Support Vector Machines

AI+X
INSTITUTE FOR ARTIFICIAL INTELLIGENCE

Support Vector Machines (SVM)

- PCA and LDA "transform" a dataset in a way that we could reduce the dimensionality of its samples
  - In both cases, we can reconstruct the original information (possibly with a certain loss)
- SVMs are a powerful and flexible class of supervised algorithms
  - SVM, performs discriminative classification
  - Rather than modeling the data, we simply find a line or curve (in two dimensions) or manifold (in multiple dimensions) that divides the classes from each other

Support Vector Machines (SVM)

- A linear discriminative classifier would attempt to draw a straight line separating the two sets of data
- Possibly there are multiple separators with similar performance

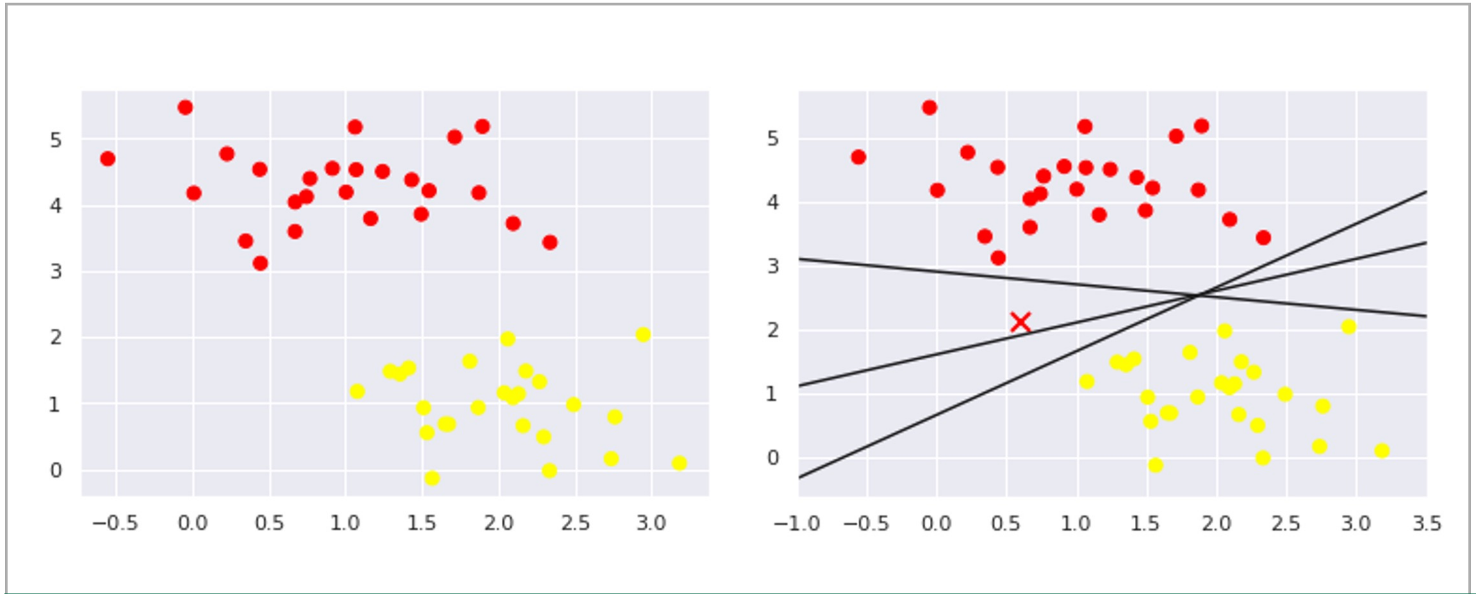

Figure 1. Three very different separators which, nevertheless, perfectly discriminate between these samples.

Support Vector Machines (SVM)

- SVM intuition:
  - Rather than simply drawing a zero-width line between the classes, draw around each line a margin of some width, up to the nearest point.
- SVM picks the line that maximizes this margin

