CS 4540/CS 8803: Algorithmic Theory of Intelligence Fall 2023

Lecture 4: Kernel Trick & SVM

Instructor: Santosh Vempala *Lecture date: 9/18*

1 The Kernel Trick

Suppose we have a set of points $x \in \mathbb{R}^2$ that are labelled in the following way:

$$\ell(x) = \begin{cases} 1 & x_1^2 + x_2^2 \leq 1 \text{ and } x_1^2 + \frac{1}{2}x_2^2 \leq 2 \\ -1 & \text{otherwise} \end{cases}$$

This is the intersection of the unit circle and an ellipse. This cannot be written as a linear classifier of the form $\ell(x) = ax_1 + bx_2 \geq c$.

Can we create a new set of features (y_1, y_2) such that $\ell(x) = 1$ if $ay_1 + by_2 \geq c$?

Yes; let $y_1 = \text{sign}(1 - x_1^2 - x_2^2)$ and $y_2 = \text{sign}(2 - x_1^2 - \frac{1}{2}x_2^2)$. Then, $\ell(x) = 1$ if $y_1 + y_2 \geq 2$.

In general, let $\phi : \mathbb{R}^n \to \mathbb{R}^N$ map x to a different set of features in $\mathbb{R}^N$ (N can be infinite!). We say $K(x, y)$ is a *kernel function* if we can write $K(x, y) = \langle \phi(x), \phi(y) \rangle$ for some function ϕ.

Examples:

- $K(x, y) = \langle x, y \rangle$, trivially with $\phi(x) = x$.
- $K(x, y) = (1 + \langle x, y \rangle)^d$ for a $d \in \mathbb{Z}^+$.

  This is the polynomial kernel, and the features are $c \prod_{i=1}^n x_i^{j_i}$ where the exponents add up to d, $\sum_{i=1}^n j_i = d$.
- $K(x, y) = e^{-\alpha \|x-y\|^2}$

  This is the Gaussian kernel; there exists a ϕ in infinite dimensions.

Kernel Perceptron. Suppose we want to learn the class defined by the kernel $K(x, y) = \langle \phi(x), \phi(y) \rangle$. Imagine we can compute $\phi(x)$ for each data point x. In this case, the label is 1 if $\langle w^*, \phi(x) \rangle > 0$, and we can learn w^* via the perceptron algorithm.

Algorithm 1 Perceptron Algorithm

```
Start with $w = 0$.
for each input $x$ do
    Predict sign($\langle w, \phi(x) \rangle$)
    On a mistake, set $w \leftarrow w + l(x)\phi(x)$
end for
```

What is the issue with this method? Recall that ϕ can have a much higher dimension than n; it can even be infinite. Therefore, we don't want to have to compute ϕ directly.

Fortunately, since K is computable, there is a way around this. Note that when guessing the label of x, we do not use $\phi(x)$ directly; instead, we only need the inner product of $\phi(x)$ and our estimate, $w = \sum_{i=1}^m \ell(x^{(i)})\phi(x^{(i)})$. This inner product is simply $\langle \phi(x), w \rangle = \sum_{i=1}^m \ell(x^{(i)}) K(x, x^{(i)})$. We will compute this by maintaining a list of inputs x where we have previously made a mistake.

Algorithm 2 Kernelized Perceptron Algorithm

```
Start with $w = 0$.
Let $M = []$.
for each input $x$ do
    Predict $\text{sign}(\sum_{y \in M} \ell(y)\langle \phi(y), \phi(x) \rangle)$ =$\text{sign}(\sum_{y \in M} \ell(y) K(y, x))$
    On a mistake, append $x$ to $M$.
end for
```

2 Support Vector Machines

Previously, we studied the perfect classification task. Given a set of inputs x, find a w such that $\forall x, \ell(x)\langle w, x \rangle \geq 1$.

