Support Vector Machines

Non-probabilistic models for classification of the form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}^i) + w_0$$

Learning fixes values of α_i. These are chosen in such a way that only a few *i*s have non-zero α_i. These are called support vectors.

Assume $y_i \in \{-1, +1\}$, binary classification model.

Predicted class label = $\text{sign}(f(\mathbf{x}))$

When Kernel is linear

If kernel is linear: $K(\mathbf{x}, \mathbf{x}^i) = \mathbf{x}.\mathbf{x}^i$

Then the SVM classifier $f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}^i) + w_0$ reduces to $f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + w_0$ where $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

$K(x, x) =$

$x^i \in R^d$

$\alpha_i \in R, \; y_i \in \{+1, -1\}$

$f(x) = \sum_i \alpha_i y_i \underbrace{K(x, x^i)} + w_0$

$= \boxed{\sum_i \alpha_i y_i x^i} x + w_0$

$f(x) = w \cdot x + w_0$

α_i	SVM $D =$	$d=2$
0	x^1:	$(1, 2), +1$
0.1	x^2:	$(0.5, -0.5), -1$
	x^3	
	.	
0.3	x^N	$(2, -1), +1$

Binary logistic regression

$W = [+1.3, -1.7]$ $f(-1)$

$k=2$; assume class labels are $\{+1, -1\}$

$f(x) = w \cdot x + w_0$

when $k=2$ then

instead of separate parameters for each class: we create $w^1 = w$; $w^{(-1)} = -w$ $\quad w^1 \quad w^{(-1)}$

Training SVMs

$D = \{(x^i, y_i) : i = 1 \cdots N\}$

Given a training dataset, find α_is and w_0 so as to minimize a new margin loss called the Hinge loss.

For correct prediction: $yf(x) > 0$ for a test example $(\underline{x}, \underline{y})$

Hinge loss: $L_H(y, f(\mathbf{x})) = \max(0, 1 - yf(\mathbf{x}))$

Loss is 0 iff $yf(\mathbf{x}) \geq 1$.

To minimize 0/1 prediction error we want to minimize # of examples for which $yf(x) < 0$

$$\text{minimize}_f \sum_{i=1}^{N} -y_i f(x_i)$$

unbounded even after training loss is zero.

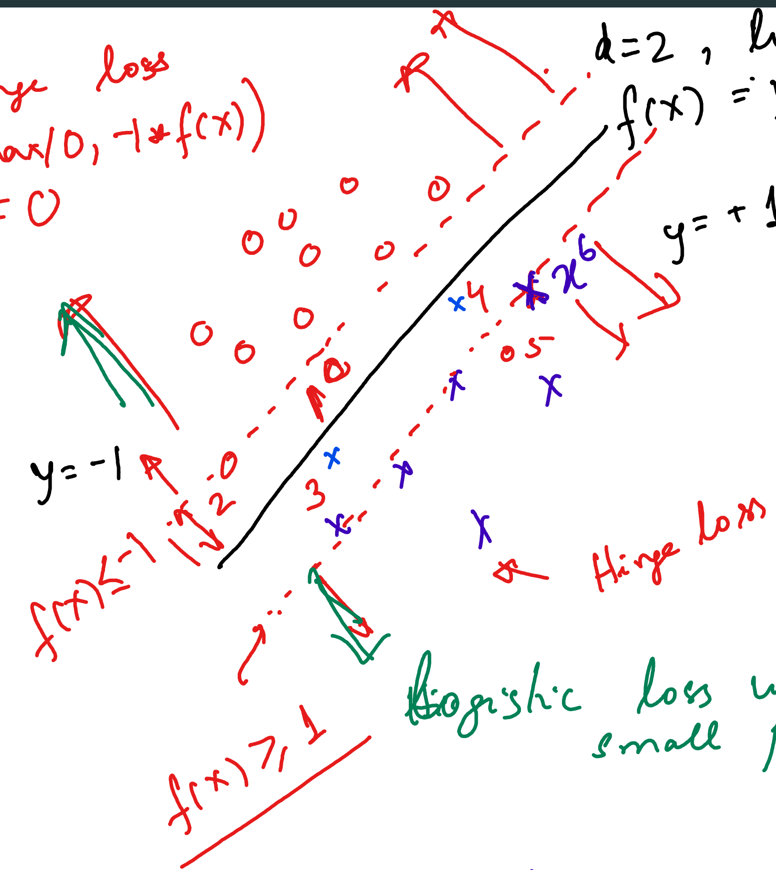
Perceptron loss: $\text{minimize}_f \sum_{i=1}^{N} \max(0, -y_i f(x_i))$

Understanding Hinge Loss geometrically

Hinge loss $\max(0, -1 * f(x)) = 0$

$d = 2$, linear kernel.

