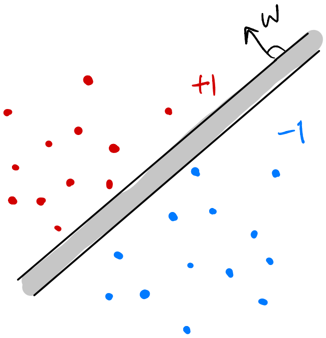
LECTURE 20

- Last class: Binary Classification Problem



Training data: $\Big(\underset{\mathbb{R}^d}{\overset{\text{test results}}{x_i}}, \underset{\{-1,\,1\}}{\overset{\text{diagnosis}}{y_i}}\Big)$, $i=1,\dots,n$.

Learning: find $w \in \mathbb{R}^d$: $\langle w, x_i\rangle \begin{cases} >1 & \text{if } y_i=1 \\ <-1 & \text{if } y_i=-1 \end{cases}$

$$\iff \langle w, x_i\rangle y_i > 1 \quad \forall i$$

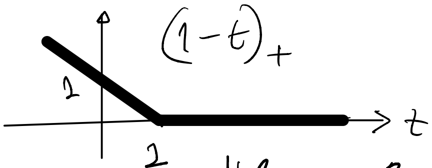
Prediction: diagnosis for a new x : $\boxed{f(x) = \text{sign}\langle w, x\rangle}$

- Typical data is not perfectly separable : ∃ bad pts, outliers.

To allow them, we

- Penalize : pay penalty $1-\langle w, x_i\rangle y_i$ for each bad pt :

$\Rightarrow$ loss $\ell(w) := \sum_{i=1}^{n} \big(1-\langle w, x_i\rangle y_i\big)_+$ where



"hinge loss"

Minimize $\ell(w)$ over $w \in \mathbb{R}^n$ (convex program).

- Furthermore, regularize (for robustness):

$$\boxed{\ell(w) = \sum_{i=1}^{n} \big(1-\langle w, x_i\rangle y_i\big)_+ + \lambda \|w\|_2^2}$$

"Soft-margin SVM"

- And, since data may not be linearly separated, kernelize.

KERNEL METHOD (Lec.19)

- Rewrite the algorithm in terms of *inner products*: $w=\sum \alpha_j x_j \Rightarrow$

loss $$\ell(w)=\sum_{i=1}^{n}\left(1-\sum_{j=1}^{n}\alpha_j\langle x_i,x_j\rangle y_i\right)_+ +\lambda\sum_{i,j=1}^{n}\alpha_i\alpha_j\langle x_i,x_j\rangle,$$

which we minimize in (α_j).

prediction: $f(x)=\text{sign}\left(\sum\alpha_j\langle x,x_j\rangle\right)$.

- Transform data using a nonlinear *feature map* (implicit!) $\phi:\mathbb{R}^d\to \underset{\uparrow}{H}$ Hilbert space

$$\Rightarrow \langle x,y\rangle \longmapsto \langle\phi(x),\phi(y)\rangle =: K(x,y) \quad \text{"kernel"}$$

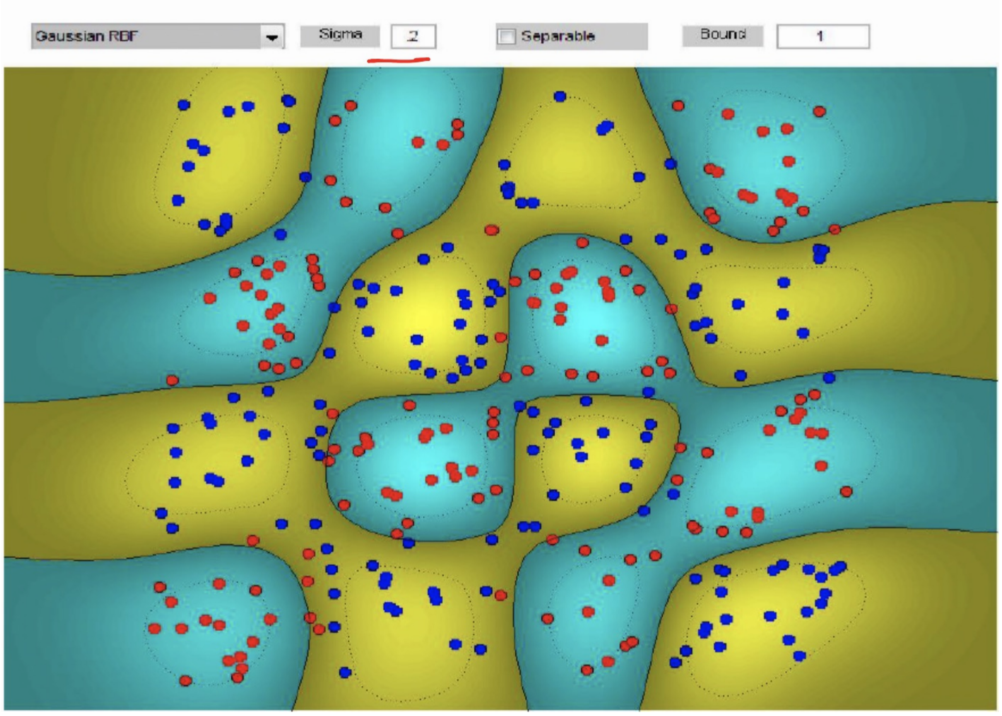
- Choose a nice *kernel*, e.g. $K(x,y)=\exp\left(-\frac{\|x-y\|_2^2}{2\sigma^2}\right)$ (RBF)

and replace all $\langle\cdot,\cdot\rangle$ by $K(\cdot,\cdot)$.

