INT104 ARTIFICIAL INTELLIGENCE

L6 – Support Vector Machine

Fang Kang
Fang.kang@xjtlu.edu.cn
Xi'an Jiaotong-Liverpool University
西交利物浦大学

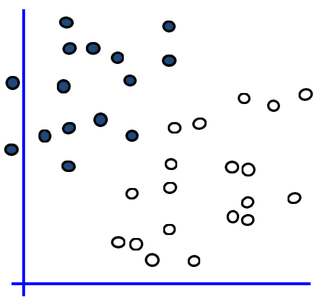
CONTENT

Support Vector Machine

- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis in 1963.
- SVMs are learning systems that
  - use a hyperplane of *linear functions*
  - in a high dimensional feature space — *Kernel function*
  - trained with a learning algorithm from optimization theory — *Lagrangian duality*
  - Implements a learning bias derived from statistical learning theory — *Generalization*

Linear Classifiers

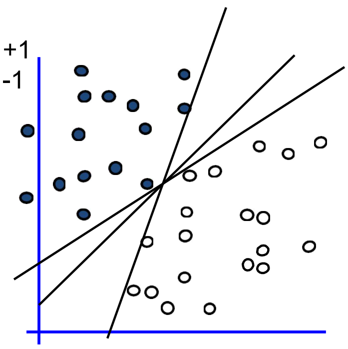
• denotes +1
◦ denotes -1



How would you classify this data?

Linear Classifiers

•denotes +1
◦denotes -1



Any of these would be fine..

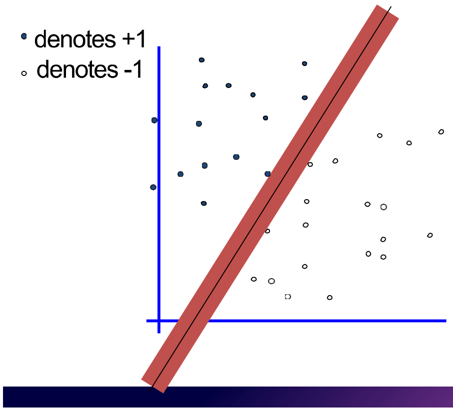
..but which is best?

Classifier Margin

$x \rightarrow$ **f** $\rightarrow$ y^{est}

$w^T x + b = 0$

$f(x,w,b) = sign(w^T x + b)$



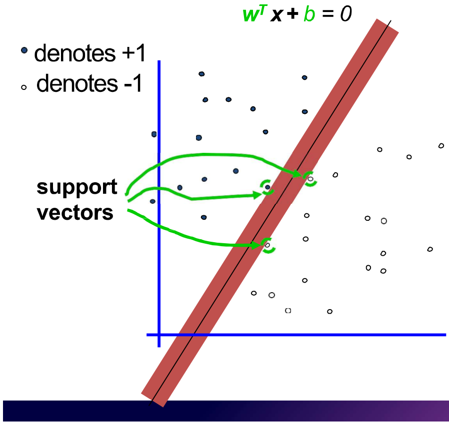
Margin defined as the width of the decision boundary: when the width of the decision boundary grows to exactly touch the instances.

Maximum Margin

$x \rightarrow$ **f** $\rightarrow$ y^{est}

$w^T x + b = 0$

$f(x,w,b) = sign(w^T x + b)$



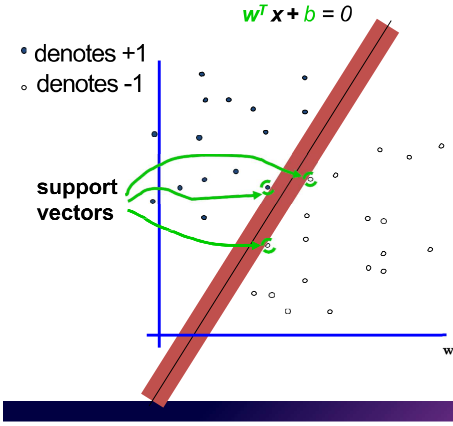
Maximum margin linear classifier is a linear classifier with maximum boundaries. It's the simplest form of SVM, called Linear SVM (LSVM)

Maximum Margin

$x \rightarrow$ **f** $\rightarrow$ y^{est}

$w^T x + b = 0$

$f(x,w,b) = sign(w^T x + b)$



Define:

1. Training data and labels.

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2) \cdots (\mathbf{x_N}, y_N)$$

$$\mathbf{x_1} = \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1M} \end{bmatrix} \qquad y = \begin{cases} +1 \\ -1 \end{cases}$$

