Support Vector Machines

Marc Berghouse and Lazaro Perez

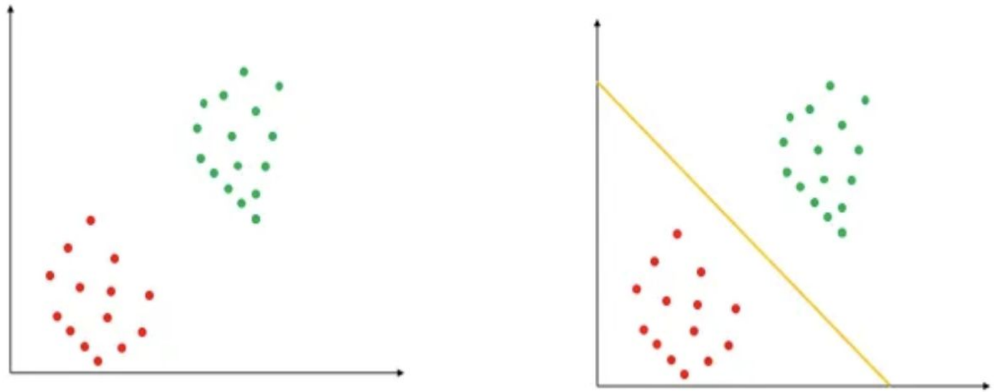
Upcoming Plans

- Homework 3
  - 10-15 minute presentation to the class about a paper that uses machine learning to solve a problem in hydrology
    - We will supply some paper suggestions, but you should try to use this homework to find a project topic
    - People can work in pairs but it will then be a 20-30 minute presentation
    - We will plan to do the presentations the first week of April
- Homework 4
  - Undecided assignment on applying ML to a hydro dataset
  - Due the third week of April
- Semester project
  - Use ML to answer a question in hydrology
    - Explore a novel problem/solution
    - Use a new approach to add additional insight to an old problem
    - Do a model comparison study
  - In the last week of the semester, you'll do a 20-25 minute presentation (or 30-45 for pairs)
  - On the day of the final, you'll turn in a written report
  - We will send out a specific rubric soon
  - But generally, the paper and report should include intro, methods, results and discussion

Support Vector Machines (SVM)

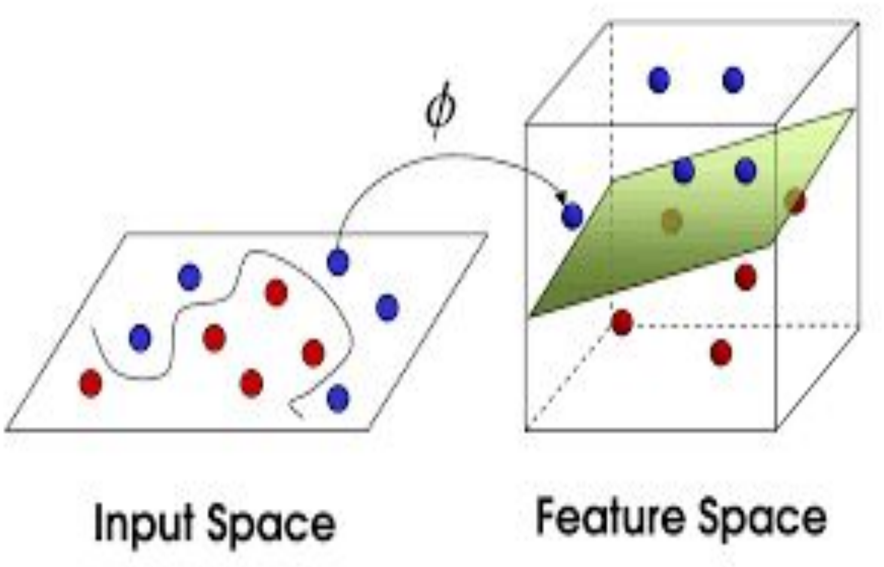
- Developed by Vladmir Vapnik and colleagues at AT&T Bell Labs in the 1990's
- Support vector classifieers (SVCs) were created for binary classification of linearly separable data
  - Kernel methods are used for non-linear data
  - SVR is the regression version of SVC
- Supervised learning (uses training data to fit coefficients)
- Derived from statistical learning theory

https://towardsdatascience.com/an-introduction-to-support-vector-machine-3f353241303b



How Do SVCs Work?