"*Kernel SVM*"

- *Warning*: for practical implementation, see Wikipedia (kernelize the dual program).

EXAMPLE:



Gaussian RBF

Sigma 2

☐ Separable

Bound 1

WHAT FUNCTIONS $K(x,y)$ ARE ALLOWED?

- Can we express $\forall$ function K as $K(x,y) = \langle \phi(x), \phi(y) \rangle$?
- No: the matrix $[K(x_i, x_j)]_{i,j=1}^n = [\langle \phi(x_i), \phi(x_j) \rangle]_{i,j=1}^n$

  is a Gram matrix $\Rightarrow$ must be PSD.

DEF A kernel is a symmetric function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that

$$[K(x_i, x_j)]_{i,j=1}^n \succcurlyeq 0 \quad \forall \; x_1, \dots, x_n \in \mathbb{R}^d .$$

- This is a sufficient condition:

THM (Mercer's Condition) $\forall$ continuous kernel K

$\exists$ map $\phi : \mathbb{R}^d \to H$ (H=Hilbert space) such that

"feature map" "feature space"

$$K(x,y) = \langle \phi(x), \phi(y) \rangle_H \qquad \forall x, y \in \mathbb{R}^d$$

[This is an ∞-dimensional version of the HW problem:
$\forall$ PSD matrix is a Gram matrix $(x \mapsto i, \; y \mapsto j)$]

- Mercer's condition (PSD) is difficult to check.

  However, using it, we can build new kernels from old ones using simple rules (HW7), such as addition, exponentiation, etc.

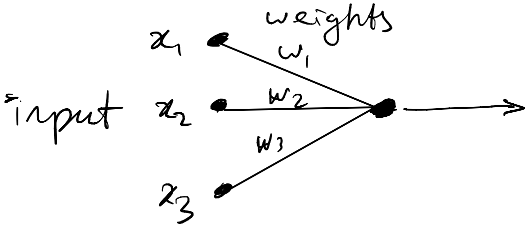
- Examples: (a) RBF $K(x,y) = \exp\left(-\frac{\|x-y\|_2^2}{2}\right)$,

  (b) polynomial $K(x,y) = (1 + \langle x,y \rangle)^p$, ...

Kernels arise in:

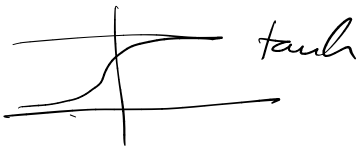
NEURAL NETWORKS

Def A neuron is a composition of linear & nonlinear function



$\sigma(w_1x_1+w_2x_2+w_3x_3)$ output

e.g. $\sigma=\text{sign}$ or

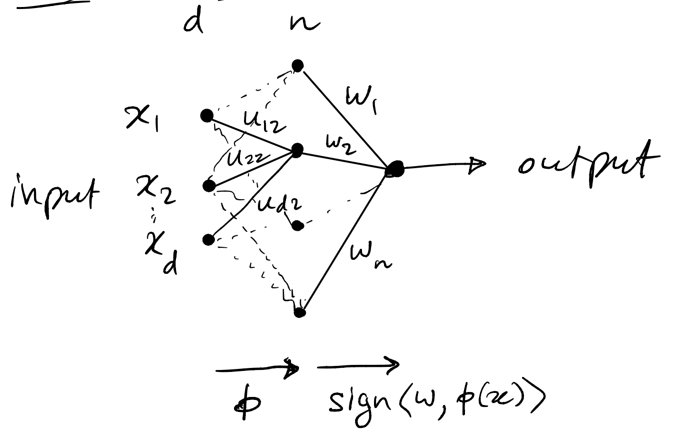


Ex: SVM is a neuron: $\text{sign}\langle w,x\rangle = \text{sign}(w_1x_1+\dots+w_dx_d)$

We train weights w_j

Def A neural network is a superposition of neurons.

We train weights $(u_{ij}),(w_j)$.



Connection to kernels:

- The first layer computes a feature map $\phi: R^d \to R^n$:

$$\phi(x)=\begin{bmatrix}\text{sgn}\langle u_1,x\rangle\\ \cdots\\ \text{sgn}\langle u_n,x\rangle\end{bmatrix}$$

- Kernel:

$$K(x,y)=\langle\phi(x),\phi(y)\rangle=\frac{1}{n}\sum_{i=1}^{n}\text{sgn}\langle u,x\rangle\langle u,y\rangle \quad (\rightarrow)$$

Let's initialize all weights $u_{ij}\sim N(0,1)$ iid

$$(\rightarrow)\quad \mathbb{E}\,\text{sgn}\langle u,x\rangle\,\text{sgn}\langle u,y\rangle \quad \text{where} \quad u\sim N(0,I_n)$$

$$=\boxed{\frac{2}{\pi}\arcsin\langle x,y\rangle.}$$

Hence: a neural network $\simeq$ kernel SVM.

- But: after initialization, a neural network is training weights u_{ij} $\Rightarrow$ trains kernel K.

- Dynamics of the prediction function is described by the neural tangent kernel (NTK).