2. Linear model

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \qquad w^T x + b = 0$$

hyperplane

Linear Separable

Training data $\{(x_i, y_i)\}_{i=1 \dots N}$

$\exists (\mathbf{w}, b)$ let:

$\forall i = 1 \sim N$ have:

a) If $y_i = +1$ then $\mathbf{w}^\top \mathbf{x}_i + b \geq 0$

b) If $y_i = -1$ then $\mathbf{w}^\top \mathbf{x}_i + b < 0$

$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 0$ (Eq. 1)

Support Vector Machine （SVM）

optimization problem

(convex quadratic optimization problems with linear constraints)

Minimize: $\frac{1}{2}||\mathbf{w}||^2$

Subject to: $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad (i = 1 \sim N)$

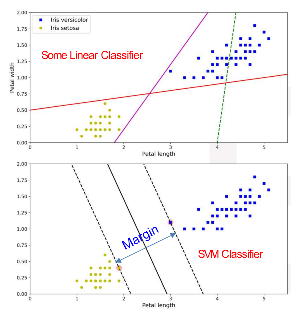
Why?

Linear SVM Classification

- Linear separability
- Fitting widest possible "street" between classes

Performs better with new data

- **Large Margin Classification**
- Margin, Support Vectors



Support Vectors

- Decision boundary is not affected by more training instances
- It is determined by support vectors (instances located on the edge of street)

Feature Scales

- Large Margin Classification

Sensitive to scale

Use Scikit-Learn's StandardScaler

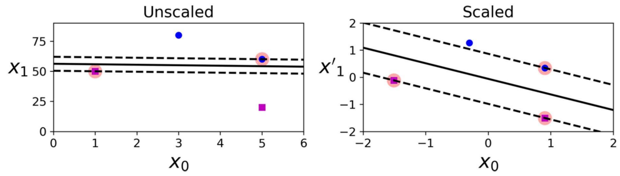


Figure 5-2. Sensitivity to feature scales

- The decision boundary could be much better if the feature is scaled.

Hard Margin Classification

- All instances being off the street and on the right side is named "*hard margin classification*"
- The main limitation of hard margin classification is
  - The data must be linearly separable
  - Sensitive to outliers

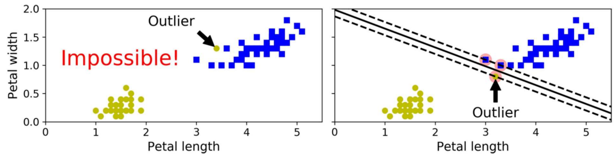


Figure 5-3. Hard margin sensitivity to outliers

Outlier

- What effects does the outlier introduce?
- How do we deal with outliers?

Soft Margin SVM

Minimize: $\frac{1}{2}||\mathbf{w}||^2$

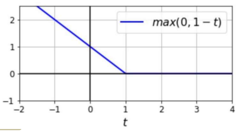
Subject to: $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad (i = 1 \sim N)$

Hard

violate the margin

Minimize: $\frac{1}{2}||\mathbf{w}||^2 + C \cdot \text{loss}$

$\text{loss} = \sum_{i=1}^{N} max\{0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}$ **Hinge loss**



$\xi_i = max\{0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)\}$

Minimize: $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i$ (Slack variable)

Subject to: $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad (i = 1 \sim N)$

$\xi_i \geq 0$

Soft

Soft Margin Classification

Minimize: $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i$ (Slack variable)

Subject to: $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad (i = 1 \sim N)$

$\xi_i \geq 0$

- Allow margin violations
- The algorithm balances
  - The width of street
  - The amount of margin violations
- A hyper-parameter C is defined
  - A low value of C leads to more margin violations
  - A high value of C limits the flexibility

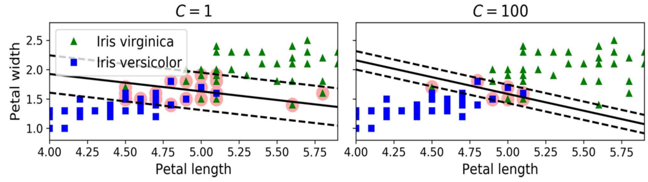


Figure 5-4. Large margin (left) versus fewer margin violations (right)