$f(x) = \vec{w} \cdot x + w_0$



Hinge loss $= \max(0, 1 - y f(x))$

$= 0$

Logistic loss will have small positive values.

Hinge loss (L_H) of point 1 x^1 between 0 & 1 because $-1 \le f(x) \le 0$

L_H of x^5 $[f(x^5) > 1]$

$L_H(x^5, y = -1) = \max(0, 1 - (-1) f(x^5)) = 1 + f(x) \geq 2$

$L_H(x^3) \in (0, 1)$

$L_H(x^6) = 0$

Leads to selecting separators that pass through sparse regions of the data distribution.

Comparing Hinge loss and logistic loss.

Advantage of hinge-loss: sparsity.

Perceptron loss: $\max(0, -yf(x))$ is inferior to Hinge loss because of not imposing margin

- Hinge loss: $L_H = \max(0, 1 - yf(x))$
- Logistic loss: loss used in logistic regression classifier:

(for $k = 2$) $L_\ell = \log(1 + e^{-yf(x)})$

If $yf(x) > 1$ then $L_H = 0$

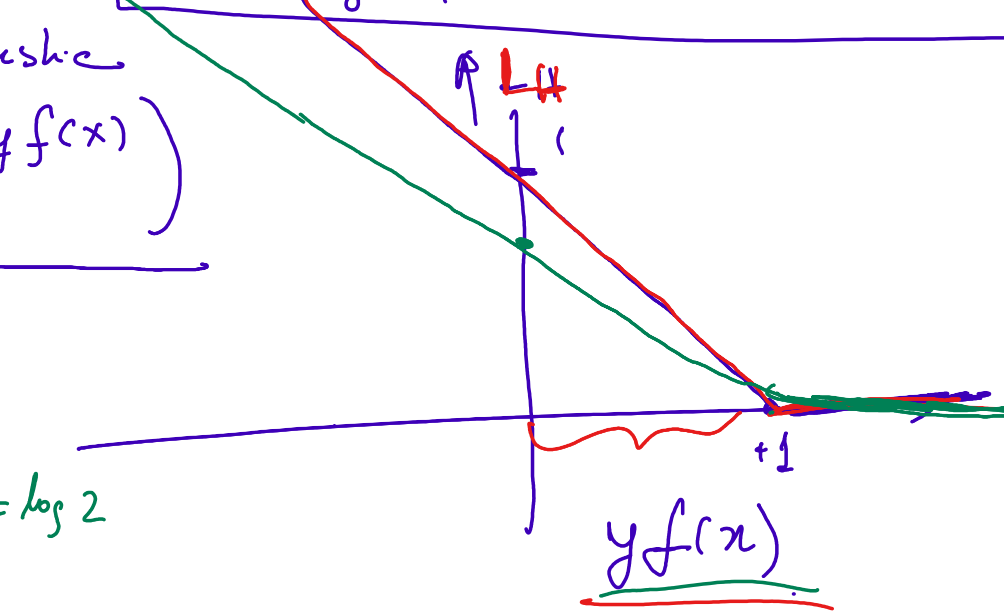
else $L_H = 1 - yf(x)$

Logistic

If $yf(x) = 0$ the $L_\ell = \log(1+1) = \log 2$

If $yf(x) \to \infty$ $L_\ell \to 0$

If $yf(x) >> -1$ $L_\ell \approx -yf(x)$



Comparing Hinge loss and logistic loss.

Advantage of hinge-loss: sparsity.

Training SVMs

First consider the linear case:

Training objective:

hyper-parameter

$$\min_{\mathbf{w}, w_0} C \sum_{i=1}^{N} \underbrace{\max(0, 1 - y_i(\mathbf{w}.\mathbf{x}^i + w_0))}_{\text{Hinge loss on training data with } f(x^i) = w \cdot x^i + w_0} + ||\mathbf{w}||^2 \quad (1)$$

regularizer to further prevent over-fitting.

