Support Vector Machines

* Margin

Quadratic programming

Sparsity of solution

Basis expansion

Kernel function

**George Runger 2020*

Maximum Margin *

- Consider a **separable** classification problem with two classes $y = \pm 1$

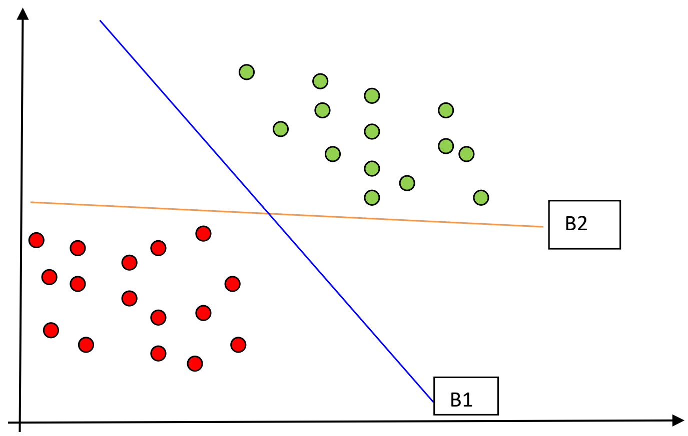
- A linear classifier is $f(\vec{x}) = w_0 + \vec{w}^T\vec{x}$ where $\hat{y} = sign \hat{f}(\vec{x})$

- Classifier $f(\vec{x}) = w_0 + \vec{w}^T\vec{x}$ defines a hyperplane (affine) with normal vector $\vec{w}$

- Instance i is classified correctly when $y_i(w_0 + \vec{w}^T\vec{x}_i) > 0$

- Perceptron problem: many solutions for a separable problem, how to pick one?

* *George Runger 2020*


B2
B1

Maximum Margin *

- Signed distance of point x_0 from the hyperplane is $\frac{1}{\|w\|}(w_0 + \vec{w}^T \vec{x}_0)$

- Distance of point to boundary unchanged by transform $\vec{w} \to b\vec{w}$, $w_0 \to bw_0$

- Select scale so that for nearest $+1$ instance, $(w_0 + \vec{w}^T \vec{x}) = 1$ and for nearest -1 instance, $(w_0 + \vec{w}^T \vec{x}) = -1$

- Then for all instances $y_i(w_0 + \vec{w}^T \vec{x}_i) \geq 1$

- **Margin** is distance between hyperplanes (parallel to decision boundary) through nearest instances $= 2/\|\vec{w}\|$

* *George Runger 2020*

Global Optimum *

- Goal: classify correctly with maximum margin

- $\min \frac{\|\vec{w}\|^2}{2}$ with constraints
  $y_i(w_0 + \vec{w}^T \vec{x}_i) \geq 1$ for $i = 1, \ldots, n$

- Quadratic programming problem with linear inequality constraints,

- Karush-Kuhn-Tucker (KKT) solution is obtained numerically to a *global* optimum
  $\hat{\vec{w}} = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$
  $\hat{w}_0 = y_i - \hat{\vec{w}}^T \vec{x}_i$ for any i where $\alpha_i > 0$

* *George Runger 2020*

Support Vectors *

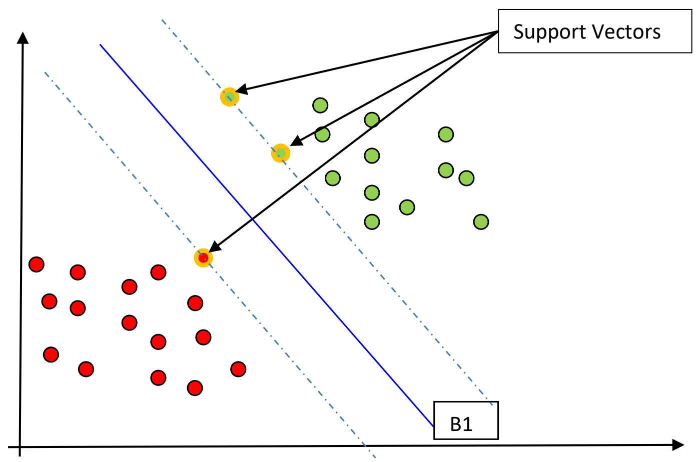
- KKT implies $\alpha_i = 0$ when $y_i(w_0 + \vec{w}^T\vec{x}_i) > 1$, that is, $\vec{x}_i$ is NOT on a margin hyperplane