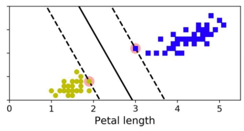
Tip:
- To regularize SVM, reduce C
- C is inverse of regularizing hyperparameter alpha $J(\boldsymbol{\theta}) = \text{MSE}(\boldsymbol{\theta}) + \alpha\frac{1}{2}\sum_{i=1}^{n}\theta_i^2$

Soft Margin Classification

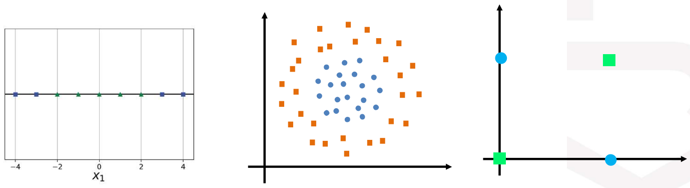
- LinearSVC:
  - `LinearSVC(loss="hinge", C=1)`
  - Accepts two loss functions: `"hinge"` and `"squared_hinge"`
  - Default loss is `"squared_hinge"`
  - Doesn't output support vectors (use `.intercept_` & `.coef_` to find support vectors in the training data)
  - Regularizes bias term too...so center the data by using `StandardScaler`
  - Set `dual=False` if training instances > number of features
- SVC:
  - `SVC(kernel="linear", C=1)`
  - For linear classifier use `kernel="linear"`
  - For hard margin classifier use `C=float("inf") or C=1e10` (a large value)
- SGDClassifier:
  - `SGDClassifier(loss="hinge", alpha = 1/(m*C))`
  - Slow to converge, but good for online or huge datasets

Nonlinear SVM Classification

Linearly separable



How do we deal with these cases?



Not linearly separable

Nonlinear SVM Classification

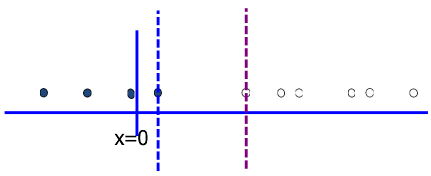
How do we deal with nonlinearity?

Solution

- **1. Directly tackle nonlinear problems**
  Multilayer Perceptron， Deep Learning, etc.
- **2. Transform nonlinear problems into linear issues**
  Kernel SVM

Suppose we're in 1-dimension

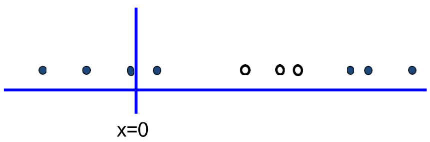
- What would SVMs do with this data?



- Not a big surprise

Harder 1-dimensional dataset

- What would SVMs do with this data?

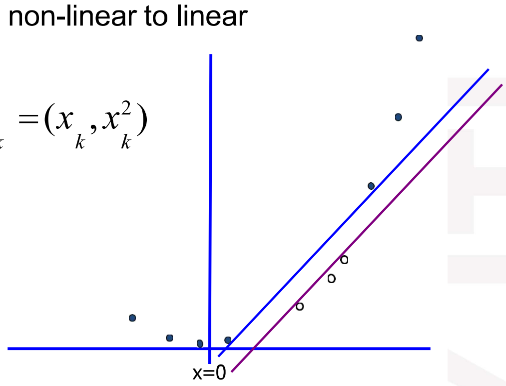


- That's wiped the smirk off SVM's face.
- What can be done about this?

Harder 1-dimensional dataset

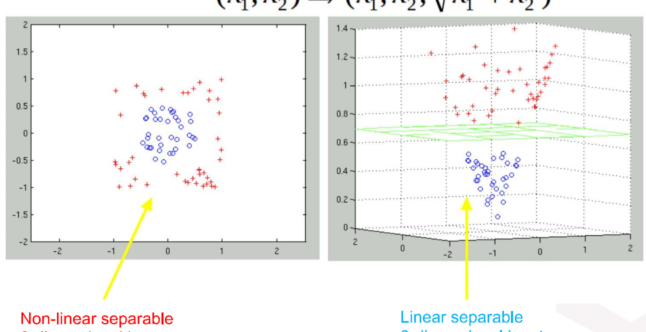
- From non-linear to linear

$$\mathbf{z}_k = (x_k, x_k^2)$$

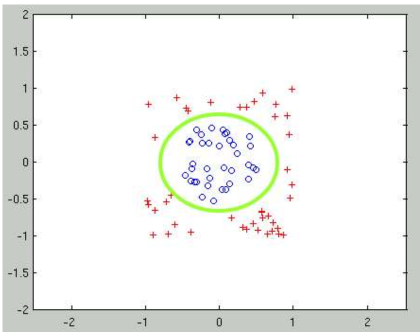


2-dimensional dataset

$$(x_1, x_2) \Rightarrow (x_1, x_2, \sqrt{x_1^2 + x_2^2})$$



When transformed back to R^2, the decision boundary is nonlinear.

