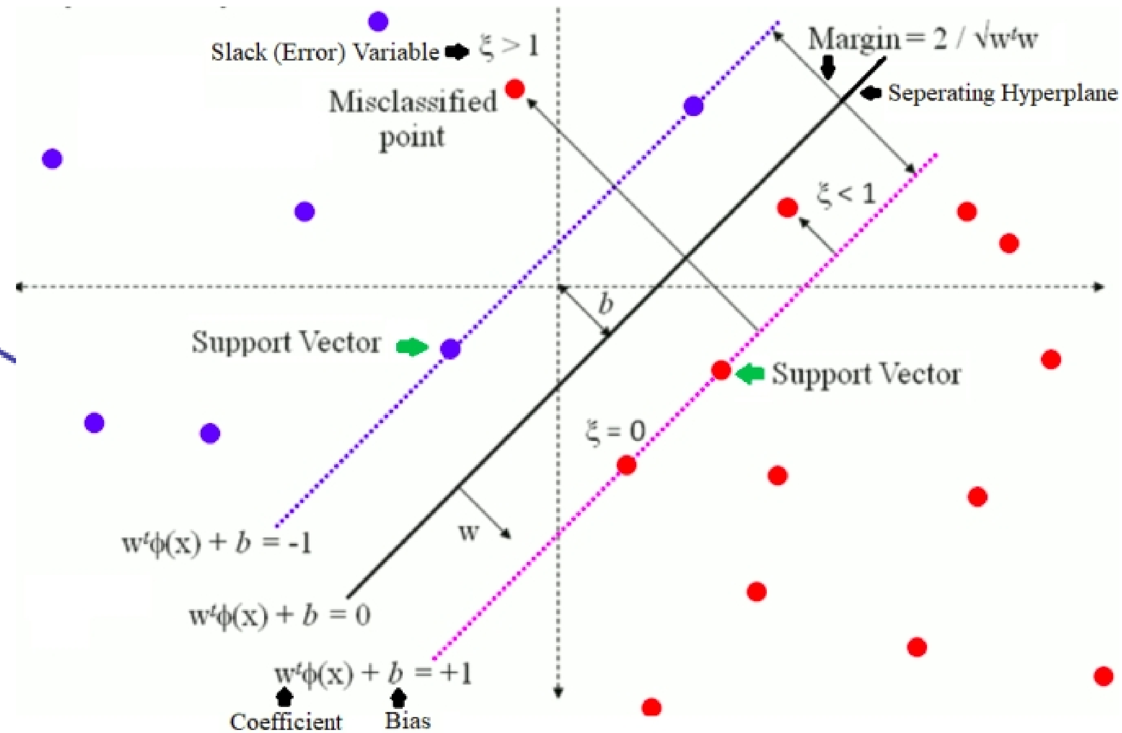
Hard/Soft Margins



$$r^t(\mathbf{w} \cdot \mathbf{x}^t + \mathbf{b}) \geq 1$$

https://stackoverflow.com/questions/4629505/svm-hard-or-soft-margins

$$r^t(\mathbf{w} \cdot \mathbf{x}^t + \mathbf{b}) \geq 1 - \xi^t$$



https://www.datasciencecentral.com/profiles/blogs/implementing-a-soft-margin-kernelized-support-vector-machine

In order to maximize the margin and minimize error, we must minimize: $\frac{1}{2}||\mathbf{w}||^2 + C\sum_t \xi^t$

Subject to: $r^t(\mathbf{w} \cdot \mathbf{x}^t + b) \geq 1 - \xi^t$, where C is the complexity factor (otherwise known as penalty). This is called the Primal Problem.

Dual Problem

Solving for the Lagrangian dual of this problem obtains the simpler problem (Note the lack of dependence on **w** and **b**):

$$\mathbf{w} = \sum_t a^t r^t \mathbf{x}^t$$

$$\text{maximize:} \quad \sum_t \alpha^t - \frac{1}{2} \sum_t \sum_s \alpha^t \alpha^s r^t r^s (\mathbf{x}^t \cdot \mathbf{x}^s)$$