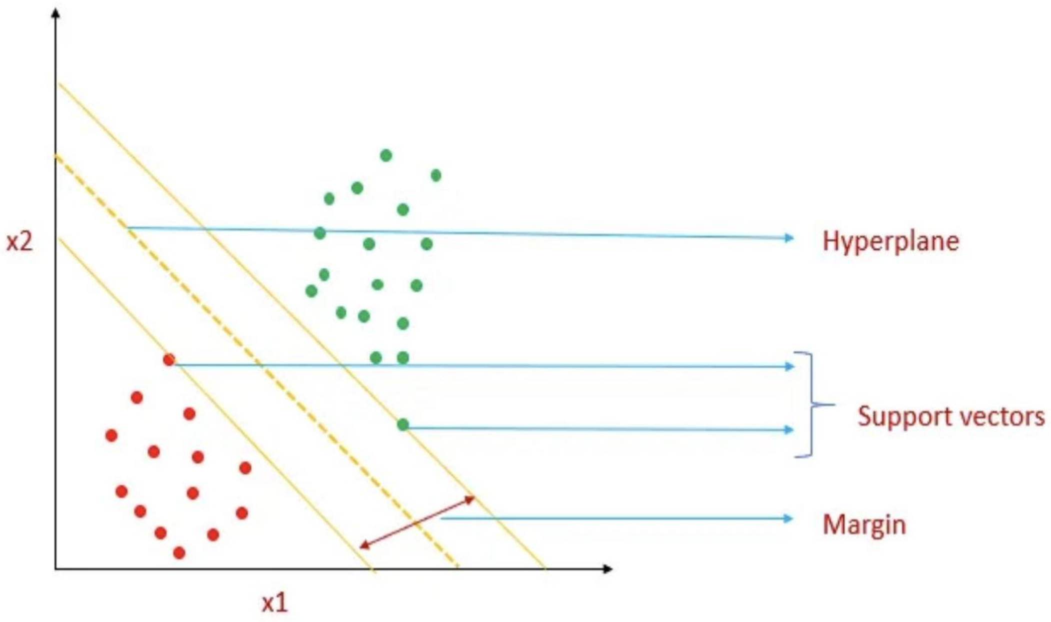
- SVCs map the data to a higher dimensional feature space then find an optimal hyperplane that divides the data into two classes
- What is a hyperplane?
  - An n-1 dimensional subspace in an n dimensional space
    - We normally only call things hyperplanes in they are >= 3 dimensions
  - Imagine we have a dataset with 10 variables. This represents a 9-dimensional space (1 target and 9 features). SVC will find the optimal 8-dimensional hyperplane to separate the target variable into two classes



https://socialmetwork.blog/2021/04/23/support-vector-machine-for-classification-of-space-weather/

Key Model Components

- Support Vectors
  - Think of these as end-members that represent the closest data point of a class to the hyperplane
  - Used to define the margin
  - Will change the optimal hyperplane if changed
- Feature vectors
- Margin
  - Hard margin
  - Soft margin
- Kernel methods
  - Used for non-linearly separable data



https://towardsdatascience.com/an-introduction-to-support-vector-machine-3f353241303b