- KKT implies $\alpha_i > 0$ when $y_i(w_0 + \vec{w}^T\vec{x}_i) = 1$, that is, $\vec{x}_i$ is on a margin hyperplane

- Therefore $\widehat{\vec{w}} = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$ is **sparse**, only $\vec{x}_i$ on a margin hyperplane contribute to solution

- Points $\vec{x}_i$ with $\alpha_i > 0$ are called **support vectors**

* *George Runger 2020*


Support Vectors
B1

Non-Separable Case *

- Still maximize margin, but allow errors in classifier

- Define slack variables $\xi_1, \ldots, \xi_n$
  Modify constraints to
  $y_i(w_0 + \vec{w}^T \vec{x}_i) \geq (1 - \xi_i), \xi_i \geq 0$

- Would like all $\xi_i = 0$, for no errors, but not possible in non-separable case

- $\xi_i / \|\vec{w}\|$ is a measure of distance of $\vec{x}_i$ on wrong side of margin, error occurs when $\xi_i > 1$

- min $\frac{\|\vec{w}\|^2}{2} + C \sum_{i=1}^{n} \xi_i$ with constraints
  $y_i(w_0 + \vec{w}^T \vec{x}_i) \geq (1 - \xi_i), \xi_i \geq 0, C > 0$

* *George Runger 2020*

Non-Separable Case *

- C large implies a smaller margin, with fewer error on the training data, less robust (and corresponds in the nonlinear case to a more complex model)

- C small implies a larger margin, with more error on the training data, but a more robust fit

- Same as separable case–solution is a quadratic programming problem, global optimum
  $\hat{\vec{w}} = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$
  $\hat{w}_0 = y_i - \hat{\vec{w}}^T \vec{x}_i$ for any support vector

- KKT again show sparse solutions (many α_i's $= 0$)

*George Runger 2020

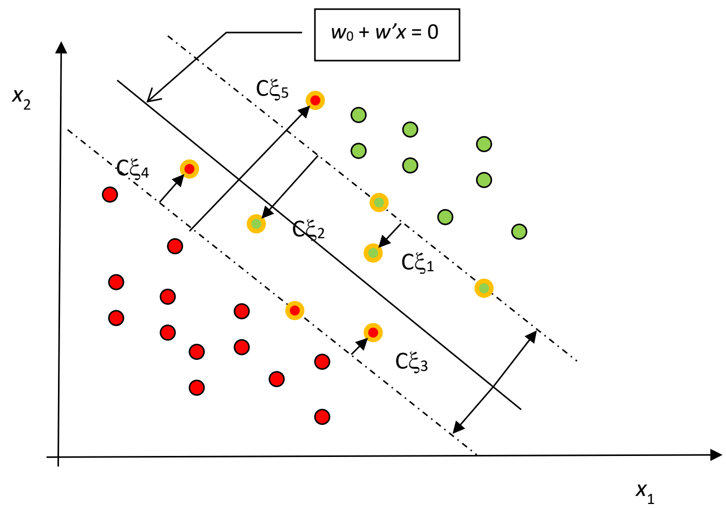
Non-Separable Case *

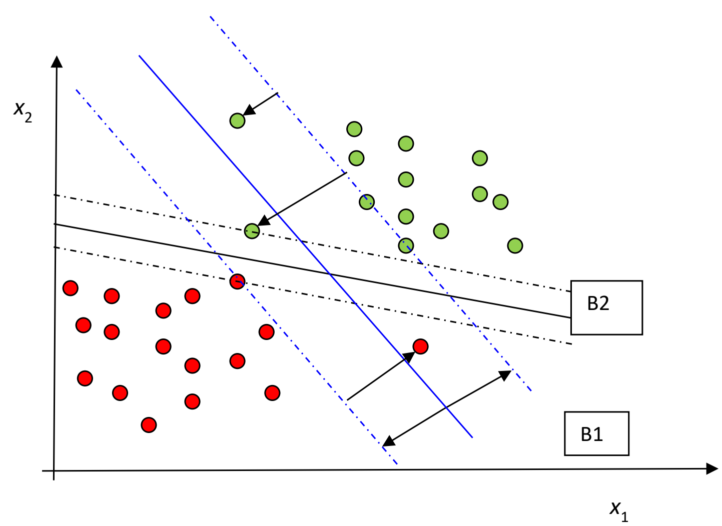
- Because misclassified point has $\xi_i > 1$, $\sum_{i=1}^{n} \xi_i$ is upper bound on number of errors

