CHEME 5820: Machine Learning for Engineers **Spring 2025**

Lecture 5c: Support Vector Machines (SVMs)

Lecturer: Jeffrey Varner

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications.*

The key concepts covered in this lecture include:

- A support vector machine is a supervised machine learning algorithm that finds an optimal hyperplane in an N-dimensional space to classify (binary) data points distinctly, maximizing the *margin* between different classes. The *margin* in a support vector machine is defined as the distance from the separating hyperplane to the closest data points of either class.
- A hard margin support vector machine is a binary linear classifier that finds the optimal hyperplane to separate two classes of data points with the maximum possible *margin*, allowing no misclassifications and requiring the data to be linearly separable.
- A soft margin support vector machine is a variant of the SVM algorithm that allows for some misclassification of training data points, enabling it to handle non-linearly separable datasets and reduce overfitting by finding a balance between maximizing the decision boundary margin and minimizing classification errors.

1 Introduction

In the previous lectures, we've discussed several binary classification algorithms, including the perceptron, logistic regression, K-nearest neighbors, and the concept of Kernel functions and the kerner trick. In this lecture, we will conclude our dicussion (for now) of classification algorithms by introducing support vector machines (SVMs), a powerful and versatile machine learning algorithm that can be used for both classification and regression tasks. SVMs are particularly well-suited for binary classification problems, where the goal is to separate data points into two classes using a hyperplane, e.g., like the perceptron. However, SVMs are based on the concept of finding an *optimal seperating hyperplane* (not just some seperating hyperplane) that maximizes the margin between the two classes.

2 Support Vector Machines (SVMs)

Support vector machines (SVMs) are a class of supervised learning algorithms that can be used for both classification and regression tasks, published by Cortes and Vapnik (1) while they worked at AT&T Bell Labs. In many ways, SVMs are similar to the perceptron and logistic regression, as they are also based on the concept (in the simplest case) of finding a hyperplane that separates data points into different classes. Suppose, we have dataset $\mathcal{D} = \{(\hat{\mathbf{x}}_i, y_i) \mid i = 1, 2, \ldots, n\}$, where $\hat{\mathbf{x}}_i \in \mathbb{R}^p$ is an *augmented* feature vector (m features with additional 1 last entry to model the bias) and $y_i \in \{-1, 1\}$ is the corresponding class label. Then, the goal of SVMs is to find the hyperplane $\mathcal{H} = \{\hat{\mathbf{x}} \mid \langle \hat{\mathbf{x}}, \theta \rangle = 0\}$ that separates the data points into two classes (those points above the hyperplane, and those points below the hyperplane), where $\theta \in \mathbb{R}^p$ ($p = m + 1$) is the normal vector to the hyperplane, or alternatively, the parameters of the model that we need to estimate. So far, this is similar to the perceptron and logistic regression, but the key difference is that

SVMs aim to find the *optimal* hyperplane in some sense. Let's explore the notion of an *optimal hyperplane* in more detail.

2.1 Maximum Margin Classifier

Suppose we have found a hyperplane $\mathcal{H}$ that separates the data points into two classes. Then, the margin of the hyperplane is defined as the distance γ between the hyperplane and the closest data point from either class. Thus, the margin is a measure of how well the hyperplane separates the two classes, and the goal of a maximizing margin SVM classifier is to find the hyperplane that maximizes the margin. Let's develop a model for the margin of the seperating hyperplane.

Consider some feature vector $\hat{\mathbf{x}} \in \mathbb{R}^m$. Let $\mathbf{d}$ denote the vector from the hyperplane $\mathcal{H}$ to the feature vector $\hat{\mathbf{x}}$. Finally, let the point $\mathbf{p}$ be the projection of $\hat{\mathbf{x}}$ onto the hyperplane $\mathcal{H}$. Because the vector $\mathbf{d}$ is orthogonal to the hyperplane $\mathcal{H}$, we can write $\mathbf{d} = \hat{\mathbf{x}} - \mathbf{p}$. Futher, the vector $\mathbf{d}$ can be written as some scalar multiple of the normal vector θ, i.e., $\mathbf{d} = \lambda\theta$. Then we can find the value of λ by taking the dot product of $\mathbf{d}$ with the normal vector θ:

$$
\begin{aligned}
\hat{\mathbf{p}} &= \hat{\mathbf{x}} - \mathbf{d} && \mid \text{take dot product with } \theta \\
\langle \hat{\mathbf{p}}, \theta \rangle &= \langle \hat{\mathbf{x}}, \theta \rangle - \langle \mathbf{d}, \theta \rangle = 0 && \mid \text{substitute } \mathbf{d} = \lambda\theta \\
\langle \hat{\mathbf{p}}, \theta \rangle &= \langle \hat{\mathbf{x}}, \theta \rangle - \lambda \langle \theta, \theta \rangle = 0 && \mid \text{solve for } \lambda \\
\lambda &= \frac{\langle \hat{\mathbf{x}}, \theta \rangle}{\langle \theta, \theta \rangle}
\end{aligned}
$$

We can now find the length of the vector $\mathbf{d}$ by computing $||\mathbf{d}||_2$:

$$
\begin{aligned}
||\mathbf{d}||_2 &= \sqrt{\mathbf{d}^\top \mathbf{d}} && \mid \text{substitute } \mathbf{d} = \lambda\theta \\
&= \sqrt{\lambda^2 \theta^\top \theta} && \mid \text{substitute } \lambda = \frac{\langle \hat{\mathbf{x}}, \theta \rangle}{\langle \theta, \theta \rangle} \\
&= \sqrt{\frac{\langle \hat{\mathbf{x}}, \theta \rangle^2}{\langle \theta, \theta \rangle^2} \theta^\top \theta} && \mid \text{substitute } \langle \theta, \theta \rangle = \theta^\top \theta \text{ and simplify} \\
&= \frac{\langle \hat{\mathbf{x}}, \theta \rangle}{||\theta||_2} && \mid \text{where } ||\theta||_2 = \sqrt{\langle \theta, \theta \rangle}
\end{aligned}
$$

The length of the distance vector $\mathbf{d}$ gives us the distance from a feature vector $\hat{\mathbf{x}}$ to the hyperplane $\mathcal{H}$. Thus, we can define the margin γ_θ of the hyperplane $\mathcal{H}$ as the distance between the hyperplane and the closest data point (Defn. 1).

Definition 1 (Margin)**.** *The margin γ_θ of a hyperplane $\mathcal{H}$ is given by the distance between the hyperplane and the closest data point:*

$$
\gamma_\theta = \min_i \left\{ \frac{| \langle \hat{\mathbf{x}}_i, \theta \rangle |}{||\theta||_2} \right\}
$$

where we use the absolute value to account for the fact that the distance can be positive or negative, i.e., the data point can be on either side of the hyperplane.

On interesting feature of the margin is that it is scale-invariant, i.e., the margin is the same regardless of the scale of the normal vector θ (Lemma 1).

Lemma 1 (Scale Invariance)**.** *The margin γ_θ is scale-invariant, i.e., the margin is the same regardless of the scale of the normal vector θ. To see that the margin is scale-invariant, consider the scaled normal vector $\theta' = \alpha\theta$, where the scalar $\alpha \neq 0$. Then, we can show that the margin of the scaled normal vector θ' is the same as the margin of the original normal vector θ:*

$$\begin{aligned}
\gamma_{\theta'} &= \min_i \left\{ \frac{|\langle \hat{\mathbf{x}}_i, \theta' \rangle|}{||\theta'||_2} \right\} \quad | \text{ substitute } \theta' = \alpha\theta \\
&= \min_i \left\{ \frac{|\langle \hat{\mathbf{x}}_i, \alpha\theta \rangle|}{||\alpha\theta||_2} \right\} \quad | \text{ substitute } ||\alpha\theta||_2 = |\alpha|\,||\theta||_2 \\
&= \min_i \left\{ \frac{|\alpha \langle \hat{\mathbf{x}}_i, \theta \rangle|}{|\alpha|\,||\theta||_2} \right\} \quad | \text{ cancel out } \alpha \\
&= \min_i \left\{ \frac{|\langle \hat{\mathbf{x}}_i, \theta \rangle|}{||\theta||_2} \right\} = \gamma_\theta
\end{aligned}$$

Practically this means the margin is only dependent on the direction of the normal vector θ, not its magnitude. Thus, we can set the scale to be whatever is most convenient.