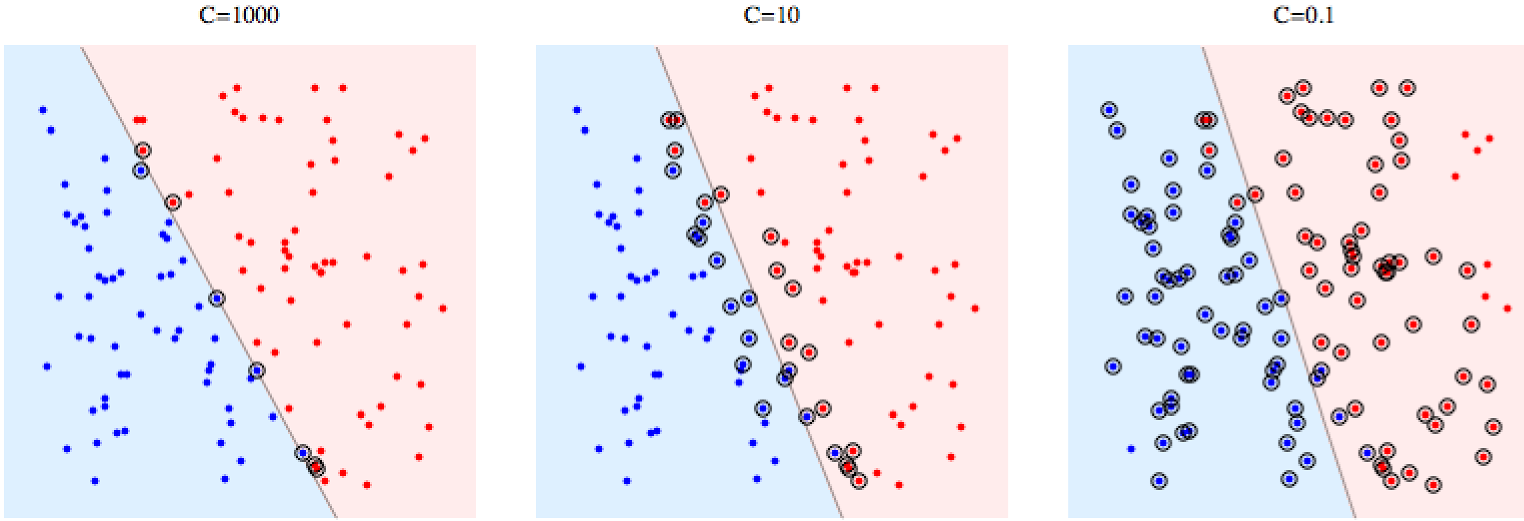
$$\text{subject to:} \quad \sum_t \alpha^t r^t = 0, \text{and } 0 \leq \alpha^t \leq \frac{1}{n}$$

The equation $(\mathbf{x}^t \cdot \mathbf{x}^s)$ is called the kernel, and when taken as a literal dot product, creates a linear hyperplane discriminant (separating hyperplane).

A widely used kernel used is called the radial basis function (RBF): $e^{-\gamma ||\mathbf{x}^t - \mathbf{x}^s||^2}$, where $\gamma > 0$ is a hyper-parameter. Small Gamma means a Gaussian discriminant hyper-plane with a large variance, which means the influence of $\mathbf{x}^s$ spans far in the hyper-space (and vice versa for small Gamma).

Hyper-parameters:

- Penalty (C): Increasing C makes the model move closer to hard-margin. It optimizes how much you would want to avoid misclassifying each training example.

