L8

- Midterm
- Recap: SVMs & kernels
- Multiple layer perception
- Toward neural network

Midterm: 3/31 Next Tuesday

- close book
- 90 minutes
- 12-hour window
- self-enforced (honor code)
- lecture 1-8
- conceptual, algorithm & analysis, code
- ~6 questions
- 1-page cheat sheet
- posted as an assignment, you can scan and upload solution.

Support vector machines

- Hinge loss + regularizer
- optimize via dual
- gives a solution that is robust

kernel method

- Nolinear separable datasets
- kernel trick

$x = (x^{(1)}, x^{(2)}, \ldots, x^{(d)})$

$\phi \rightarrow (x^{(1)}, x^{(2)}, \ldots, x^{(d)},$
$x^{(1)}x^{(2)}, \ldots x^{(1)}x^{(d)},$
$x^{(1)}x^{(1)}x^{(1)}, \ldots$
$\exp(x^{(1)}) \ldots)$

$x \rightarrow [\phi] \rightarrow \phi(x)$

"kernel feature mapping"

Challenge: $\phi(x)$ can be very high dimensional

idea: Observe that in most linear models, we access the data via inner products

$\langle \phi(x), w \rangle \rightarrow$ really what we need

$\langle \phi(x_i), \phi(x_j) \rangle \rightarrow$ Dot products in kernel space

$k(x_i, x_j)$ "kernel inner product"

Examples: ① $k(x_i, x_j) = \langle x_i, x_j \rangle$ linear kernel

② $k(x_i, x_j) = (\langle x_i, x_j \rangle)^2$ or $(c + \langle x_i, x_j \rangle)^2$ quadratic kernel

③ $k(x_i, x_j) = (1 + \langle x_i, x_j \rangle)^p$ polynomial kernel ?

④ $k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|_2^2) = \exp(-\frac{\|x_i - x_j\|_2^2}{\sigma^2})$

Gaussian kernel, or Radial basis function / RBF

Rules of kernel design

① Efficient computation $k(x_i, x_j)$

should be easy computable

② Symmetry $k(x_i, x_j) = k(x_j, x_i)$

③ $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ for some ϕ for all x_i, x_j

$= (c + \langle \phi(x_i), \phi(x_j) \rangle)^r$

Guaranteed to exist via Mercer's Theorem

[Statement : Given any n data points $x_1 \cdots x_n$

Form the symmetric mxm matrix M

$M_{ij} = k(x_i, x_j)$

Then the eigenvalues of M have to be non-negative

i.e. M has to positive semi-definite]

经马定法则

Thumb rules for constructing kernels

* $k(x, z) = x^T z$ is a kernel
* $k(x, z) = \text{poly}(k(x, z))$ is a kernel
* $k(x, z) = a k_1(x, z) + b k_2(x, z)$ is a kernel
* $k(x, z) = \exp(k(x, z))$ is a kernel

$k(x, z) = f(x) k(x, z) f(z)$ is a kernel

Exercise: show using above rules that

$k(x,z) = \exp(-\gamma \|x-z\|_2^2)$ is a valid kernel

Instantiate in a concrete example → perceptron

---

kernel Perceptron

Input: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

ϕ

Output: $\text{Sign}(\langle w, \phi(x_i) \rangle) = y_i$

Algorithm:

0. initialize $w_0 \leftarrow 0$

1. Repeat:

For each $(x_i, y_i) \in S$

if $\text{sign}\langle w_t, \phi(x_i) \rangle \neq y_i$

$w_{t+1} \leftarrow w_t + y\phi(x_i)$

if no change in w_t after sweeping through data set, exit

until $t \in$ max epochs

$\langle w_t, \phi(x_i) \rangle \rightarrow$? kernel inner product

---

trick: observe: $w_0 = 0$

$w_1 = y_i \phi(x_i)$ for some i

$w_2 = y_i \phi(x_i) + y_i' \phi(x_i')$

In general, $w = \sum_{i \in S} \alpha_i y_i \phi(x_i)$

α_i encodes # times its datapoint has been visited.

$\therefore$ just a list of α_i's $\in \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}$

During test $\text{sign}(w, \phi(x))$

$$= \text{sign}\left(\left\langle \sum_{i \in S} \alpha_i y_i \phi(x_i), \phi(x) \right\rangle \right)$$

$$= \text{sgn}\left(\sum_{i \in S} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle \right)$$

$$= \text{sgn}\left(\sum_{i \in S} \alpha_i y_i k(\phi(x_i), \phi(x)) \right)$$

model

other algorithms [linear regression, SVMs, ridge regression, etc.] can all be kernelized.