Now that have a model for the margin of the hyperplane, we can define the problem of finding the optimal hyperplane as an optimization problem. Ideally, we would like to estimate the parameters θ of the hyperplane that maximize the margin γ_θ, i.e., the distance between the hyperplane and the closest data point is maxzimized:

$$\max_\theta \gamma_\theta \quad \text{subject to} \quad y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 0 \quad \forall i$$

We can substitute the definition of the margin γ_θ into the optimization problem, which gives as a new objective function:

$$\max_\theta \left[\min_i \left\{ \frac{|\langle \hat{\mathbf{x}}_i, \theta \rangle|}{||\theta||_2} \right\} \right] \quad \text{subject to} \quad y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 0 \quad \forall i$$

However, we can factor out the norm of the normal vector θ from the objective function, which simplifies the optimization problem:

$$\max_\theta \frac{1}{||\theta||_2} \left[\min_i \{ |\langle \hat{\mathbf{x}}_i, \theta \rangle| \} \right] \quad \text{subject to} \quad y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 0 \quad \forall i$$

Finally, we can use the scale invariance of the margin and the hyperplane, and the fact that maximing the inverse of a function is equivalent to minimizing the function, to simplify the optimization problem:

$$\begin{aligned}
\text{objective} \qquad & \min_\theta ||\theta||_2^2 \\
\text{subject to} \qquad & y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 0 \quad \forall i \\
& \min_i \{ |\langle \hat{\mathbf{x}}_i, \theta \rangle| \} = 1
\end{aligned}$$

However, the last constraint is tricky to handle. We can simplify the optimization problem futher and get rid of the last constraint by modifying the second set of constraints (Defn. 2):

Definition 2 (Maximim Hard Margin Classifier)**.** *The maximum hard margin classifier is a solution*

of the quadratic programming problem:

$$\begin{aligned} \textit{objective} \quad & \frac{1}{2} \min_{\theta} ||\theta||_2^2 \\ \textit{subject to} \quad & y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 1 \quad \forall i \end{aligned}$$

where $\theta \in \mathbb{R}^p$ denote the unknown parameters that we are trying to estimate, $\hat{\mathbf{x}}_i \in \mathbb{R}^p$ are the augmented feature vectors, $y_i \in \{-1, 1\}$ are the class labels, and $p = m + 1$ is the number of parameters, where m is the number of features. The index i runs over the training examples, i.e., there is one constraint per training example.

The maximum margin classifier problem is a quadratic optimization problem with linear constraints, thus, it can be solved using a variety of optimization approaches. However, a key practical takeaway is that the maximum hard margin classifier will only exist, i.e., find an optimal hyperplane, if the data points are linearly separable. Finally, support vector machines get the their name from the so-called support vectors (Defn. 3)

Definition 3 (Support Vectors)**.** *The special case where $y_i \langle \hat{\mathbf{x}}_i, \theta \rangle = 1$ are called the support vectors. For the maximum margin classifier, the support vectors are the data points that are closest to the hyperplane, i.e., the data points that define the margin.*

2.2 Soft Margin Classifier

Suppose there exists points in our dataset $\mathcal{D}$ that are not linearly separable, i.e., there is no hyperplane that can perfectly separate the two classes. Just like we did in our perceptron implementation, we can allow the support vector machine to make some mistakes by introducing a slack variable ξ_i for each data point. Let the slack variable ξ_i represent the cost of a mistake (in some sense) for the i-th data point. Then, we can modify the constraints and the objective function in the maximum margin classifier problem to allow for some slack, i.e., the constraints are relaxed (Defn. 4):