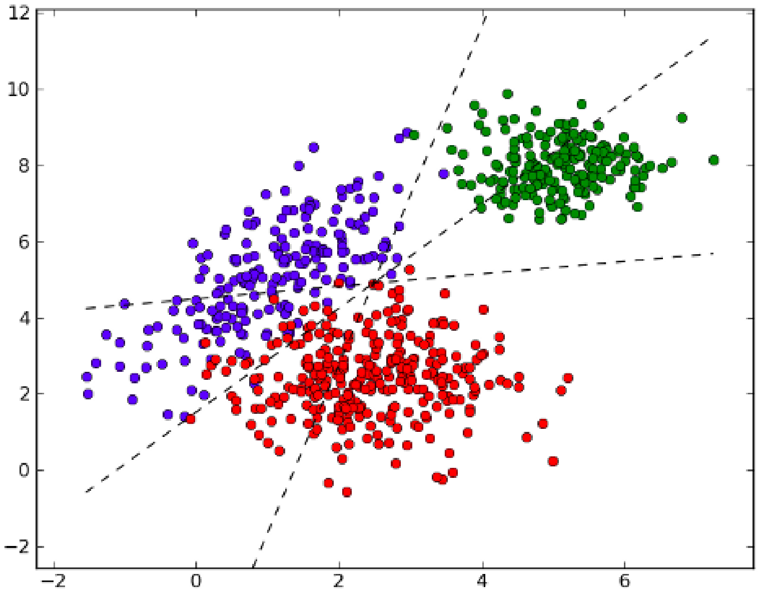


https://github.com/EvEng/papers/blob/master/SVMandSVR.md

- Kernel Function: Determines the shape of the discriminating hyper-plane.
  - Linear: $(\mathbf{x}^t \cdot \mathbf{x}^s)$
  - Polynomial (degree d): $(\mathbf{x}^t \cdot \mathbf{x}^s)^d$
  - Radial Basis Function (extra hyper-parameter Gamma): $e^{-\gamma||\mathbf{x}^t - \mathbf{x}^s||^2}$
  - Sigmoid Function (extra hyper-parameters Gamma and r): $\tanh(\gamma(\mathbf{x}^t \cdot \mathbf{x}^s) + r)$
    - Sigmoid Function is equivalent to a two-layer ANN

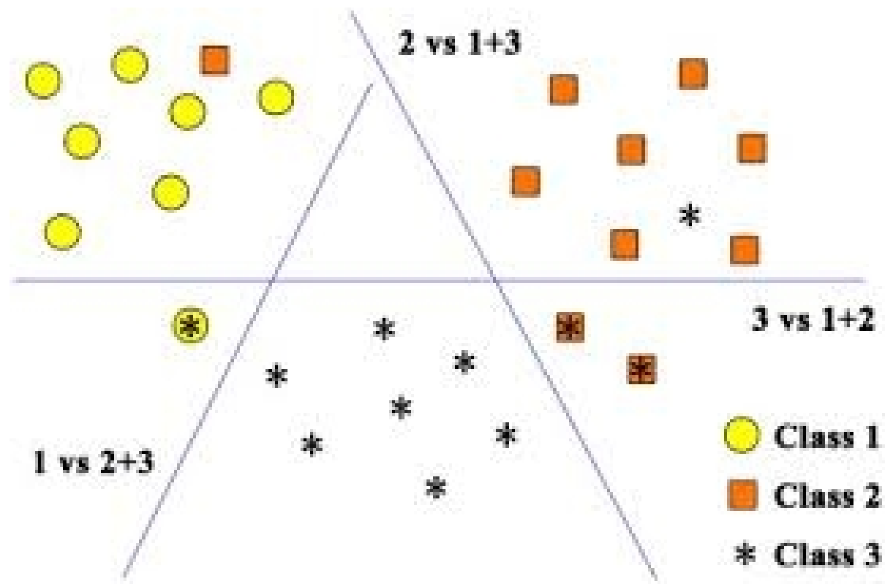
Multi-Classification

OVO



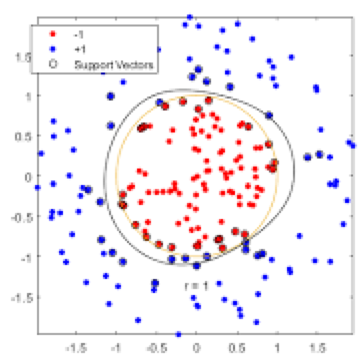
http://www.writeopinions.com/multiclass-classification

OVR



https://doi.org/10.1016/j.neucom.2011.04.024

Gaussian RBF Kernel:



https://it.mathworks.com/examples/statistics/mw/stats-ex18316588-train-svm-classifiers-using-a-gaussian-kernel

Grid Searches

- Coarse Grain:
  - Train with a large-step range of hyper-parameters, such as C = [0.1, 1, 10, 100, 1000, 10000].
  - Compare performance results such as Area Under Curve (AUC) Receiver Operating Characteristics (ROC) in order to find a smaller range of hyper-parameters to test.
  - Perform fine grain grid search next with all appropriate ranges of hyper-parameters.
- Fine Grain:
  - With prior knowledge of which hyper-parameters perform best on the training dataset, train with a small-step range of hyper-parameters centered around what is guessed to perform the best. Example range would look like C = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1].

For each choice hyper-parameter C, say 0.1, the SVM must be trained with different values of the kernel hyper-parameter (degree, Gamma, etc.) according to the ranges chosen for the grid search.

Practical Applications

- Laufer et al. [1] used an SVM classifier to distinguish between heart and kidney tissues using electrical measurements from the area around the tissues. The tissue type was correctly determined with a specificity of over 90 percent [2].
- Tabesh et al. [2] used an SVM classifier with trial and error approach to tuning hyper-parameters for distinguishing heart disease using medical data (ECG data, blood sugar, chest pain, etc.) and medical meta-data (Sex, Age, Ethnicity, etc.).
  - The initial experiment had about 65.8% accuracy: 82% accurate on categorical data, and 55% accurate on continuous data.
  - Continuous tuning of the hyper-parameters for the Gaussian RBF Kernel and SVM eventually yielded an overall accuracy of 84%.
- I trained an SVM classifier using grid searches for linear, polynomial, and RBF Kernels for the vowel-context dataset (digitally recorded vowel sounds) to see what kind of precision I could accomplish (without over-generalization).
  - C = 2 and Gamma = 5 results in 94.2% precision and 97.6% ROC AUC for the testing dataset
  - 97.7% precision and 98.9% ROC AUC for the training dataset
  - Implies unbiased generalization.

