Lecture Title and Date

Supervised Data Mining – SVMs, 2/24

Objectives of the Lecture

- Understand the properties of how support vector machines work as classifiers
- Understand how to perform feature engineering to make points separable using the kernel trick

Key Concepts and Definitions

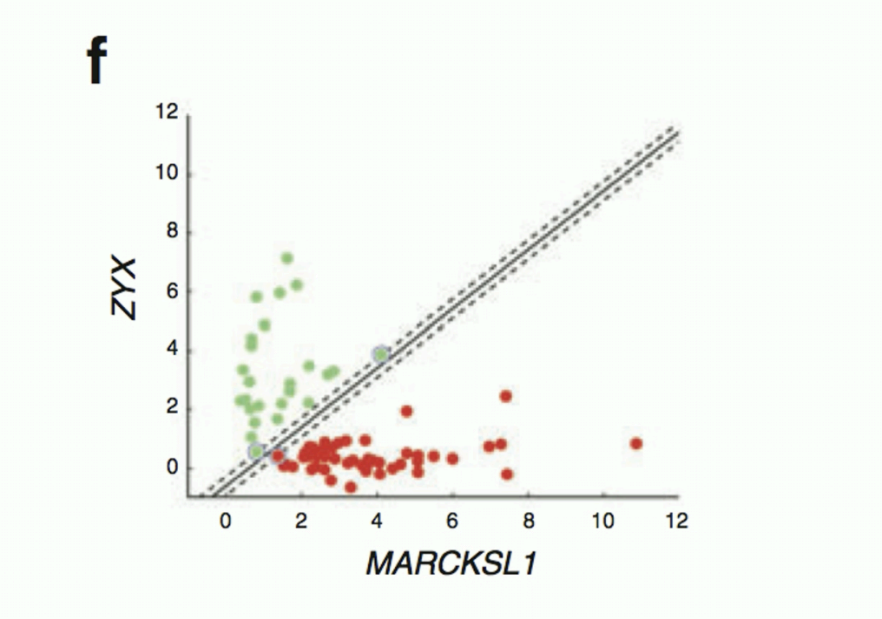
- **Support vector machines** - A classifier which tries to compute the best decision boundary by maximizing the distances between the two closest points along the decision boundary.
- **Hyperplane** - The optimal decision boundary that best separates the points from each other. For SVMs, this means that the hyperplane will maximize the distance closest data points from each class.
- **Margin Width** - The margin width represents the distance between the decision boundary and the nearest data points from either class. Mathematically, this width is calculated as 2/||w||, where w represents the weight vector of the classifier. The SVM algorithm works by maximizing this margin width, which enhances the classifier's ability to generalize to unseen data.
- **Support Vectors** - Support vectors are the critical data points that lie closest to the decision boundary and directly influence its placement. These points "support" the hyperplane by determining its optimal position. The SVM algorithm focuses its computational effort on these boundary cases rather than on the entire dataset, which contributes to its efficiency.
- **Kernel Trick** - Sometimes points are not linearly separable and so we need to do feature mapping to transform the points into a space where the hyperplane can separate the points. Additionally, rather than mapping the points to this higher-dimensional feature space, the kernel trick helps us compute the dot product of the two points in the higher-dimensional feature space which makes it easier to compute the hyperplane needed for SVM.