Summary:

- Establish that final model is a linear combination of data points

$$w = \sum_{i \in S} \alpha_i y_i x_i$$

- Ensure that intermediate updates can be implemented via inner products.
- Replace all occurance of $\langle x_i, x_j \rangle$ to $k\langle x_i, x_j \rangle$

kernel nearest neighbours ?

Train : $(x_1, y_1), \ldots, (x_n, y_n)$

Test : X

$$f(x) = \sum_{i=1}^{n} y_i \; \delta(X, x_i)$$

$\delta \rightarrow$ delta function $= \begin{cases} 1 & \text{if } x_i \text{ is nearest of } X \\ 0 & \text{otherwise} \end{cases}$

---

kernel perception / SVM.

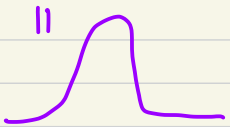
$$f(x) = \text{Sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \; k(X, X_i)\right)$$

结果更取决于 nearest label.

$X = X_i$ 时 $k(X, X_i) = 1$

从多是 KNN 方法

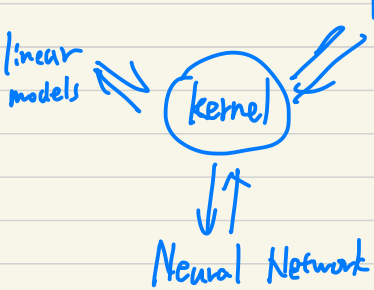
If we choose $k(X, X_i) = \exp(-\gamma \|x - x_i\|^2)$



As limit $r \rightarrow \infty$,

$k(X, X_i) \rightarrow \delta(X, X_i)$

所以有这种关系



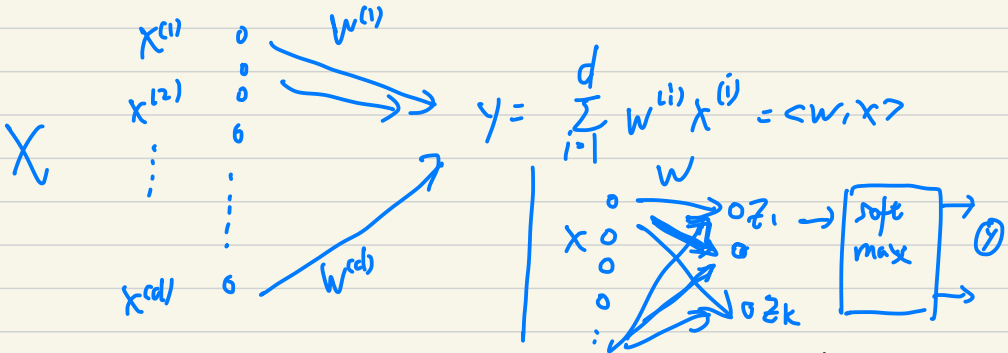
Step back

eg linear regression

- choose a linear model : $\langle w, x \rangle$
- compare with provided label via loss function $l(\ldots)$ eg.

$$L(y, \langle w, x \rangle) = \frac{1}{2} \| y - w^T x \|_2^2$$

- If this is non-zero, update via grad descent



eg. logistic regression (k - class)

- linear model $\langle w_1, x \rangle$, $\langle w_2, x \rangle$, $\vdots$, $\langle w_k, x \rangle$
- Softmax : $\hat{y} = \text{softmax}(z)$

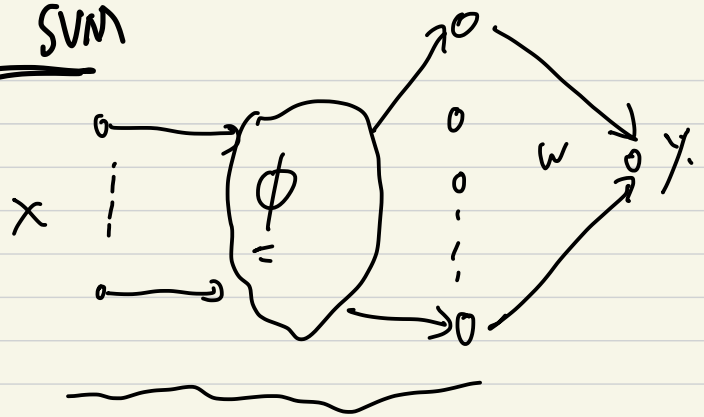
$$\hat{y}_i = \frac{\exp(\langle w_i, z \rangle)}{\sum_{i=1}^{k} \exp(\langle w_k, z \rangle)}$$

- cross entropy : $l(y, \hat{y}_i)$

$$= -\sum_{i=1}^{k} y_i \log(\hat{y}_i) \qquad y = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- if loss is non-zero, update w. $\langle \phi(x), w \rangle$

kernel SVM



kernel methods: hand chosen

Neural methods: learned feature maps.