Modern Machine Learning — Computing the Support Vector Classifier

Kenneth E. Barner
Department of Electrical and Computer Engineering
University of Delaware

Support Vector Machines

Perfectly Separable Case: Recall that in this case, the optimization can be cast as:

$$\min_{\beta_0,\boldsymbol{\beta}} \|\boldsymbol{\beta}\| \text{ subject to } y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) \geq 1, \forall i$$

Note that the constant can be arbitrarily scaled, so the optimization can be equivalently expressed as:

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 \text{ subject to } y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) \geq 1, \forall i$$

Lagrangian Approach: Consider a general optimization problem of the form

$$\min_x f_0(x)$$
$$\text{subject to} \begin{cases} f_i(x) \leq 0,\ i = 1,2,\dots,n \\ h_i(x) = 0, i = 1,2,\dots,p \end{cases}$$

This optimization problem can be recast in a Lagrangian formulation as:

$$L_p = f_0(x) + \sum_{i=1}^{n} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

which is to be minimized with respect to x and where the constraints are in the form of additive penalties

Support Vector Machines

For the perfectly separable case,

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 \text{ subject to } y_i\left(\mathbf{x}_i^T\boldsymbol{\beta}+\beta_0\right) \geq 1, \forall i$$

The equivalent Lagrange (primal) function to be minimized is:

$$L_p = \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i\left(x_i^T\boldsymbol{\beta}+\beta_0\right)-1\right] \qquad (*)$$

Setting the derivatives to zero yields: with respect to $\boldsymbol{\beta}$

$$\frac{\partial L_p}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i\, y_i x_i^T = 0 \quad \Longrightarrow \quad \boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i\, y_i x_i^T$$

And with respect to β_0

$$\frac{\partial L_p}{\partial \beta_0} = \sum_{i=1}^{n} \alpha_i\, y_i = 0$$

Support Vector Machines

Substituting these results into L_p, equation $(*)$ in previous slide, yields the Wolfe dual

$$L_D = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{k=1}^{n} \alpha_i \, \alpha_k \, y_i y_k \boldsymbol{x}_i^T \boldsymbol{x}_k$$

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

Observations:

Optimization objective is to maximize L_D subject to the constraints

Optimization problem is convex

Solution must satisfy the Karush-Kuhn-Tucker conditions:

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i \, y_i x_i^T, \qquad \sum_{i=1}^{n} \alpha_i \, y_i = 0, \qquad \alpha_i \geq 0 \; \forall i$$

and

$$\alpha_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) - 1 \right] = 0 \; \forall i$$

Support Vector Machines

Aside: Recall our previous problem development

The $\|\boldsymbol{\beta}\| = 1$ normalization was incorporated as follows:

$$\frac{1}{\|\boldsymbol{\beta}\|} y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) \geq M$$
$$\Longrightarrow y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) \geq M\|\boldsymbol{\beta}\|$$

Rescaling $(\beta_0, \boldsymbol{\beta})$ so that $\|\boldsymbol{\beta}\| = \frac{1}{M}$ yielded the optimization:

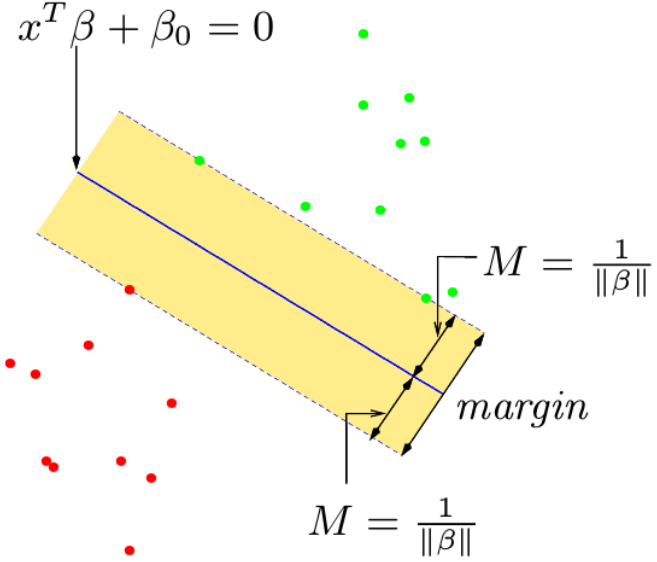
$$\min_{\beta_0, \boldsymbol{\beta} \in \mathbb{R}^p} \|\boldsymbol{\beta}\|$$

