A survey of some Machine Learning models

Chris Cornwell

April 1, 2025

Outline

Setup

Similar to a logistic model, a **support vector machine** is a model for binary classification, using a hyperplane, of the form $\{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{w} \cdot \mathbf{x} + b = 0\}$, as decision boundary.

However, the optimization goal is different.

[1]The multiple $\frac{1}{2}$ is non-essential here, but the convention is to include it.

[2]Recall, we pointed out how this is possible when the data is linearly separable.

Setup

Similar to a logistic model, a **support vector machine** is a model for binary classification, using a hyperplane, of the form $\{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{w} \cdot \mathbf{x} + b = 0\}$, as decision boundary.

However, the optimization goal is different.

Given sample data $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the goal is to minimize the value of $\frac{1}{2}|\mathbf{w}|^2$, the vector norm[1], subject to the condition that for all $1 \leq i \leq n$, $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ is satisfied.[2]

To work with a data set that is not linearly separable, one introduces so-called "slack variables" $\xi_i \geq 0$, $i = 1, \ldots, n$ into the inequalities. They change to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$.

The reason for wanting to minimize $\frac{1}{2}|\mathbf{w}|^2$?

- Supposing no $\mathbf{x}_i$ passes through hyperplane with parameters $\mathbf{w}, b$, we can scale both the normal vector and b so that $\min_i |\mathbf{w} \cdot \mathbf{x}_i + b| = 1$.
- The distance from any $\mathbf{x} \in \mathbb{R}^d$ to the hyperplane is $\frac{|\mathbf{w} \cdot \mathbf{x} + b|}{|\mathbf{w}|}$. So, if $\mathbf{x}_i \in \mathcal{S}$ is such that $|\mathbf{w} \cdot \mathbf{x}_i + b| = 1$, then its distance to decision boundary is $\rho = \frac{1}{|\mathbf{w}|}$.
- Want to maximize distance to decision boundary, so want to minimize $|\mathbf{w}|$.

[1] The multiple $\frac{1}{2}$ is non-essential here, but the convention is to include it.

[2] Recall, we pointed out how this is possible when the data is linearly separable.

Constrained minimization and SVM

We can understand minimizing $\frac{1}{2}|\mathbf{w}|^2$ subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ with a Lagrangian. (Method of Lagrange multipliers; see Section 7.2 in the Mathematics for Machine Learning book.)

Constrained minimization and SVM

We can understand minimizing $\frac{1}{2}|\mathbf{w}|^2$ subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ with a Lagrangian. (Method of Lagrange multipliers; see Section 7.2 in the Mathematics for Machine Learning book.)
For $\underline{\alpha} = (\alpha_1, \ldots, \alpha_n)$, with $\alpha_i \in \mathbb{R}$, Lagrangian is

$$L(\mathbf{w}, b, \underline{\alpha}) = \frac{1}{2}|\mathbf{w}|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right).$$

Constrained minimization and SVM

We can understand minimizing $\frac{1}{2}|\mathbf{w}|^2$ subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ with a Lagrangian. (Method of Lagrange multipliers; see Section 7.2 in the Mathematics for Machine Learning book.)

For $\underline{\alpha} = (\alpha_1, \ldots, \alpha_n)$, with $\alpha_i \in \mathbb{R}$, Lagrangian is

$$L(\mathbf{w}, b, \underline{\alpha}) = \frac{1}{2}|\mathbf{w}|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right).$$

It is minimized when

$$\begin{aligned}
\nabla_{\mathbf{w}} L = 0 \quad &\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i; \\
\nabla_b L = 0 \quad &\Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0; \\
\alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right) = 0 \quad &\Rightarrow \quad \alpha_i = 0 \quad \text{OR} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1.
\end{aligned}$$

Constrained minimization and SVM

We can understand minimizing $\frac{1}{2}|\mathbf{w}|^2$ subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ with a Lagrangian. (Method of Lagrange multipliers; see Section 7.2 in the Mathematics for Machine Learning book.)