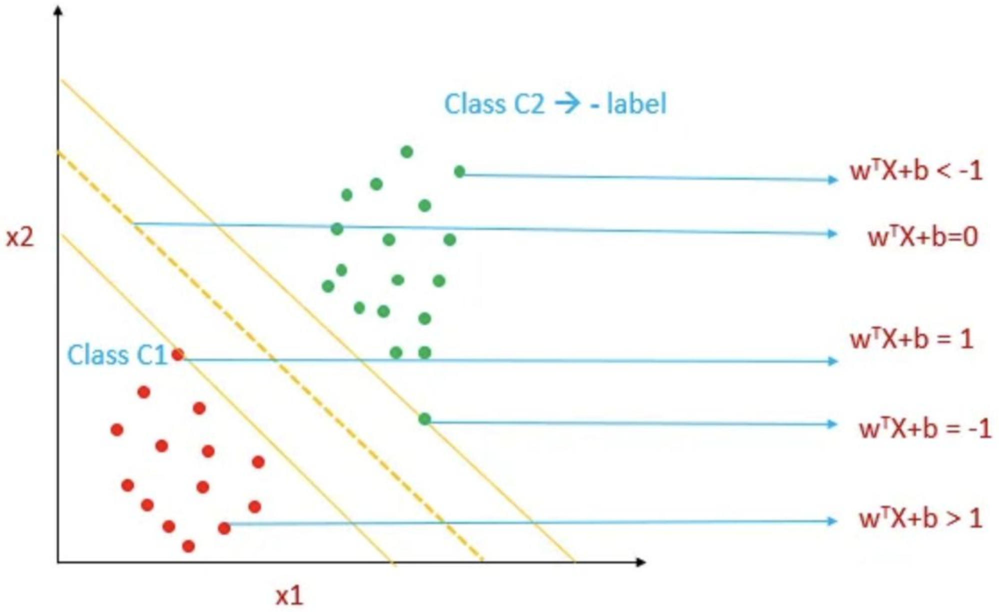
The Math Behind SVC

$$g(X) = w^T X + b = 0$$

- It's all linear algebra
- We can use this equation to separate the classes
- X represents a feature vector we are trying to classify
- w represents a weight vector that is perpendicular to the hyperplane (defines its slope)
    - The t
- b is the y-intercept, or the position of the hyperplane in the feature space
- In a 2D space, this all simplifies to y = mx+b

The Math Behind SVC

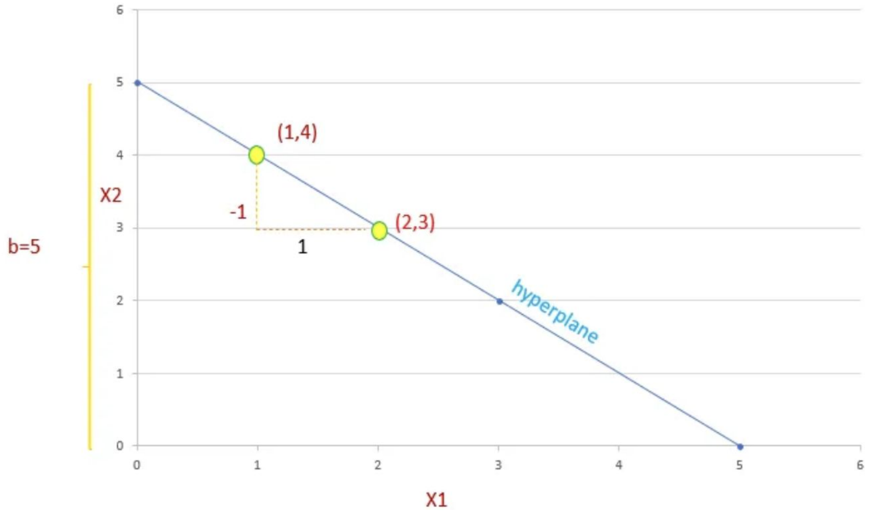
- Finding the optimal value for $w^TX+b = 0$ will define the position and slope of the hyperplane
- The margins are then defined by $w^TX+b = 1$ and $w^TX+b = -1$
  - Parallel to hyperplane
- Feature vectors lie outside the margins ($w^TX+b > 1$ or $w^TX+b < 1$)