subject to

$$y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) \geq 1, i = 1, \ldots, n$$

Recollections & Observations:

- $\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0$ is the signed a distance between $\boldsymbol{x_i}$ and the decision boundary, $\boldsymbol{x}^T\boldsymbol{\beta} + \beta_0 = 0$
- $y_i \in \{-1, 1\}$ are the class labels
- $y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) \geq 1 \Longrightarrow \boldsymbol{x_i}$ has a distance to the boundary $\geq M$
- $y_i\left(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0\right) = 1 \Longrightarrow \boldsymbol{x_i}$ is on the margin boundary, i.e., has distance M



Support Vector Machines

Resume Optimization Development: Recall the Wolfe dual

$$L_D = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i \alpha_k y_i y_k \boldsymbol{x}_i^T \boldsymbol{x}_k$$

that must be maximized and satisfy the Karush-Kuhn-Tucker conditions:

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^T, \qquad \sum_{i=1}^{n} \alpha_i y_i = 0, \qquad \alpha_i \geq 0 \; \forall i$$

and

$$\alpha_i \left[y_i \left(\boldsymbol{x}_i^T \boldsymbol{\beta} + \beta_0 \right) - 1 \right] = 0 \; \forall i$$

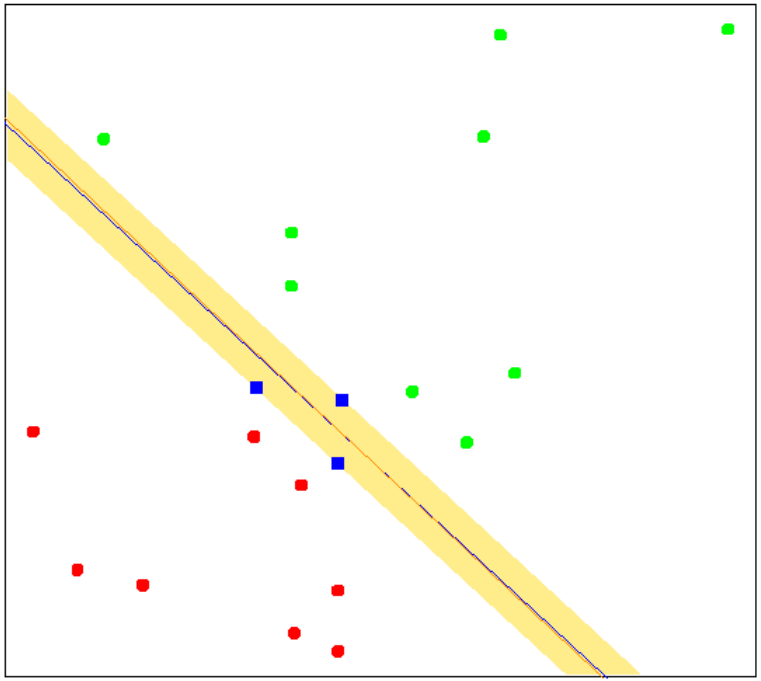
Observations:

- If $\alpha_i > 0$, then $y_i \left(x_i^T \boldsymbol{\beta} + \beta_0 \right) = 1 \Longrightarrow \boldsymbol{x}_i$ is on the boundary of the slab (has distance M to the decision boundary)
- If $y_i \left(x_i^T \boldsymbol{\beta} + \beta_0 \right) > 1$, then $\boldsymbol{x}_i$ is not on the boundary of the slab (has distance $> M$ to the decision boundary), and $\alpha_i = 0$

Interpretation: optimal $\boldsymbol{\beta}$,

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i^T$$

is defined in terms of a linear combination of support vectors $\boldsymbol{x}_i$— those points on the boundary of the slab (having $\alpha_i > 0$).