What a Grid Search Looks Like

Coarse Grid Search

Hyper-Parameter C	Hyper-Parameter γ	Mean ROC AUC	Mean Precision
≤ 1	≤ 1	≤ 0.636	≤ 0.334
1	10	0.992 ± 0.004	0.984 ± 0.007
1	≥ 100	1.0 ± 0.0	1.0 ± 0.0
10	≤ 0.1	≤ 0.597	≤ 0.253
10	1	0.921 ± 0.009	0.831 ± 0.012
10	≥ 10	1.0 ± 1.0	1.0 ± 1.0
100	≤ 0.1	≤ 0.718	≤ 0.458
100	1	0.997 ± 0.001	0.989 ± 0.005
100	≥ 10	1.0 ± 0.0	1.0 ± 0.0
1000	≤ 0.01	≤ 0.630	≤ 0.298
1000	0.1	0.920 ± 0.010	0.821 ± 0.018
1000	≥ 1	1.0 ± 0.0	1.0 ± 0.0
10000	≤ 0.01	≤ 0.722	≤ 0.461
10000	0.1	0.989 ± 0.007	0.965 ± 0.012
10000	≥ 1	1.0 ± 0.0	1.0 ± 0.0
10000	≤ 0.001	≤ 0.637	≤ 0.310
10000	0.01	0.916 ± 0.011	0.811 ± 0.018
10000	≥ 0.1	1.0 ± 0.0	1.0 ± 0.0

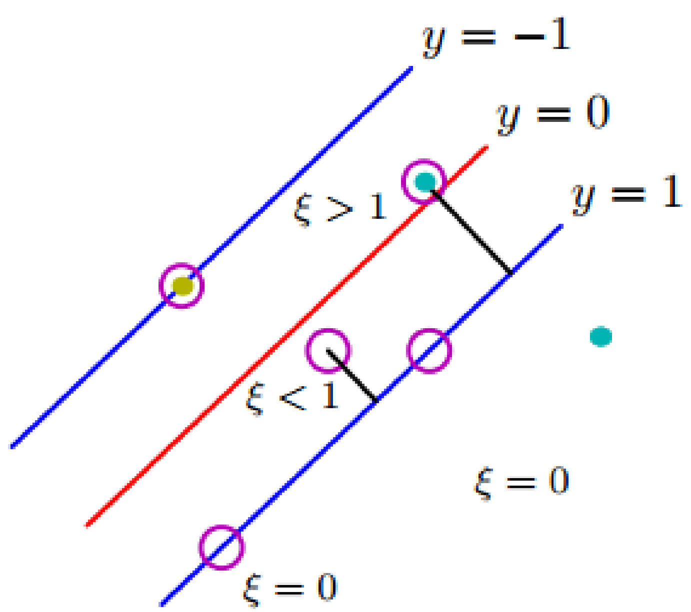
Find the “elbow point”: Where the performance measurements stop increasing drastically, and start leveling off.

Fine Grid Search

Hyper-Parameter C	Hyper-Parameter γ	Mean ROC AUC	Mean Precision
1	4	0.915 ± 0.008	0.840 ± 0.017
1	5	0.945 ± 0.010	0.898 ± 0.019
1	6	0.964 ± 0.007	0.934 ± 0.013
1	7	0.975 ± 0.006	0.954 ± 0.012
1	8	0.984 ± 0.004	0.970 ± 0.009
2	3	0.947 ± 0.009	0.894 ± 0.019
2	4	0.973 ± 0.003	0.948 ± 0.009
2	5	0.989 ± 0.008	0.977 ± 0.016
2	6	0.995 ± 0.005	0.989 ± 0.008
2	7	0.997 ± 0.003	0.993 ± 0.005
2	8	0.998 ± 0.002	0.995 ± 0.004
3	2	0.924 ± 0.008	0.848 ± 0.013
3	3	0.971 ± 0.006	0.940 ± 0.012
3	4	0.990 ± 0.006	0.978 ± 0.011
3	≥ 5	≥ 0.996	≥ 0.991
4	2	0.947 ± 0.006	0.891 ± 0.012
4	3	0.984 ± 0.006	0.964 ± 0.012
4	≥ 4	≥ 0.995	≥ 0.988
5	2	0.960 ± 0.006	0.916 ± 0.010
5	3	0.991 ± 0.005	0.979 ± 0.010
5	≥ 4	≥ 0.997	≥ 0.992
6	2	0.970 ± 0.007	0.936 ± 0.011
6	≥ 3	≥ 0.993	≥ 0.984
7	2	0.977 ± 0.008	0.948 ± 0.015
7	≥ 3	≥ 0.995	≥ 0.988
8	1	0.901 ± 0.003	0.795 ± 0.009
8	2	0.982 ± 0.006	0.958 ± 0.010
8	≥ 3	≥ 0.996	≥ 0.990