Common Kernel Functions

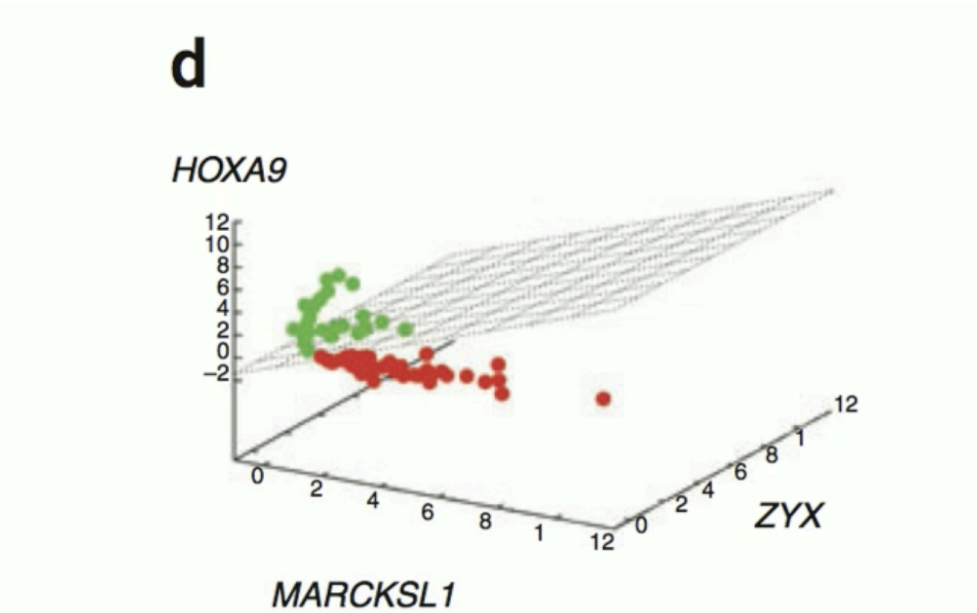
- **Linear Kernel**: $K(x, y) = x \cdot y$
- **Polynomial Kernel**: $K(x, y) = (x \cdot y + c)^d$
- **Radial Basis Function (RBF/Gaussian) Kernel**: $K(x, y) = \exp(-\gamma||x - y||^2)$
- **Sigmoid Kernel**: $K(x, y) = \tanh(\alpha(x \cdot y) + c)$

Main Content/Topics

1. Example of SVM: Leukemia Patient Classification
This case study demonstrates the application of SVMs in medical diagnosis:



- Red dots are patients with ALL: acute lymphoblastic leukemia and the green ones are AML: acute myeloid leukemia.
- We want to build a decision boundary/classifier for determining which leukemia diseases that a patient has and so we make a decision boundary to divide the two.
- For SVMs, the decision boundary is determined by finding the closest point (these are the red and green points on the decision) from each class and then trying to maximize the distance between these points and the hyperplane



- In the case of more than two genes, a line generalizes to a plane or "hyperplane".
- For generality, we refer to them all as "hyperplane"