Binary Class Example: Optimal SVM hyperplane (blue line) and logistic regression determined hyperplane (redline) are shown. Yellow region shows the maximum margin separating the two classes. Data yields three support vectors, shown in blue.

Support Vector Machines

Non-Separable Case: Recall that the non-separable case yields:

$$\min\|\boldsymbol{\beta}\| \text{ subject to } \begin{cases} y_i(\mathbf{x}_i^T\boldsymbol{\beta}+\beta_0) \geq 1-\xi_i, \forall i \\ \xi_i \geq 0, \sum_{i=1}^{n}\xi_i \leq \text{constant} \end{cases}$$

Similarly to the separable case, this can be expressed as:

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to } \xi_i \geq 0, \text{and } y_i(\boldsymbol{x}_i^T\boldsymbol{\beta}+\beta_0) \geq 1-\xi_i, \forall i$$

Note that $C=\infty$ corresponds to the separable case.

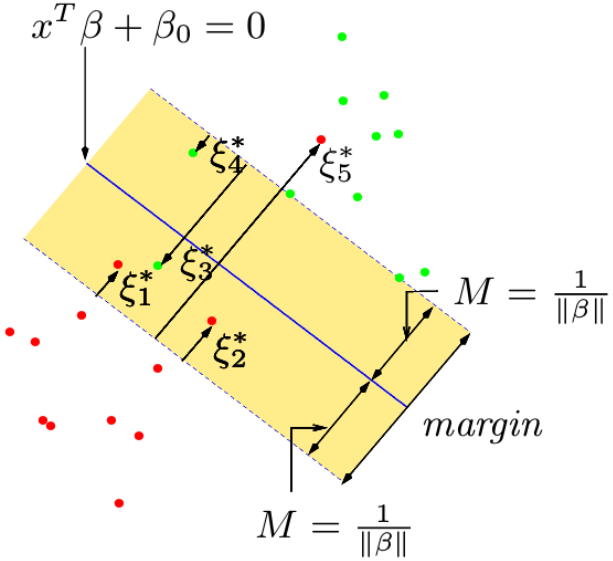
The Lagrange function equivalent representation is:

$$L_p = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[y_i(x_i^T\boldsymbol{\beta}+\beta_0)-(1-\xi_i)\right] - \sum_{i=1}^{n}\mu_i\xi_i$$

Where $\alpha_i, \mu_i, \xi_i \geq 0$. L_p is to be minimized with respect to $\boldsymbol{\beta}, \beta_0$ and ξ_i.

Setting the derivatives with respect to each to zero yields:

$$\boldsymbol{\beta} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i^T, \qquad \sum_{i=1}^{n}\alpha_i y_i = 0, \qquad \alpha_i = C-\mu_i, \forall i$$



Support Vector Machines

Substituting these results into L_p, for the non-separable case, yields the Wolfe dual

$$L_D = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i\, \alpha_k\, y_i y_k \boldsymbol{x}_i^T \boldsymbol{x}_k$$

subject to $0 \le \alpha_i \le C$ and $\sum_{i=1}^{n} \alpha_i\, y_i = 0$

Observations:

Solution must satisfy the Karush-Kuhn-Tucker conditions:

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i\, y_i \boldsymbol{x}_i, \qquad \sum_{i=1}^{n} \alpha_i\, y_i = 0, \qquad \alpha_i = C - \mu_i, \forall i$$

and

$$\alpha_i\left[y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - (1-\xi_i)\right] = 0 \qquad (*)$$

$$\mu_i \xi_i = 0$$

$$y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - (1-\xi_i) \ge 0 \qquad (**)$$

for $i = 1, \ldots, n$.