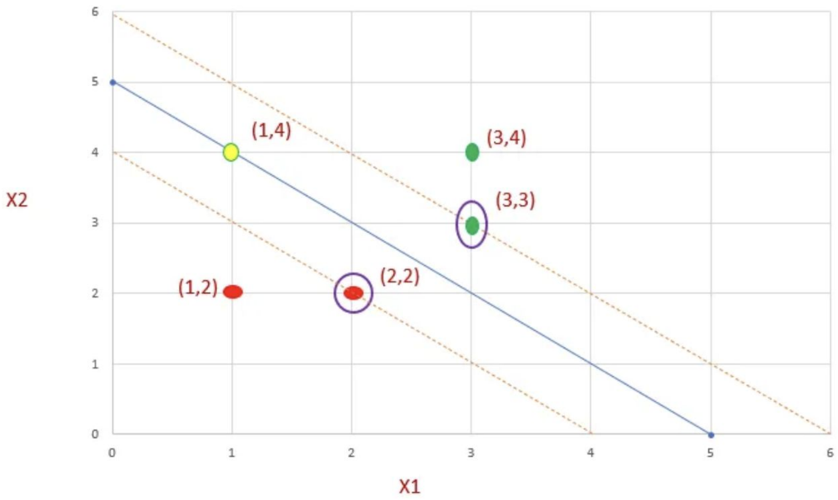


https://towardsdatascience.com/an-introduction-to-support-vector-machine-3f353241303b

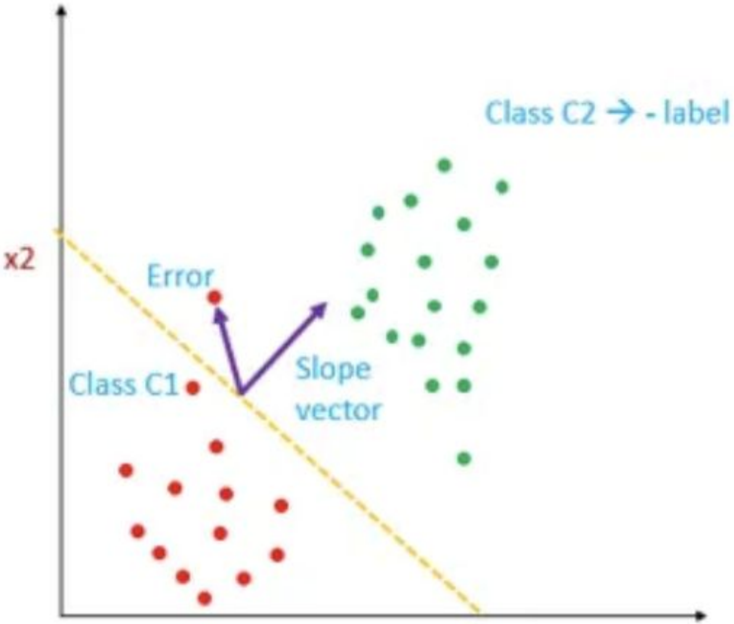
Example Calculation

- Let's say we have a hyperplane with w^T=-1 and b=5
- For the vector (2,2)
- g(X)=⟨-1,-1⟩*⟨2,2⟩+ 5 = -1*2+-1*2+5 = 1, so this is a support vector for the positive class
- For the vector (3,4)
- g(X)=⟨-1,-1⟩*⟨3,4⟩+ 5 = -1*3+-1*4+5 = -2, so this feature vector belongs to the negative class