For $\underline{\alpha} = (\alpha_1, \ldots, \alpha_n)$, with $\alpha_i \in \mathbb{R}$, Lagrangian is

$$L(\mathbf{w}, b, \underline{\alpha}) = \frac{1}{2}|\mathbf{w}|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right).$$

It is minimized when

$$\begin{aligned}
\nabla_{\mathbf{w}} L = 0 \quad &\Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i; \\
\nabla_b L = 0 \quad &\Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0; \\
\alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right) = 0 \quad &\Rightarrow \quad \alpha_i = 0 \quad \text{OR} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1.
\end{aligned}$$

Support vectors are those $\mathbf{x}_i$ for which $\alpha_i \neq 0$, and so $\mathbf{w} \cdot \mathbf{x}_i + b = \pm 1$.

Lagrangian Duality

Something interesting happens when we convert the previous Lagrangian optimization problem into its "Lagrangian dual problem." This means that we take the minimum solution for $\mathbf{w}$, put it into $L(\mathbf{w}, b, \underline{\alpha})$ and want multipliers $\alpha_i \geq 0$ that *maximize* the value of this. That is, maximize

$$\frac{1}{2}\left|\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j \right) \cdot \mathbf{x}_i + y_i b - 1 \right).$$

Lagrangian Duality

Something interesting happens when we convert the previous Lagrangian optimization problem into its "Lagrangian dual problem." This means that we take the minimum solution for $\mathbf{w}$, put it into $L(\mathbf{w}, b, \underline{\alpha})$ and want multipliers $\alpha_i \geq 0$ that *maximize* the value of this. That is, maximize

$$\frac{1}{2}\left|\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j \right) \cdot \mathbf{x}_i + y_i b - 1 \right).$$

Rearranged, you can rewrite it:

$$\max_{\underline{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$.

Lagrangian Duality

Something interesting happens when we convert the previous Lagrangian optimization problem into its “Lagrangian dual problem.” This means that we take the minimum solution for $\mathbf{w}$, put it into $L(\mathbf{w}, b, \underline{\alpha})$ and want multipliers $\alpha_i \geq 0$ that *maximize* the value of this. That is, maximize

$$\frac{1}{2}\left|\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i \left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j \right) \cdot \mathbf{x}_i + y_i b - 1 \right).$$

Rearranged, you can rewrite it:

$$\max_{\underline{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$.
This optimization problem only depends on knowing $\mathbf{x}_i \cdot \mathbf{x}_j$ for each (i, j), and this leads to what are called **kernel methods** that are very computationally efficient and allow one to use SVM models that have non-linear decision boundaries.

SVMs via Gradient Descent

An alternative for optimizing an SVM classifier is to do so with a loss function. The loss function has a fair amount of similarity to the log-loss function we used in logistic regression; however, the per-example losses use a piecewise linear function.

SVMs via Gradient Descent

An alternative for optimizing an SVM classifier is to do so with a loss function. The loss function has a fair amount of similarity to the log-loss function we used in logistic regression; however, the per-example losses use a piecewise linear function.

When $y_i = 1$ then, writing $z_i = \mathbf{w} \cdot \mathbf{x}_i + b$, the per-example loss is $C \max(1 - z_i, 0)$ for some constant C. Call this $C\text{cost}_1(z_i)$. When $y_i = -1$ (and so $\tilde{y}_i = 0$) then the per-example loss is $C\text{cost}_0(z_i) = C \max(1 + z_i, 0)$.

SVMs via Gradient Descent

An alternative for optimizing an SVM classifier is to do so with a loss function. The loss function has a fair amount of similarity to the log-loss function we used in logistic regression; however, the per-example losses use a piecewise linear function.

When $y_i = 1$ then, writing $z_i = \mathbf{w} \cdot \mathbf{x}_i + b$, the per-example loss is $C \max(1 - z_i, 0)$ for some constant C. Call this $C\mathrm{cost}_1(z_i)$. When $y_i = -1$ (and so $\tilde{y}_i = 0$) then the per-example loss is $C\mathrm{cost}_0(z_i) = C \max(1 + z_i, 0)$. However, we also include the norm of $\mathbf{w}$ in the loss:

$$\mathcal{L}_{\mathcal{S}}(\mathbf{w}, b) = \frac{1}{2}|\mathbf{w}|^2 + \frac{1}{n}\sum_{i=1}^{n} C\left(\tilde{y}_i \mathrm{cost}_1(\mathbf{w} \cdot \mathbf{x}_i + b) + (1 - \tilde{y}_i)\mathrm{cost}_0(\mathbf{w} \cdot \mathbf{x}_i + b)\right).$$