ECE-GY 6143
Intro to ML

- Midterms
- Recap: SVMs & kernels
- Multilayer perceptrons
- Towards neural networks

Mid term: Next Tuesday, 3/31.

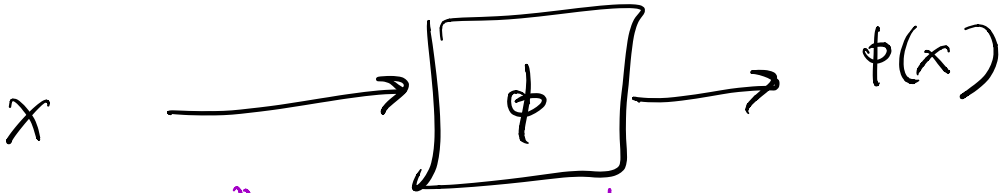
- Closed book
- timed, 90 minutes.
- 12-hour window
- Self enforced (honor code).
- Lectures 1-8
- Conceptual, algorithms & analysis, code.
- ~6 questions
- 1-page cheat sheet.
- Posted as an assignment; you scan and upload answers.

1) Support vector machines.
   - Hinge loss + regularizer
   - Optimize via dual.
   - Gives a solution that is robust.

2) Kernel methods
   - Nonlinear datasets
   - Kernel trick.

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

$\phi \rightarrow (x^{(1)}, x^{(2)}, \ldots, x^{(d)}, x^{(1)}x^{(2)}, \ldots x^{(1)}x^{(d)}, \ldots, x^{(1)}x^{(2)}x^{(3)}, \ldots, \exp(x^{(1)}), \ldots)$



"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.

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

$\langle \phi(x_i), \phi(x_j) \rangle$ → 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 $(1 + \langle x_i, x_j \rangle)^2$

"Quadratic kernel".

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

"polynomial kernel".

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

"Gaussian kernel" or "Radial basis function / RBF".

Rules of kernel design.

① Efficient computation : $k(x_i, x_j)$ should be easily 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

Guaranteed to exist via **Mercer's Theorem**

[Statement: Given any n data points $x_1, x_2, \ldots, x_n$.
Form the symmetric $n \times n$ 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.]

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
- $k(x, z) = \exp(k(x, z))$
- $k(x, z) = f(x)\, k(x, z)\, f(z)$

is a kernel.

Exercise: Show using above rules that

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

Instantiate kernels in a concrete example → perceptron.

Kernel Perceptron

Input: $S = (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$.

ϕ

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

Algorithm

0. Initialize $w_0 \leftarrow 0$
1. Repeat:
   a. For each $(x_i, y_i) \in S$
   b. If $\text{sign}\langle w_t, \phi(x_i)\rangle \neq y_i$ //
   c. $w_{t+1} \leftarrow w_t + y\,\phi(x_i)$
   d. If no change in w_t after sweeping through data, exit.

until $t \leq$ 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

$w_2 = y_i \phi(x_i) + y_{i'} \phi(x_{i'})$

$w_3 = y_i \phi(x_i) + y_{i'} \phi(x_{i'}) + y_{i''} \phi(x_{i''})$

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

α_i encodes # times i^{th} data point 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 $\text{sgn}(\langle w, \phi(x) \rangle)$

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

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

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

Other algorithms (linear regression, SVMs, ridge regression, etc) can all be kernelized.

Summary:

1) Establish that final model is a linear combination of data points.

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

2) Ensure that intermediate updates can be implemented via inner products.

3) Replace all occurrences of $\langle x_i, x_j \rangle$ to $K(x_i, x_j)$.

Kernel nearest neighbors (?)

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

Test : x

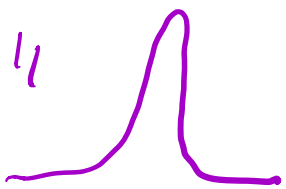
$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} y_i \, \delta(x, x_i)\right)$$

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

Kernel perceptron / SVM

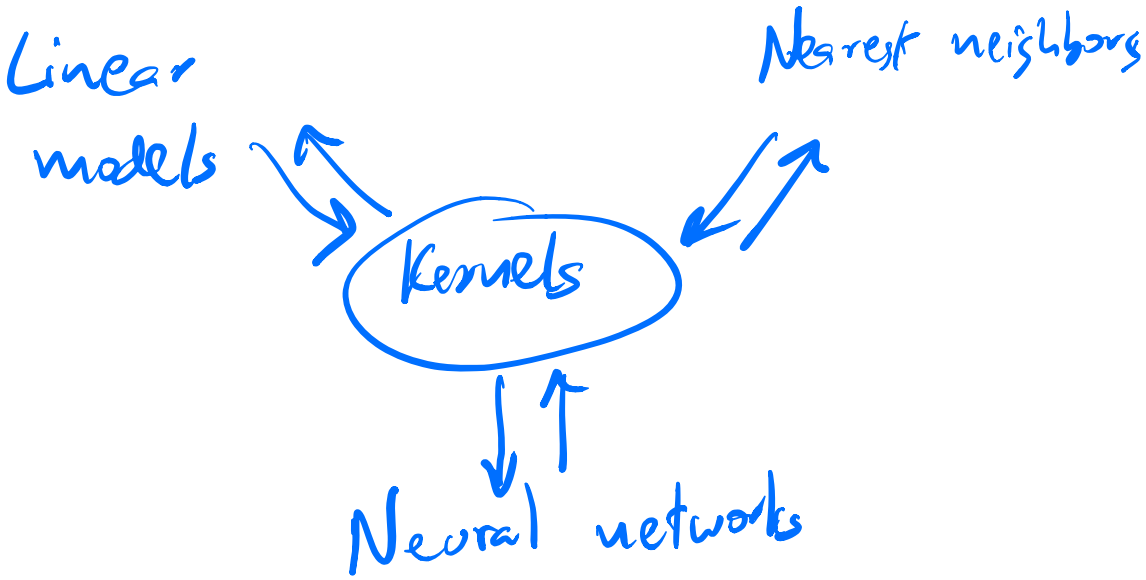
$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i \, K(x, x_i)\right)$$