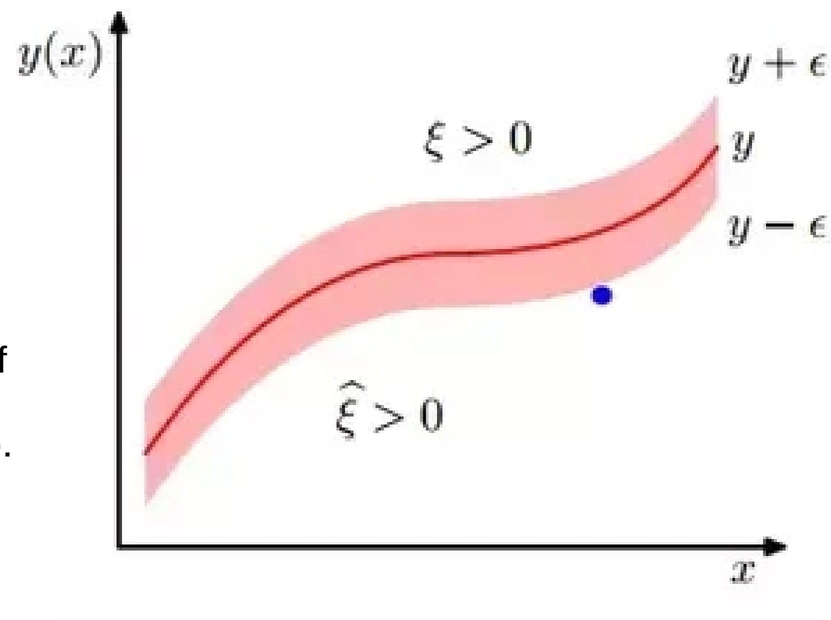
Support Vector Regression (SVR)

SVM



- For SVR we still want to maximize the margin (in order to find the line of best fit).
- However, measuring the error is now equivalent to measuring the distance of the points outside of the margin (opposite of SVM).

SVR



https://www.quora.com/What-is-the-main-difference-between-a-SVM-and-SVR

In order to maximize the margin and minimize error, we must minimize: $\frac{1}{2}||\mathbf{w}||^2 + C\sum_t(\xi^t + \hat{\xi}^t)$

Subject to: $y(\mathbf{x}^t) - (\mathbf{w} \cdot \mathbf{x}^t + b) \leq \epsilon + \hat{\xi}^t$ and $(\mathbf{w} \cdot \mathbf{x}^t + b) - y(\mathbf{x}^t) \leq \epsilon + \xi^t$

Useful Python Libraries

- from sklearn import preprocessing # For standardizing
- from sklearn import svm # For OVO, SVM, and SVR
- from sklearn.multiclass import OneVsRestClassifier # For OVA
- from sklearn import model_selection # For Grid Search
- from sklearn.model_selection import cross_validate # For Cross-Validation
- from sklearn.metrics import make_scorer # For Creating Custom Performance Metrics
- from sklearn.metrics import average_precision_score # For Calculating Average Precision
- from sklearn.metrics import f1_score # For using F1 Score Metric
- from sklearn.metrics import roc_auc_score # For using ROC AUC Metric

And of course, import numpy

Relevant Papers

[1] Laufer S, Rubinsky B (2009) Cellular Phone Enabled Non-Invasive Tissue Classifier. PLOS ONE 4(4): e5178. https://doi.org/10.1371/journal.pone.0005178

[2] Tabesh, Pooya & Lim, Gino & Khator, Suresh & Dacso, Clifford. (2010). A support vector machine approach for predicting heart conditions. Full copy available at: https://www.researchgate.net/publication/290542607_A_support_vector_machine_approach_for_predicting_heart_conditions