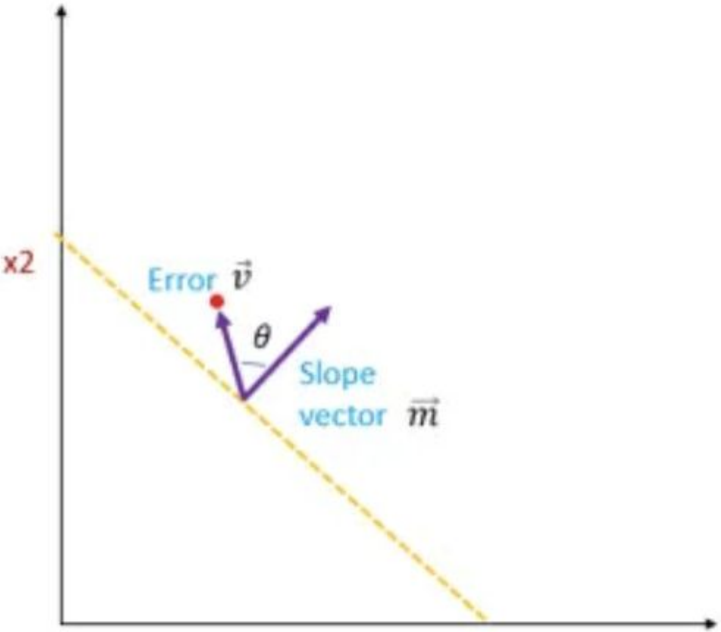




Optimization of the Hyperplane

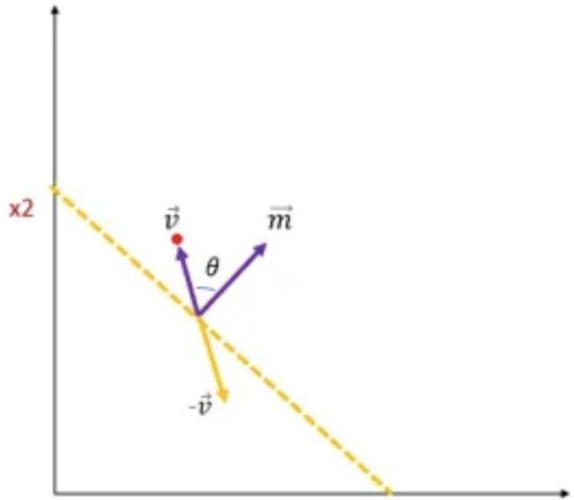


Random hyperplane

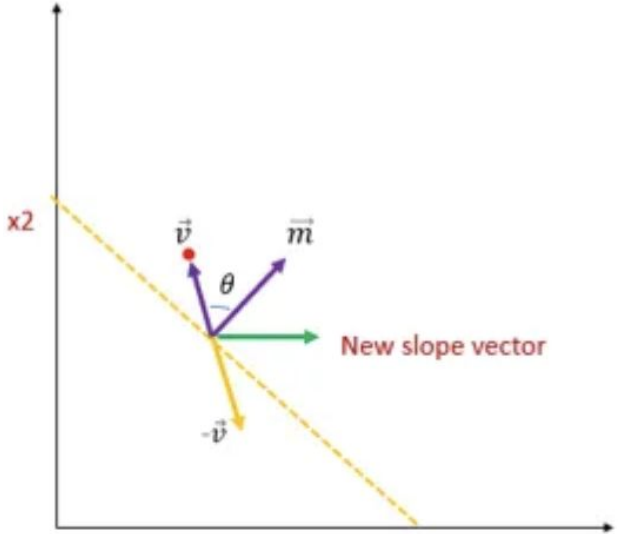


Draw a slope vector and draw a vector the error point

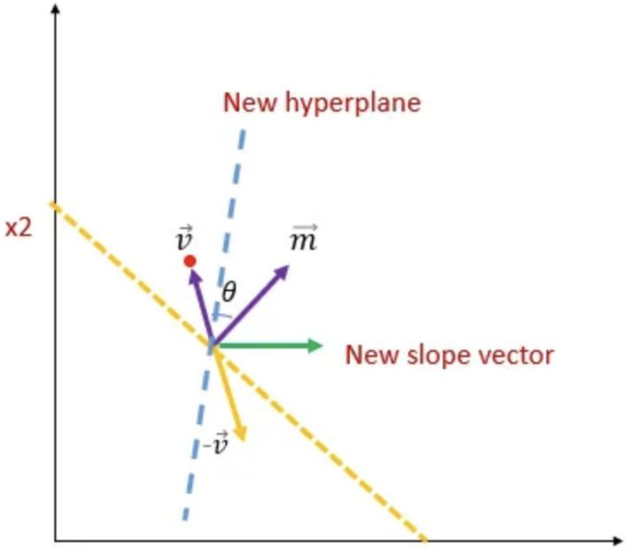
Optimization of the Hyperplane



Flip the direction of data vector , so we will get -$\vec{v}$



If we add $\vec{m}$ +(-$\vec{v}$) , we get new slope vector which is shown in green line.



New hyperplane is perpendicular to new slope vector

Pros and Cons of SVC

- Pros
  - Works well when there is a clear separation between classes (but so do most classification models)
  - Effective for high-dimensional data (if you have more rows than columns, try SVC)
  - Memory efficient
  - Flexible due to choice of kernel
  - Good with small datasets
  - Good generalization performance
- Cons
  - Not usable for large datasets (>100,000 data points)
    - Computationally expensive
    - Sometimes you can use linear SVC to help speed things up though
  - Poor performance on noisy data
