

Support Vector Machines

Sidharth Baskaran

June 2021

Optimization Objective

- Alternative view of logistic regression
 - If $y = 1$, we want $h_\theta(x) \approx 1$, $\theta^T x \gg 0$
 - If $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$
 - Cost is $J(\theta) = -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log(1 - \frac{1}{1+e^{-\theta^T x}})$
 - * If $y = 1$, consider only $-y \log \frac{1}{1+e^{-\theta^T x}}$, and $-(1-y) \log(1 - \frac{1}{1+e^{-\theta^T x}})$ for $y = 0$
 - Can approximate cost function term for each $y = 0, 1$ by 2 line segments to simplify optimization
- Recall for logistic regression must find

$$\text{LR} \rightarrow \min_{\theta} \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \underbrace{(-\log h_\theta(x^{(i)}))}_{\text{cost}_1(\theta^T x^{(i)})} + (1-y^{(i)}) \underbrace{(-\log(1 - h_\theta(x^{(i)})))}_{\text{cost}_0(\theta^T x^{(i)})} \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

$$\text{SVM} \rightarrow \min_{\theta} C \left[\sum_{i=1}^m y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{\lambda}{2} \sum_{j=1}^n \theta_j^2$$

- Support vector machine
 - The $\text{cost}_{0,1}(\theta^T x^{(i)})$ are the line approximations
 - Get rid of $\frac{1}{m}$ by convention
 - Use a parameter C instead of λ to weight for regularization, so $C = \frac{1}{\lambda}$
 - Directly outputs 0 or 1 from hypothesis

Large Margin (SVM) Classifier

- Properties due to piecewise sigmoid line approximation
 - If $y = 1$, then $\theta^T x \geq 1$
 - If $y = 0$, then $\theta^T x \leq -1$
- SVM decision boundary
 - When C is very large, want coefficient term to be 0
 - In a linear decision boundary, SVM finds largest margin (distance to clusters)
- Vector Inner Product
 - Let $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ and $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$
 - $u^T v = u_1 v_1 + u_2 v_2 = p \|u\|$ where p is $\text{len}(\text{proj}(u \rightarrow v))$
 - * Thus if angle between v and u is greater than 90 then $p < 0$
 - $\|u\| = \sqrt{u_1^2 + u_2^2} \in \mathbb{R}$
- SVM Decision boundary
 - Involves $\min_{\theta} \sum_{j=1}^n \theta_j^2 = \frac{1}{2}(\theta_1^2 + \theta_2^2) = \frac{1}{2}(\sqrt{\theta_1^2 + \theta_2^2})^2 = \frac{1}{2} \|\theta\|^2$ where $C = 0$ and $\theta_0 = 0$ for simplicity
 - $\theta^T x^{(i)} = p^{(i)} \|\theta\| = \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$

- Thus this the decision boundary can be redefined as $p^{(i)}||\theta|| \geq 1$ if $y^{(i)} = 1$ and $p^{(i)}||\theta|| \leq -1$ if $y^{(i)} = 0$

Kernels

- Complex nonlinear decision boundary
 - Given x , compute new features based on proximity to landmarks (points) $l^{(i)}$

$$f_i = \text{similarity}(x, l^{(i)}) = \exp\left(-\frac{||x - l^{(i)}||^2}{2\sigma^2}\right)$$

- Similarity function is the Gaussian kernel function $k(x, l^{(i)})$
 - Is ≈ 0 when far and ≈ 1 when close
 - Allows for defining new features for SVM to model complex nonlinear boundaries
- Given set of points $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$
 - Choose $l^{(i)} = x^{(i)}$
 - For training example $(x^{(i)}, y^{(i)})$, $f_m^{(i)} = \text{sim}(x^{(i)}, l^{(m)})$ where $f_0^{(i)} = 1$
- Hypothesis \rightarrow given x , compute features $f \in \mathbb{R}^{m+1}$ and predict $y = 1$ if $\theta^T f \geq 0$ and $y = 0$ if $\theta^T f < 0$
- Can plug this into the SVM cost function
- SVM parameters
 - Large $C \rightarrow$ low bias, high variance (small λ)
 - Small $C \rightarrow$ high bias, low variance (large λ)
 - $\sigma^2 \rightarrow$ a large value means smoother variance of features f_i , so high bias, low variance

Using SVM

- Use an SVM software package (e.g. `liblinear`, `libsvm`) in order to solve for θ
- Specify
 - Choice of C
 - Choice of kernel
- No kernel \rightarrow linear kernel
 - Would just use $\theta^T x \geq 0 \implies y = 1$
- If using Gaussian kernel, must choose σ^2
- Implementing kernel function
 - Perform feature scaling prior to kernel
 - Similarity function needs to satisfy Mercer's theorem \rightarrow cannot diverge

```
function f = kernel(x1, x2)
    f = e^(-(norm(x1-x2)^2)/(2*sigma^2));
end
```

- Kernel options
 - Polynomial $\rightarrow k(x, \ell) = (x^T \ell + c)^d$
 - * Worse performance than Gaussian
 - String, chi-square, histogram intersection
- Multi-class classification SVM where $y \in S = \{1, 2, 3, \dots, K\}$
 - Most have built-in handling, else use one-vs-all method
 - One-vs-all \rightarrow Train K SVMs, 1 to distinguish $y = i$ from rest for $i \in S$, then get $\theta^{(1)}, \dots, \theta^{(K)}$, then pick class i with largest $(\theta^{(i)})^T x$
- Logistic regression vs. SVMs
 - Let n = number of features $x \in \mathbb{R}^{n+1}$, m = number of training examples
 - If $n \geq m$, use logistic regression or SVM with linear kernel
 - If n is small, m is intermediate, use SVM with Gaussian kernel
 - If n is small, m is large, create/add more features and use logistic regression/SVM without kernel

- Convex optimization problem, no divergence issue
- Neural network works well for all, but slower to train