If we choose $K(x, x_i) = \exp\left(-\gamma \|x - x_i\|^2\right)$



As limit $\gamma \rightarrow \infty$,

$K(x, x_i) \rightarrow \delta(x, x_i)$.



Step back

e.g. linear regression

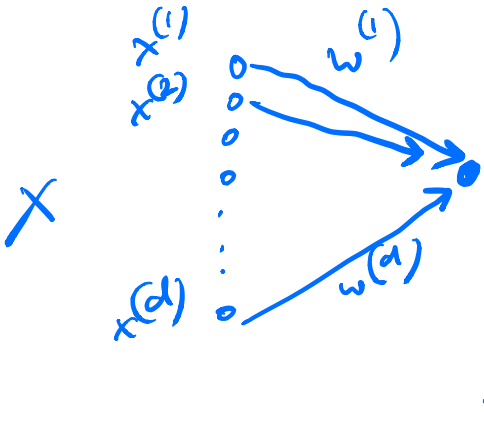
- Choose a linear model:

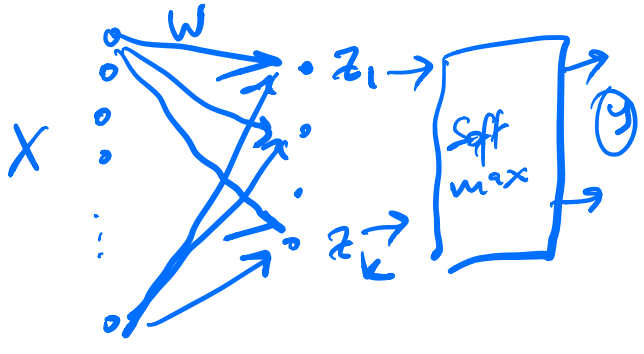
  prediction: $\langle w, x \rangle$

- Compare with provided label via loss function $\ell(\cdot, \cdot)$ e.g.

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

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





e.g. logistic regression (k-class)

- linear model: $\begin{matrix} \langle w_1, x \rangle \\ \langle w_2, x \rangle \\ \vdots \\ \langle w_k, x \rangle \end{matrix} = z$

- soft max: $\hat{y} = \text{softmax}(z)$

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

$\sum_{i=1}^{k} \exp(\langle w_i, z \rangle)$.

- Cross entropy loss: $\ell(y, \hat{y})$

$$= -\sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

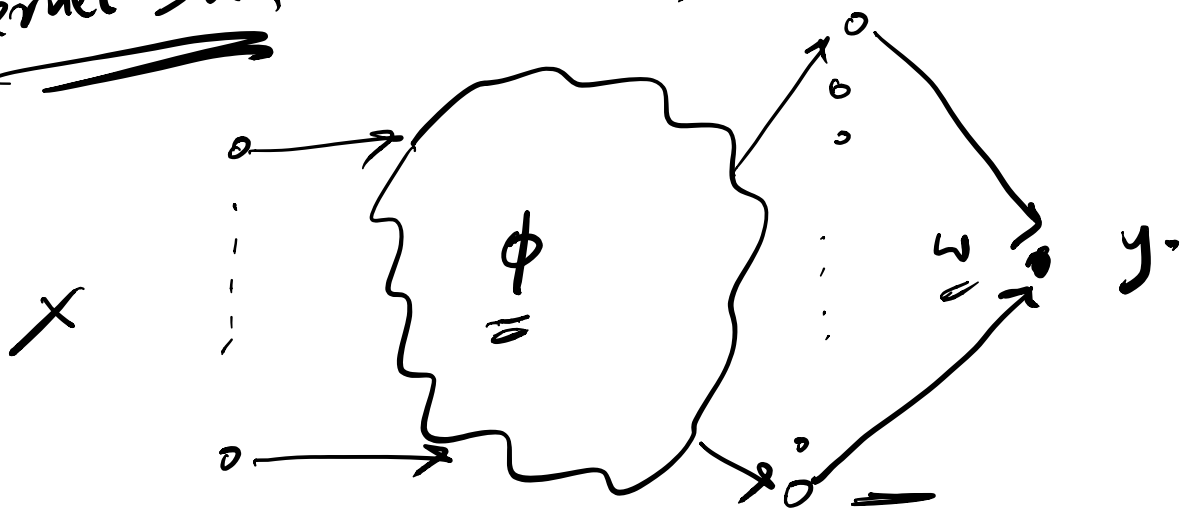
$$y = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$$

"One-hot encoding"

- If loss is non-zero, update w.

Kernel SVM

$\langle \phi(x), w \rangle = y$



Kernel methods: hand chosen feature maps

Neural networks: learned feature maps.