Maximizing L_D, subject to the constraints, is a convex quadratic programming problem — standard quadratic programming algorithms can determine the solution

Thus

$$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i\, y_i \boldsymbol{x}_i$$

where $\hat{\alpha}_i$ terms are established by the quadratic programming solution maximizing solved determined by the quadratic programming algorithm maximizing L_D

Observations:

- Constraint $(*) \Longrightarrow \alpha_i > 0$ only if $(**)$ is satisfied with equality, i.e.

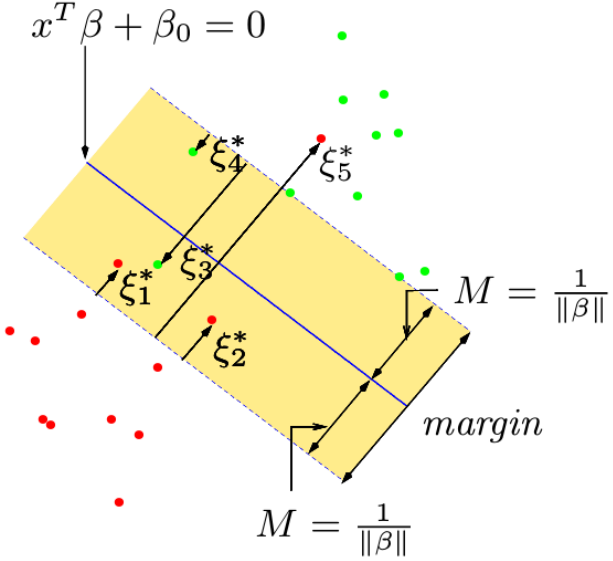
$$y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - (1-\xi_i) = 0 \qquad (***)$$

- Those $\boldsymbol{x}_i$ satisfying $(***)$ are the support vectors
- Support vectors are those on the wrong side of the boundary and those on the correct side of the boundary, but close to the boundary (within the margin)
- $\hat{\boldsymbol{\beta}}$ is determined as a linear combination of the support vectors
- The tuning parameter is the cost C

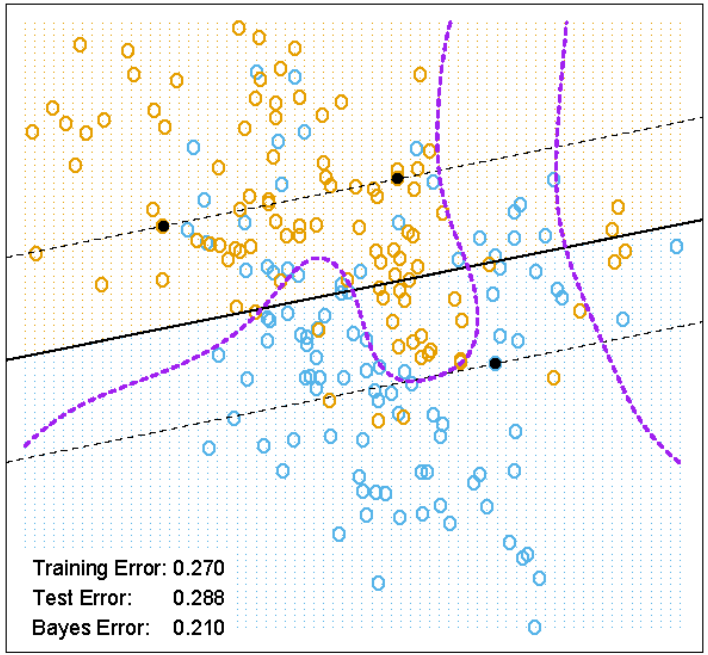
Support Vector Machines

Additional Observations:

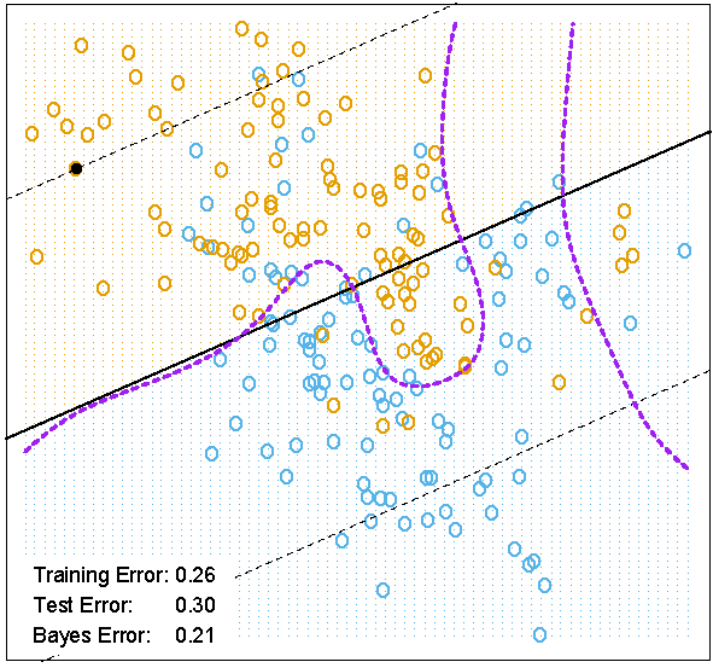
- The tuning parameter is the cost C
- Small $C \implies$ a larger margin, while a large $C \implies$ a smaller margin
- Large C focuses more on correctly classified points near the decision boundary (because margin is smaller)
- Small C involves correctly classified samples further away from the decision boundary (because margin is bigger)
- Incorrectly classified samples are always considered
- Optimal value for C can be estimated by cross-validation



Support Vector Machines



$C = 10000$



$C = 0.01$

Example: Mixture data with overlapping classes. Dashed lines indicate the margins, where $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta} + \beta_0 = \pm 1$. Support vectors ($\alpha_i > 0$) are those on the wrong side of the margin. Black dots are support vectors falling exactly on the margin ($\xi_i = 0, \alpha_i > 0$). Purple line is the Bayes decision boundary.

Support Vector Machines

Summary:

- Linear Support Vector Machines determine the optimal separating hyperplane between two classes
- If the data is not separable, the SVM problem statement can be expressed as the Wolfe dual

$$L_D = \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{k=1}^{n} \alpha_i\, \alpha_k\, y_i y_k \boldsymbol{x}_i^T \boldsymbol{x}_k$$

  subject to $0 \le \alpha_i \le C$ and $\sum_{i=1}^{n} \alpha_i\, y_i = 0$
- Solution must satisfy the Karush-Kuhn-Tucker conditions:

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i\, y_i \boldsymbol{x}_i, \qquad \sum_{i=1}^{n} \alpha_i\, y_i = 0, \qquad \alpha_i = C - \mu_i, \forall i$$

$$\alpha_i\left[y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - (1 - \xi_i)\right] = 0\,, \qquad \mu_i \xi_i = 0, \qquad y_i\left(\boldsymbol{x}_i^T\boldsymbol{\beta} + \beta_0\right) - (1 - \xi_i) \ge 0, \forall i$$

- Maximizing L_D, subject to the constraints, is a convex quadratic programming problem — standard quadratic programming algorithms can determine the solution
- $\widehat{\boldsymbol{\beta}}$ is determined as a linear combination of the support vectors — the vectors on the wrong side of the margin
- The tuning parameter C controls the margin; Small $C \implies$ a larger margin, while a large $C \implies$ a smaller margin
- For $f(\boldsymbol{x}) = x^T\boldsymbol{\beta} + \beta_0$, $f(\boldsymbol{x}) = 0$ defines the decision hyperplane boundary, $f(\boldsymbol{x}) = \pm 1$ define the hyperplane margins, and classification is determined as $G(x) = \text{sgn}(f(\boldsymbol{x}))$ for binary (± 1) classes