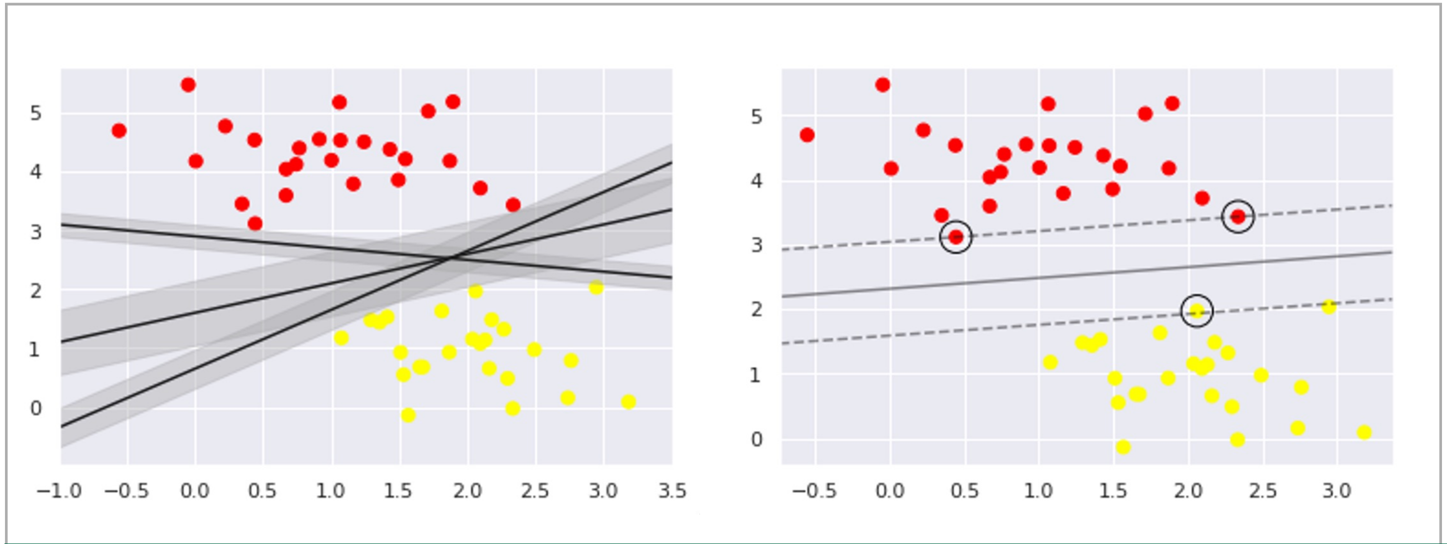

Figure 2. Support vector machines are an example of a maximum margin estimator.

SVM Optimization

Consider a linear classifier for a binary classification problem with labels $\mathbf{y} \in \{-1, 1\}$ and features $\mathbf{x}$. The linear classifier is parameterized with **w,b**:

$$\mathbf{h}_{\mathbf{w},\mathbf{b}}(\mathbf{x}) = \mathbf{g}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x} + \mathbf{b}\right)$$

where **g(z)=1** if **z≥0** and **g(z)=−1** otherwise. To find the values of **w,b** that achieve the maximum margin, we optimize:

$$\mathbf{min}_{\mathbf{w},\mathbf{b}}\ {}^{\mathbf{1}}\!/_{\mathbf{2}}\ \|\mathbf{w}\|^{\mathbf{2}}$$

such that:

$$\mathbf{y}_{\mathbf{i}}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}_{\mathbf{i}} + \mathbf{b}\right) \geq \mathbf{1}, \forall\ \mathbf{i}$$

The above is an optimization problem with a convex quadratic objective and only linear constraints. Its solution gives us the optimal margin classifier. The solution will lead to the inference of a new point **x** using the following equation:

$$\mathbf{g}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x} + \mathbf{b}\right) = \mathbf{g}([\textstyle\sum_{\mathbf{i}}\boldsymbol{\alpha}_{\mathbf{i}}\mathbf{y}_{\mathbf{i}}\mathbf{x}_{\mathbf{i}}]\mathbf{x} + \mathbf{b}) = \mathbf{g}(\textstyle\sum_{\mathbf{i}} \boldsymbol{\alpha}_{\mathbf{i}}\mathbf{y}_{\mathbf{i}}\langle\mathbf{x}_{\mathbf{i}}, \mathbf{x}\rangle + \mathbf{b})$$

with $\boldsymbol{\alpha}_{\mathbf{i}}$'s being zero except for the support vectors.

scikit-learn: SVM implementation

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

```
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
)
```

Parameters::

C : *float, default=1.0*
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*
Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, default=3*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *{'scale', 'auto'} or float, default='scale'*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

Changed in version 0.22: The default value of `gamma` changed from 'auto' to 'scale'.

coef0 : *float, default=0.0*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

Attributes::

class_weight_ : *ndarray of shape (n_classes,)*
Multipliers of parameter C for each class. Computed based on the `class_weight` parameter.

classes_ : *ndarray of shape (n_classes,)*
The classes labels.

n_iter_ : *ndarray of shape (n_classes * (n_classes - 1) // 2,)*
Number of iterations run by the optimization routine to fit the model. The shape of this attribute depends on the number of models optimized which in turn depends on the number of classes.