  - High sensitivity to choice of hyperparameters and kernel

Implementation With Sklearn

sklearn.svm.SVC

class `sklearn.svm.SVC`(**, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*) ¶ [source]

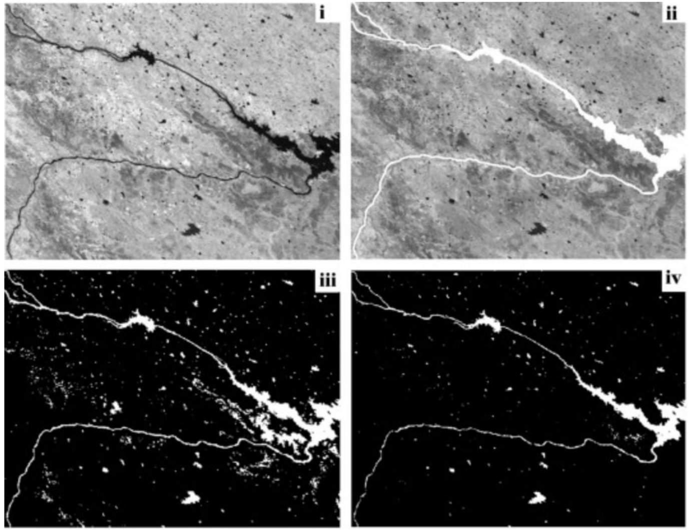
```
>>> import numpy as np
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
>>> y = np.array([1, 1, 2, 2])
>>> from sklearn.svm import SVC
>>> clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
>>> clf.fit(X, y)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(gamma='auto'))])
```

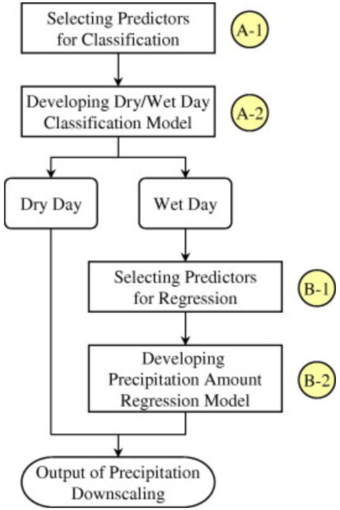
```
>>> from sklearn import datasets, svm
>>> from sklearn.kernel_approximation import Nystroem
>>> X, y = datasets.load_digits(n_class=9, return_X_y=True)
>>> data = X / 16.
>>> clf = svm.LinearSVC()
>>> feature_map_nystroem = Nystroem(gamma=.2,
...                                 random_state=1,
...                                 n_components=300)
>>> data_transformed = feature_map_nystroem.fit_transform(data)
>>> clf.fit(data_transformed, y)
LinearSVC()
>>> clf.score(data_transformed, y)
0.9987...
```