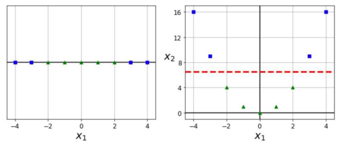


$$\mathbf{z} = \Phi(\mathbf{x}) = (x_1, x_2, \sqrt{x_1^2 + x_2^2})$$

Nonlinear SVM Classification

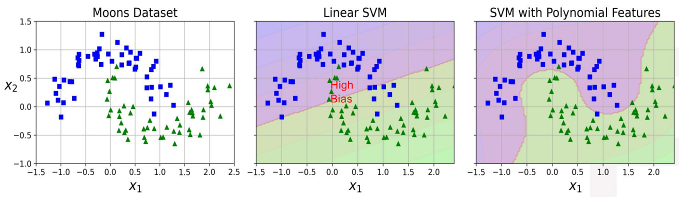
- Classes are not linearly separable in the input space
- What to do? Project input space into a high dimensional feature space.
- Polynomial Features

Polynomial features involve taking an existing feature and raising it to a power. This is useful for capturing non-linear relationships between the feature and the target variable. For example, if you have a feature X, polynomial features could include X^2, X^3, etc.



$[x_1] \rightarrow \begin{bmatrix} x_1 \\ x_1^2 \end{bmatrix}$

Nonlinear SVM Classification



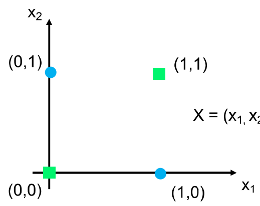
$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2x_2 \\ x_1x_2^2 \\ x_1^3 \\ x_2^3 \end{bmatrix}$

Polynomial features of degree 3
2D to 10D

Nonlinear SVM Classification

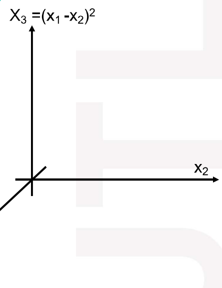
Exercise

Separate the two classes in a higher dimensional space



XOR question

$X = (x_1, x_2) \xrightarrow{\emptyset(x)} Z = (x_1, x_2, (x_1 - x_2)^2)$



Nonlinear SVM Classification

Linear SVM

Minimize: $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i$ (Slack variable)

Subject to: $y_i(\mathbf{w}^{\top}\mathbf{x}_i + b) \geq 1 - \xi_i \quad (i = 1 \sim N)$

$\xi_i \geq 0$

Nonlinear SVM

Minimize: $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i$ (Slack variable)

Subject to: $y_i(\mathbf{w}^{\top}\phi(\mathbf{x})_i + b) \geq 1 - \xi_i \quad (i = 1 \sim N)$

$\xi_i \geq 0$

$\phi(\mathbf{x})_i$ High dimension (infinite)

Kernel Function $= <\emptyset(x_1), \emptyset(x_2)> = \emptyset(x_1)^{\top}\emptyset(x_2)$

Nonlinear SVM: Kernel Method

Linearly separable	A little violations	Strict nonlinearity
Hard-margin SVM	Soft-margin SVM	$\emptyset(x)$+hard-margin -> kernel SVM

Kernel Method: Ideologically, transform low-dimensional non-linear space to high-dimensional linear space, using kernel function.

Kernel Function: Kernel Function $= <\emptyset(x), \emptyset(x')>$, <·> means dot-product
It covers non-linear transformations and an inner product operation on nonlinear transformations.

Kernel Trick: Computationally, avoiding explicitly computing the transformation to another feature space.

- Project input space into a very high-dimensional feature space, may be even infinity
- Problem:
  - Projecting training data in to a high-dimensional space is expensive
  - large number of parameters
- Trick:
  - Compute dot-product between training samples in the projected high-dimensional space without ever projecting.