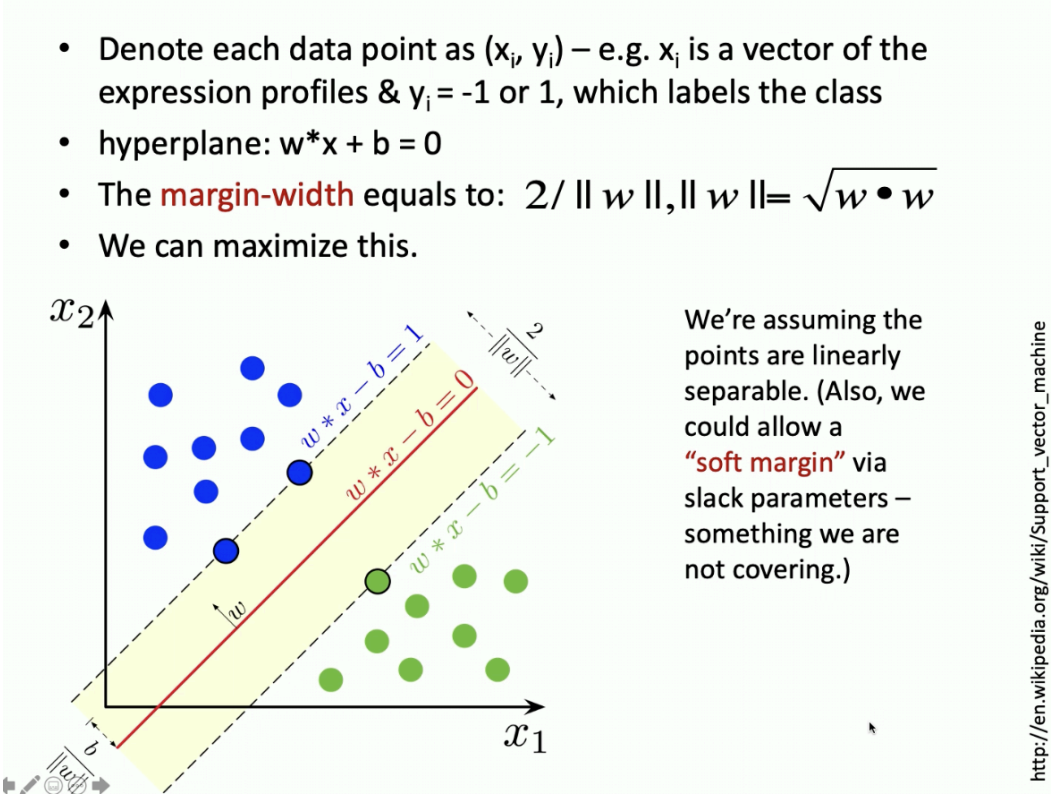
SVM - Hyperplanes

- In cases where there are multiple dimensions (< 2) to the data, we can use a hyperplane instead of a line for dividing the two classes when using SVMs.
- This diagram shows an example of a hyperplane for the classification task when there are three different coordinate features.

SVM - Maximizing Margin Width Between Classes

The optimal SVM hyperplane/decision boundary is created by:

1. Assuming that in some high-dimensional space, the points are linearly separable
2. Finding the hyperplane that maximizes the margin width (the distance between the closest points from either class to the decision boundary)
3. Calculating the margin width as 2/||w||, where w represents the weight vector of the classifier
4. Computing the weights of the classifier to maximize this distance
5. Identifying support vectors (the points that lie exactly on the margin boundaries)



- This slide provides more detail about how SVM hyperplanes/decision boundaries are made. We assume that in some high-dimensional space, the points are linearly separable. The best decision boundary is the one where we maximize the **margin-width** which is the distance between the closest points (the blue and green dots on the dotted lines) from either class to the decision boundary.

- The margin width between the two sides of the decision boundary for a class is computed as 2/||**w**||. Hence, we compute the weights of the classifier, **w**, so that we can maximize the distance this value. This helps give us the best classifier, **w**, by the SVM

definition which maximizes the distance between the two closest points from both classes that are on different sides of the decision boundary.

Mathematical Formulation of SVMs

The SVM optimization problem can be formulated as:

For linear SVMs:

- Minimize: $(1/2)||w||^2 + C\sum\xi_i$
- Subject to: $y_i(w \cdot x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

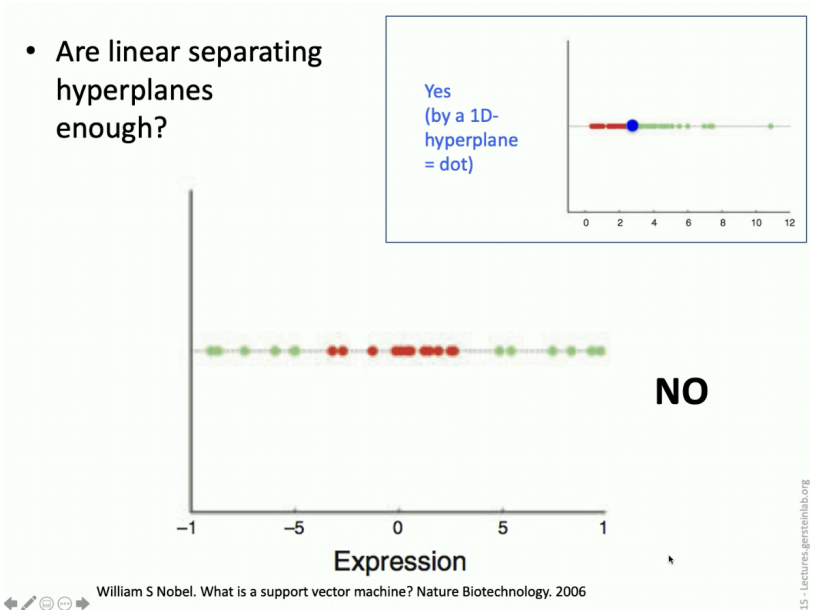
Where:

- w is the weight vector
- b is the bias term
- C is the regularization parameter
- ξ_i are slack variables that allow for misclassifications
- x_i are the input features
- y_i are the class labels (either +1 or -1)

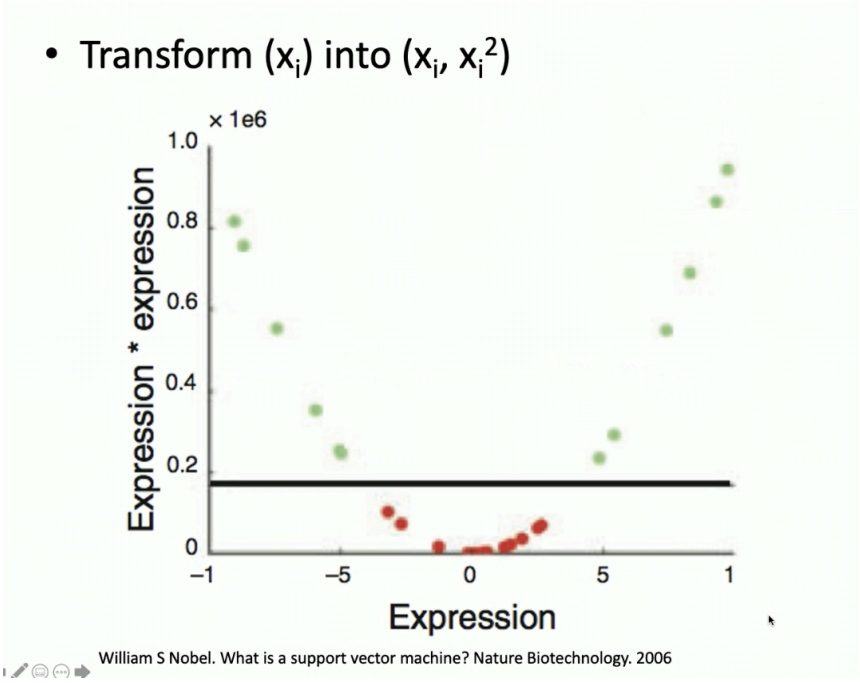
Feature Mapping - Non-linear SVM

When points are not linearly separable in their original dimensional space:

1. There is no single line that can divide the green and red points into separate classes
2. Feature mapping transforms the points into a higher-dimensional space
3. For example, mapping each point x_i to x_i^2 can make the data linearly separable
4. After mapping, a linear separator can effectively divide the classes
5. The original non-linear boundary in the input space corresponds to a linear boundary in the transformed feature space



- Sometimes, the points are not linearly separable by one decision boundary in their current dimensions. This can be demonstrated by this slide which shows that there is no single line for dividing the green and red points such that they are in separate classes.



- To solve this problem, you can perform feature mapping by taking the points and then mapping them to a higher dimensional subspace. In this case, we map each point x_i to the x_i^2. From this, we can make a linear separator dividing the two classes.

Kernel Trick

The kernel trick avoids explicitly computing the coordinates in a higher-dimensional space by:

1. Defining a kernel function K(x, y) that computes the dot product of two points in the feature space
2. Using this kernel function directly in the SVM algorithm instead of computing the explicit mapping
3. Significantly reducing computational complexity, especially for high-dimensional feature spaces
4. Allowing SVMs to handle non-linear classification efficiently