Kernel approximation may result in less accuracy but will greatly reduce the computational load of the problem

Applications in Hydrology

- Water quality classification
  - https://www.mdpi.com/2073-4441/14/19/2939
  - https://www.kaggle.com/code/pedrovinciusmeerholz/waterquality-classification-with-xgboost-and-svm
- General image classification and segmentation
  - Land use/land cover classification
    - https://www.tandfonline.com/doi/full/10.1080/08839514.2021.2014185
  - River/flooding classification
    - https://www.sciencedirect.com/science/article/pii/S0273117712004413
- Classification of soils
  - https://www.mdpi.com/2306-5729/5/1/2
- Downscaling of precipitation
  - https://www.sciencedirect.com/science/article/pii/S0022169410000533





Support Vector Regression (SVR)

https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0#

- SVR uses the same basic concepts as SVC
  - Instead of finding the best hyperplane to separate two classes, we find the hyperplane that contains the max number of data points

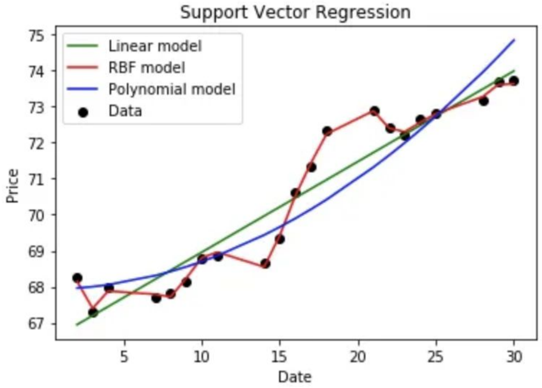
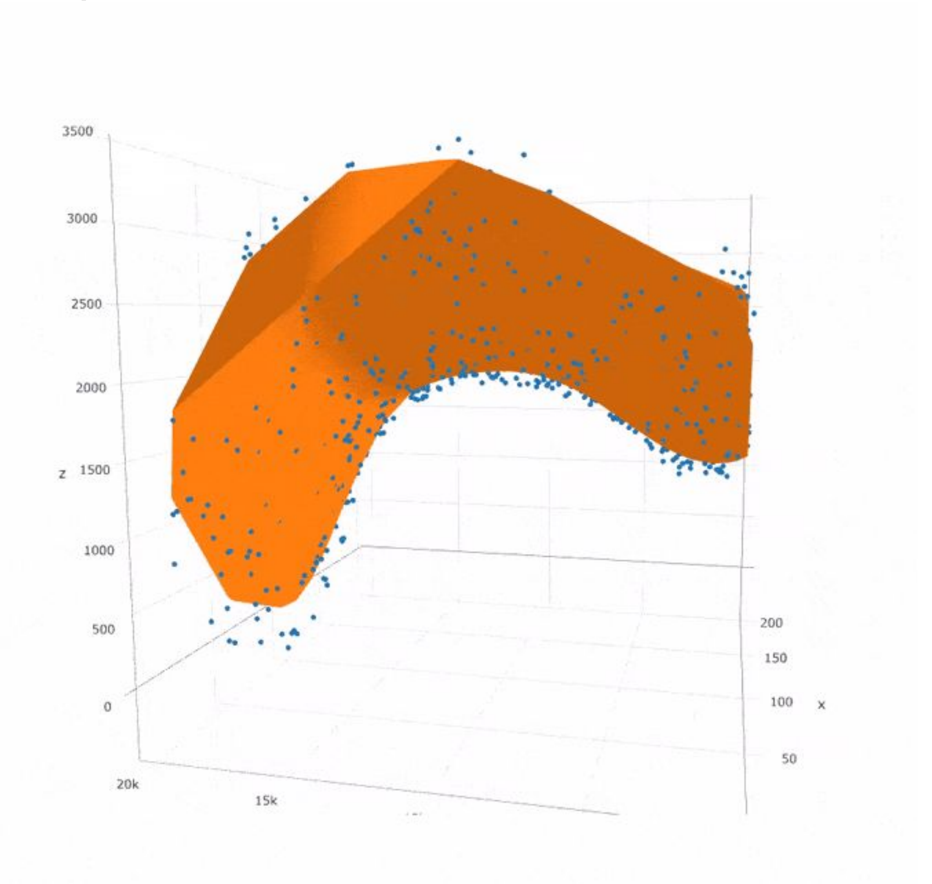