Nonlinear SVM: Kernel Trick

Input Space: dimension n

High-dimensional Feature Space: dimension N >> n

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$$\phi(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \phi_3(x) \\ \vdots \\ \phi_N(x) \end{bmatrix} \quad N \gg n$$

Expensive operation and requires large memory

$$K(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T\boldsymbol{b} = [a_1 \quad a_2 \quad a_3 \quad \cdots \quad a_n] \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

$$K(\phi(\boldsymbol{a}), \phi(\boldsymbol{b})) = \underbrace{\phi(\boldsymbol{a})^T\phi(\boldsymbol{b})}_{} = [\phi_1(a) \quad \phi_2(a) \quad \phi_3(a) \quad \ldots \quad \phi_N(a)] \begin{bmatrix} \phi_1(b) \\ \phi_2(b) \\ \phi_3(b) \\ \vdots \\ \phi_n(b) \end{bmatrix}$$

$$\phi(\boldsymbol{a})^T\phi(\boldsymbol{b}) = \text{function } (\boldsymbol{a}^T\boldsymbol{b})$$

Kernel Trick

Common kernels:

$$\text{Linear: } K(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\top\mathbf{b}$$
$$\text{Polynomial: } K(\mathbf{a}, \mathbf{b}) = (\gamma\mathbf{a}^\top\mathbf{b} + r)^d$$
$$\text{Gaussian Radial Basis Function: } K(\mathbf{a}, \mathbf{b}) = \exp(-\gamma\|\mathbf{a} - \mathbf{b}\|^2)$$
$$\text{Sigmoid: } K(\mathbf{a}, \mathbf{b}) = \tanh(\gamma\mathbf{a}^\top\mathbf{b} + r)$$

Universal approximator.
Corresponding feature space
$\phi(x)$ is infinite dimensional space

non-linearly separable data

infinite-dimensional space

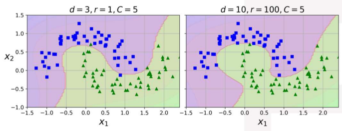
Nonlinear SVM: Polynomial Kernels

```python
# Model training
from sklearn import svm
clf_poly = svm.SVC(kernel='poly', degree=3, coef0=1, C=5)
clf_poly.fit(x_train, y_train)
Visualization(clf_poly, x_train, x_test, y_train, y_test)
```

- `d` is degree of polynomial features
- `r` is polynomial kernel hyperparamter (aka `coef0`)
- `C` is the soft margin hyperparameter

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} \sqrt{r^3} \\ \sqrt{3r^2}x_1 \\ \sqrt{3r^2}x_2 \\ \sqrt{6r}x_1x_2 \\ \sqrt{3r}x_1^2 \\ \sqrt{3r}x_2^2 \\ \sqrt{3}x_1^2x_2 \\ \sqrt{3}x_1x_2^2 \\ x_1^3 \\ x_2^3 \end{bmatrix}$$

Polynomial features of degree 3 2D to 10D with `coef0`



Degree of polynomial `d`:

- Increase if underfitting, decrease if overfitting

`Coef0` influences high-degree terms vs low-degree terms

Use grid search for tuning hyperparameters

- Coarse grid search first
- Followed by finer grid search

SVM Regression

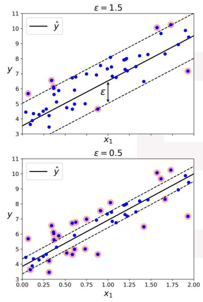
SVM Regression

- SVM algorithm is versatile: Classification & Regression
- Linear and nonlinear regression

SVM Classification
Fitting widest possible "road" between classes with few *on street* violations

SVM Regression
Fitting narrowest possible "road" that captures instances with few *off street* violations

- Hyperparameter ε
- Model is "$\varepsilon - insensitive$" (training instances within margin doesn't affect the model prediction)

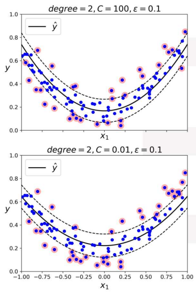



```python
from sklearn.svm import LinearSVR

svm_reg = LinearSVR(epsilon=1.5, random_state=42)
svm_reg.fit(X, y)
```


SVM Regression

- Nonlinearity through kernelized SVM
- Example on a quadratic training set
  - Use SVR class with `kernel = "poly"`
  - Soft margin via hyperparameter C

```python
from sklearn.svm import SVR

svm_poly_reg = SVR(kernel="poly", degree=2, C=100, epsilon=0.1, gamma="scale")
svm_poly_reg.fit(X, y)
```



- `LinearSVC` ⟺ `LinearSVR`
  - No support vector attribute, no kernel
- `SVC` ⟺ `SVR`
  - Support vectors, kernels, slow to train