Above loss is convex in W, and W_0.

1) $\max(1 - yf(x))$ is convex — and $f(x)$ is linear in W.

2) $||W||^2$ is convex in W

3) Sum of convex functions is convex.

In general if $L(W) = g(z)$ where z is a linear function of W and $z = W^T x$ and $g(z)$ is convex then $L(W)$ is convex.

Training objective as a constrained program.

Equivalent to: But loss is not differentiable in w.
Can use stochastic sub-gradient descent algorithm for training w, w_0

$\max(0, 1-y^i(w \cdot x^i + w_0))$

$= \xi_i$

$\xi_i \geq 1 - y^i(w \cdot x^i + w_0)$
$\xi_i \geq 0$

$$\boxed{\begin{aligned} &\min_{\mathbf{w}, w_0, \xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{s.t. } y^i(\mathbf{w}.\mathbf{x}^i + w_0) \geq 1 - \xi_i \quad \forall i : 1 \ldots N \\ &\xi_i \geq 0 \quad i : 1 \ldots N \end{aligned}} \tag{2}$$

Show

Training objective in dual form

$$\max_{\alpha_1,\ldots,\alpha_N} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}^i . \mathbf{x}^j \right) \right\}$$

quadratic in $\alpha_1 .. \alpha_N$

concave in $\alpha_1 .. \alpha_N$

$$\text{s.t. } \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3}$$

← linear constraint

$$0 <= \alpha_i <= C, \quad i : 1 \ldots N$$

[box constraints]

Training algorithm for SVMs

⇒ Objective is concave in α-s

⇒ constraints are easy: box-constraint balancing constraint.

⇒ Can solve with projected gradient ascent algorithm.

⇒ one popular method is: SMO algorithm:

$\alpha_1^0 \ldots \alpha_N^0 \leftarrow [0 \ldots 0]$ $t=0$

while no change: $(t = 1 \ldots \infty)$

→ pick two α_i^t, α_j^t

find change Δ s.t.:

$\alpha_i^{t+1} \leftarrow \alpha_i - \Delta$ → ensures that $\sum \alpha_i^{t+1} y_i = 0$

$\alpha_j^{t+1} \leftarrow \alpha_j - \Delta \, \boxed{S(y_i, y_j)}$ → $+1$ if $y_i \neq y_j$; → -1 if $y_i = y_j$

Δ so as to cause maximum ascent in objective.

and $[0 \leq \alpha_i^{t+1} \leq C]$ $0 \leq \alpha_j^{t+1} \leq C$.

Kernel Trick: extending to non-linear kernels

In the dual objective replace $\mathbf{x}^i.\mathbf{x}^j$ with $K(\mathbf{x}^i, \mathbf{x}^j)$.

Same trick in $f(\mathbf{x}) = \mathbf{w}.\mathbf{x} + w_0$ where $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}^i$ ...

Dual objective for ~~linear~~ kernels.

$$\max_{\alpha_1, \alpha_2 \dots \alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boxed{x^i . x^j}$$

replace this with $k(x^i, x^j)$

s.t

$$0 \leq \alpha_i \leq C \qquad \forall i = 1 \dots N$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \iff \sum_{i: y_i = +1} \alpha_i = \sum_{i: y_i = -1} \alpha_i$$

Deployment:

$$f(x) = w \cdot x + w_0 \iff \underbrace{\boxed{\sum_{i=1}^{N} \alpha_i y_i x^i}}_{w} \cdot x + w_0$$

for linear kernel

$$\xrightarrow{\text{apply kernel trick}} \sum_{i=1}^{N} \alpha_i y_i k(x^i, x) + w_0$$

Demo

Training w_0

1) Pick any example j s.t ~~$\alpha_j = C$~~ $\alpha_j < C$

Now, solve for w_0 s.t.: $y_j f(x^j) = 1$

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i k(x^i, x) + w_0$$

https://drive.google.com/file/d/1LihEFIe5fyFbWPS342PA9NZdy2Kxjkvy/view?usp=sharing

Summary

- Kernel methods provide a new way to represent data and build models based purely on comparision of two points.
- Various kernels are available, each with its own characteristics.
- Kernel methods find applications in several ML algorithms: classifiers (Nearest neighbor, few-shot classification, SVMs) and regression (Kernel regression, support vector regression, Gaussian processes)