Definition 4 (Maximim Soft Margin Classifier)**.** *The soft margin classifier problem is defined as the quadratic programming problem:*

$$\begin{aligned} \textit{objective} \quad & \frac{1}{2} \min_{\theta} ||\theta||_2^2 + C \sum_{i=1}^{n} \xi_i \\ \textit{subject to} \quad & y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

where $C \geq 0$ is a user-adjustable parameter that controls the trade-off between maximizing the margin and minimizing the slack variables. If $C \gg 1$ the classifier will behave like the maximum margin classifier, and if $C \ll 1$ the classifier will allow more slack (mistakes).

We could solve the soft margin classifier problem (directly) using the same optimization techniques as the maximum margin classifier problem. However, we can also reformulate it as an unconstrained problem by introducing the `hinge loss` function. Let's consider what value the slack variables ξ_i can take on (assuming $C > 0$) where the classifier is correct, and when it makes a mistake. For data points that are correctly classified, the slack variable ξ_i will be zero, i.e., $y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 1$ (the data point is on the correct side of the hyperplane, the SVM didn't make a mistake). However, for data points that are misclassified, the slack variable ξ_i will be positive, i.e., there is a penalty associated with making a mistake. Putting these two cases

together gives the possible values of the slack variables ξ_i:

$$\xi_i = \begin{cases} 0 & \text{if } y_i \langle \hat{\mathbf{x}}_i, \theta \rangle \geq 1 \quad \text{correct} \\ 1 - y_i \langle \hat{\mathbf{x}}_i, \theta \rangle & \text{if } y_i \langle \hat{\mathbf{x}}_i, \theta \rangle < 1 \quad \text{mistake} \end{cases}$$

These two cases can be rewritten in a more convient form:

$$\xi_i = \max\{0, 1 - y_i \langle \hat{\mathbf{x}}_i, \theta \rangle\}$$

Now we have an expression for the slack variables ξ_i in terms of the inner product between the augmented feature vectors and the parameter vector. We can substitute this expression into the objective function of the soft margin classifier problem, which gives us the `hinge loss` function (Defn. 5):

Definition 5. *The optimal soft margin classifier problem can be reformulated as an unconstrained optimization problem using the* ***`hinge loss`*** *function:*

$$\min_{\theta} ||\theta||_2^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i \langle \hat{\mathbf{x}}_i, \theta \rangle\}$$

where $C > 0$ is a user-adjustable parameter that controls the trade-off between maximizing the margin and minimizing the slack variables. The hinge loss is not differentiable everywhere, i.e., the gradient is not defined at tie points. Thus, we need to fancier gradients, or alternative techniques such as simulated annealing (or another huerstic method) to solve the problem.

Finally, while we are not going to discuss the underlying theory in this lecture, SVMs can be extended to handle non-linearly separable data using the kernel trick. This involves solving the dual optimization problem, which is beyond the scope of this lecture. For more details on the kernel trick and the dual optimization problem, see the CS4780 (2018 fall) lecture 14 notes.

3 Summary and Conclusions

In this lecture, we introduced support vector machines (SVMs), a powerful and versatile machine learning algorithm that can be used for both classification and regression tasks. SVMs are particularly well-suited for binary classification problems, where the goal is to separate data points into two classes using an optimal hyperplane. In this lecture, we discussed the concept of the maximum margin classifier, which aims to find the hyperplane that maximizes the margin between the two classes. However, the maximum margin classifier only exists if the data points are linearly separable. In cases where the data points are not linearly separable, we can use the soft margin classifier, which allows for some slack (mistakes) in the constraints. Finally, we discussed the `hinge loss` function, which can be used to reformulate the soft margin classifier problem as an unconstrained optimization problem. While not covered in this lecture, SVMs can be extended to handle non-linearly separable data using the kernel trick, which allows us to map the data points into a higher-dimensional space where they are linearly separable.

References

1. Cortes C, Vapnik V. Support-vector networks. Machine Learning. 1995;20(3):273–297. doi:10.1007/BF00994018.