Common Kernels

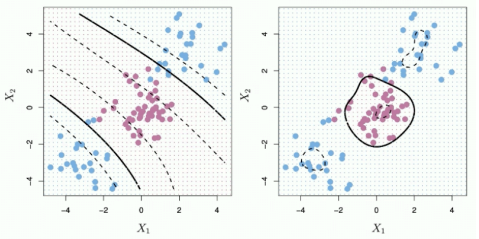


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

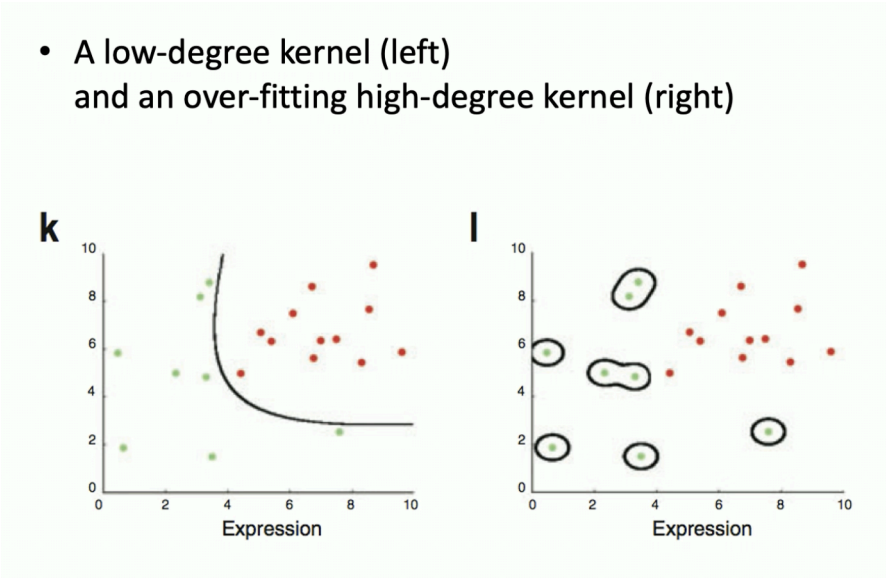
Radial Kernel K

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

Polynomial Kernel K of degree d (d = 1 just gives a linear classifier)

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d.$$

- For mapping to a higher subspace that is separable by a SVM, we can apply a kernel function to map points to a higher space.
- It should be noted that it is sufficient to compute the dot product of the points in higher dimensional space and we don't need to compute the value of each point in the higher-dimensional space when determining the SVM.
- This figure is showing two different kernel functions which are applied to the points in the dataset. We can use the kernel function to identify a linear separator in higher-dimensional space and then map the decision-boundary back to the original subspace, which will be non-linear.



A major drawback of using kernel functions is that they can overfit to the data. This is especially true when using higher degree polynomials. As shown in figure K, a lower dimensional kernel is properly able to separate the data points onto two separate sides. However, when using a higher dimensional kernel, figure I shows that we can overfit to the data when making decision boundaries that separate within the same group when projecting back to the original subspace. Hence, it is important to choose a kernel function that does not overfit to the dataset by making very specific decision boundaries over certain points.

SVM Advantages and Limitations

Advantages:

- Effective in high-dimensional spaces
- Memory efficient as it uses only a subset of training points (support vectors)
- Versatile through different kernel functions
- Robust against overfitting when properly tuned

Limitations:

- Computationally intensive for large datasets
- Requires careful selection of kernel and regularization parameters
- Not directly suitable for multi-class classification (requires strategies like one-vs-rest)
- Performance degrades with overlapping classes or noisy data

Discussion/Comments

I would suggest adding "Pattern Recognition and Machine Learning" by Christopher Bishop (Chapter 7) as an additional reference. Bishop provides clear explanations of the probabilistic interpretation of SVMs and offers valuable insights on kernel selection strategies for different types of data, which complements the current materials nicely.

For students struggling with the mathematical foundations, I recommend: Andrew Ng's machine learning course materials on SVMs, which break down the concepts more gradually with helpful visualizations and simplified mathematical notation.

List all suggested reading here and please answer:

Cristianini, Nello, and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

References ISL/ESL (if any)

Chapter 9 - 9.4: Covers SVMs and the math behind them from ISL book

Other Suggest references for many of the key concepts

- For leaning more about the math behind how the hyperplanes in SVMs are compute along with how the kernel functions can help make the optimal SVM: https://www.geeksforgeeks.org/support-vector-machine-algorithm/#