However, this may not be possible in all applications if, for example, some labels are flipped due to noise or error. Suppose there is no perfect linear classifier, but we still want to find a w^* that is a 'good' linear classifier for our data.

One natural idea is to minimize the number of 'violations', datapoints that are misclassified by w.

Definition 1 (0-1 Loss).

$$L_{0-1}(w, X) = \sum_{x \in X} \mathbf{1}\{sign(\langle w, x \rangle) \neq \ell(x)\}$$

However, it is not feasible to compute a vector w which minimizes the 0-1 loss. In fact, it is NP-Hard to approximately minimize the loss. In practice, it is more feasible to minimize the sum of the distance from the separating plane for all misclassified points.

Definition 2 (Hinge Loss).

$$\min_{w, \zeta} C \sum_i \zeta_i + \|w\|_2^2$$

such that

$$\begin{array}{ccccc} \langle w, x^{(i)} \rangle & \geq & 1 - \zeta_i & & \textit{whenever } \ell(x^{(i)} = 1) \\ \langle w, x^{(i)} \rangle & \leq & -1 + \zeta_i & & \textit{whenever } \ell(x^{(i)} = -1) \\ \zeta_i & \geq & 0 & & \forall i \end{array}$$

The term $\|w\|_2^2$ is the regularization, which prevents the program making w very large to achieve good accuracy. The weight of this regularization term can be tuned using the constant C. Note that $\gamma_w = \min_x \frac{|\langle w, x \rangle|}{\|w\|_2}$. So, $\|w\|_2^2 \propto 1/\gamma_w^2$; the program minimizes the square of the inverse margin.

Unlike 0-1 Loss, the hinge loss is convex; this means that the optimal w can be found in polynomial time.

Theorem 3. *The number of mistakes made by percpetron is bounded by*

$$M \leq \min_w \frac{1}{\gamma_w^2} + 2 \sum_i \zeta_i$$

Proof. Let w^* be the true minimizer of the convex program in Definition 2. Consider the potential function $\langle w, w^* \rangle / \|w\|$, starting at 0. Whenever the algorithm makes a mistake, we update as $\tilde{w} := w + \ell(x_i)x_i$. For the numerator, we have

$$\langle \tilde{w}, w^* \rangle = \langle w + \ell(x_i)x_i, w^* \rangle = \langle w, w^* \rangle + \ell(x_i)\langle x_i, w^* \rangle \geq \langle w, w^* \rangle + 1 - \zeta_i$$

The next step is the same as our original proof of perceptron. We assume that $\|x\|$ is bounded by 1, so we have

$$\begin{aligned}\|\tilde{w}\|^2 =&\langle \tilde{w}, \tilde{w}\rangle = \langle w + \ell(x)x, w + \ell(x)x\rangle = \langle w, w\rangle + \langle x, x\rangle + 2\ell(x)\langle w, x\rangle \\ \leq&\langle w, w\rangle + \langle x, x\rangle \\ \leq&\|w\|^2 + 1\end{aligned}$$

After T mistakes, the numerator is bounded by $T - \sum_{i=1}^{T} \zeta_i$. The denominator is $\|w\| \leq \sqrt{T}$. By the Cauchy Schwarz inequality, $\langle w, w^*\rangle \leq \|w^*\|\|w\|$. So, $\frac{\langle w, w^*\rangle}{\|w\|} \leq \|w^*\|$.

$$\|w^*\| \geq \frac{\langle w, w^*\rangle}{\|w\|} \geq \frac{T - \sum_{i=1}^{T} \zeta_i}{\sqrt{T}}$$

Let $L = \sum_{i=1}^{T} \zeta_i$. In other words.

$$(T - L)^2 \leq T\|w^*\|^2,$$

which implies that

$$T^2 - 2TL \leq T\|w^*\|^2$$

and hence

$$T \leq \|w^*\|^2 + 2L = \frac{1}{\gamma_w^2} + 2\sum_{i=1}^{T} \zeta_i$$

as claimed.

$\square$