New in version 1.1.

support_ : *ndarray of shape (n_SV)*
Indices of support vectors.

support_vectors_ : *ndarray of shape (n_SV, n_features)*
Support vectors.

Figure 3. scikit-learn: SVM implementation.

scikit-learn: SVM Implementation

- Methods

`decision_function`(X)	Evaluate the decision function for the samples in X.
`fit`(X, y[, sample_weight])	Fit the SVM model according to the given training data.
`get_params`([deep])	Get parameters for this estimator.
`predict`(X)	Perform classification on samples in X.
`predict_log_proba`(X)	Compute log probabilities of possible outcomes for samples in X.
`predict_proba`(X)	Compute probabilities of possible outcomes for samples in X.
`score`(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
`set_params`(**params)	Set the parameters of this estimator.

Figure 4. scikit-learn: SVM implementation.

SVM Multi-class Classification

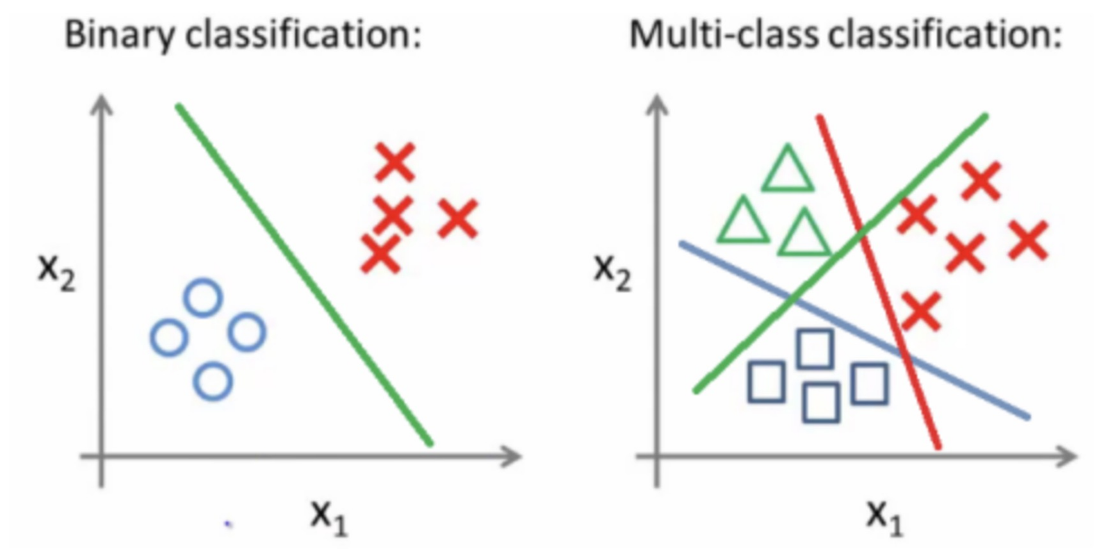

Figure 5. Straightforward application of SVM to multi-class classification problems..

Knowledge Check 1

Which approach can successfully classify the red point as blue or green using the points in the plot on the left as training data?

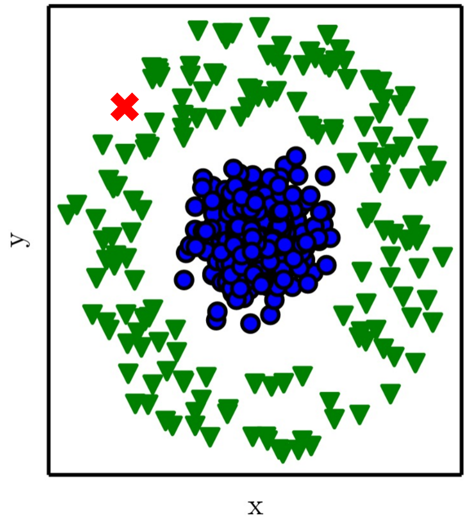

Figure 6. Training data plot.

A K-Means with K=2

B PCA trained using blue points only

C LDA

D SVM with linear kernel

You have reached the end of the lecture.

Image/Figure References

Figure 1. Three very different separators which, nevertheless, perfectly discriminate between these samples.. Source: VanderPlas, Python Data Science Handbook, O'Reilly Media, Inc., 2016.
Figure 2. Support vector machines are an example of a maximum margin estimator. Source: VanderPlas, Python Data Science Handbook, O'Reilly Media, Inc., 2016.
Figure 3. scikit-learn: SVM implementation. Source: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
Figure 4. scikit-learn: SVM implementation. Source: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
Figure 5. Straightforward application of SVM to multi-class classification problems. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 6. Training data plot. Source: Goodfellow, Bengio and Courville, Deep Learning, MIT Press, 2016.