- $\xi_i = 0 \rightarrow x_i$ correct
  $0 < \xi_i < 1 \rightarrow x_i$ correct, but inside margin
  $\xi_i = 1 \rightarrow x_i$ on decision boundary
  $\xi_i > 1 \rightarrow x_i$ incorrect

- Classifier is $\widehat{y} = sign(\widehat{f}(\vec{x}))$ where $f(\vec{x}) = \widehat{w}_0 + \widehat{\vec{w}}^T \vec{x}$

*George Runger 2020


$w_0 + w'x = 0$
x_2
$C\xi_5$
$C\xi_4$
$C\xi_2$
$C\xi_1$
$C\xi_3$
x_1


x_2
B2
B1
x_1

Nonlinear Classifier *

- Function $f(\vec{x}) = w_0 + \vec{w}^T\vec{x}$ is linear and limited

- Basis expansion is used to dramatically enlarge the space of predictors (think polynomial terms)

- Instead of predictor $\vec{x}_i$ we use many more transformed features $\vec{\phi}(\vec{x}_i) : T \times 1$ with $T >> M$

- Linear model in $\vec{\phi}(\vec{x}_i)$, but nonlinear in $\vec{x}_i$

- Use C to control complexity, still maintain sparsity of solution

- Known as a **support vector machine** (SVM)

* *George Runger 2020*

Support Vector Machine *

- In the quadratic programming problem, feature vectors $\vec{\phi}(\vec{x}_i)$ only appear as *inner products* between instances $\vec{\phi}^T(\vec{x}_i)\vec{\phi}(\vec{x}_j)$

- Also, $\widehat{f}(\vec{x}) = \widehat{w}_0 + \sum_{i=1}^{n} \alpha_i y_i \vec{\phi}^T(\vec{x}_i)\vec{\phi}(\vec{x})$ so that inner products only appear in the solution

- **Key**: Do not need to calculate transformed vectors, only need their inner products

- **Kernel** function $K(\vec{x}_1, \vec{x}_2) = \vec{\phi}^T(\vec{x}_1)\vec{\phi}(\vec{x}_2)$ provides inner products in the transformed (higher-dimensional) feature space

* *George Runger 2020*

Support Vector Machine *

- Some common kernel functions are

$$K(\vec{x}_i, \vec{x}_j) = (1 + \vec{x}_i^T \vec{x}_j)^m$$

$$K(\vec{x}_i, \vec{x}_j) = \exp\left(-\|\vec{x}_i - \vec{x}_j\|^2/(2\sigma^2)\right)$$

$$K(\vec{x}_i, \vec{x}_j) = \tanh(\beta \vec{x}_i^T \vec{x}_j - \delta)$$

  for hyperparameters σ^2, β, δ

- Also solution written with kernel function
  $\hat{f}(\vec{x}) = \hat{w}_0 + \sum_{i=1}^n \alpha_i y_i \vec{\phi}^T(\vec{x}_i)\vec{\phi}(\vec{x})$
  $= \hat{w}_0 + \sum_{i=1}^n \alpha_i y_i K(\vec{x}_i, \vec{x})$

* *George Runger 2020*

Support Vector Machine *

- With 50 inputs, degree-2 polynomial contains 1325 terms

- Kernel function makes it feasible to compute these models

- Not unusual to use Gaussian kernel, how many polynomial terms?

- SVM "'magic"' comes from the feature expansion, the simplicity of the kernel function, and a global optimum

*George Runger 2020

Penalized Method *

- Another view: SVM solves the regularized problem

$$L = \min_{w_0, \vec{w}} \sum_{i=1}^{n} [1 - y_i(w_0 + \vec{w}^T \vec{x}_i)]_+ + \lambda \|\vec{w}\|^2$$

  where $[u]_+ = \max(u, 0)$ and hyperparameter $\lambda > 0$

- Note the error + smoothness penalty in L

- L is responsible for the sparseness of the solution

- As usual, $\widehat{y} = sign(\widehat{w}_0 + \widehat{\vec{w}})$

*George Runger 2020

- Such a result starts to integrate ridge regression, boosted decision trees, and SVM–on the surface very different algorithms, but with common roots