Image by Dale Nguyen



Pros and Cons of SVR

- Essentially the same as SVM
  - Good for lots of features and strong correlations between/clustering of variables
  - Flexible, good with small datasets, strong generalization due to regularization
  - Not suitable for large datasets (although linear SVR with kernel approximation can be used to speed up the calculation)
  - Results are highly dependent on choice of hyperparameters
  - Poor performance for noisy data/data with weak correlations

Implementation With Sklearn (SVR)

sklearn.svm.SVR

class `sklearn.svm.SVR`(**, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=-1*) [source]

- C is a regularization parameter
  - Reduces model complexity/weights of fitting parameters to combat overfitting
- Epsilon is a noise factor that lets you constrain how far a point needs to be from the optimal hyperplane to be included in the loss function

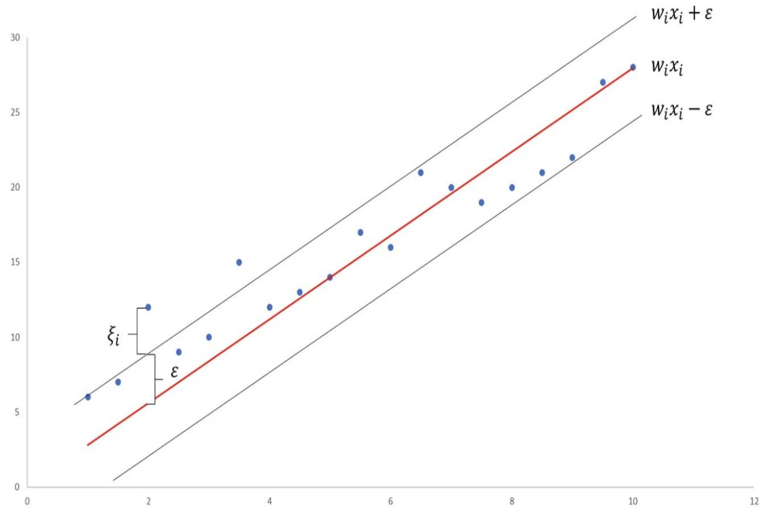
```
>>> from sklearn.svm import SVR
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import StandardScaler
>>> import numpy as np
>>> n_samples, n_features = 10, 5
>>> rng = np.random.RandomState(0)
>>> y = rng.randn(n_samples)
>>> X = rng.randn(n_samples, n_features)
>>> regr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
>>> regr.fit(X, y)
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svr', SVR(epsilon=0.2))])
```

Minimize:

$$MIN \ \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}|\xi_i|$$

Constraints:

$$|y_i - w_i x_i| \le \varepsilon + |\xi_i|$$



https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2

Applications in Hydrology (SVR)

- Rainfall/runoff forecasting
- Streamflow and sediment load forecasting
- Evaporation and evapotranspiration forecasting
- Drought and flood forecasting
- Groundwater level prediction
- Soil moisture prediction
- Review on SVM in hydrology: https://www.sciencedirect.com/science/article/pii/S1568494614000611#bib0120
    - Mainly regression applications