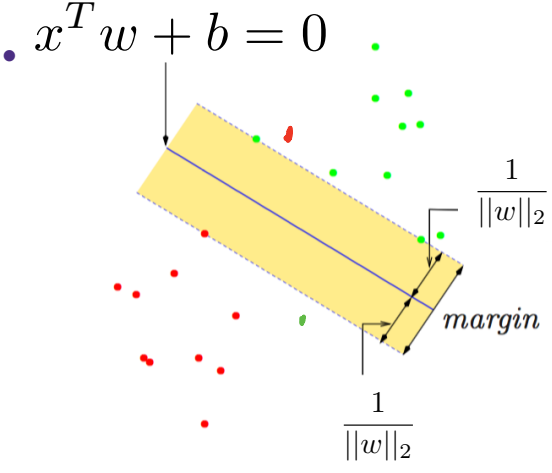
HW 2 is due

Kernels

What if the data is not linearly separable?



some points don't satisfy margin constraint:

$$\min_{w,b} ||w||_2^2$$

$$y_i(x_i^T w + b) \geq 1 \quad \forall i$$

Two options:

1. Introduce slack to this optimization problem
2. **Lift to higher dimensional space**

$x \mapsto \phi(x)$

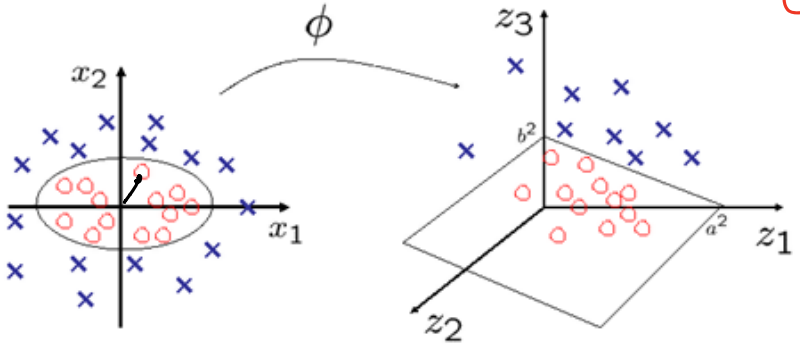
What if the data is not linearly separable?

Use features of features…

$(x_1, x_2) \rightarrow (r, \theta)$

$r = \sqrt{x_1^2 + x_2^2}$

$\theta = [0] \quad \frac{x_1 \cdot x_2}{|x_1||x_2|}$



$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \sqrt{x_1^2 + x_2^2} \end{pmatrix}$$

Creating Features

$x \in \mathbb{R}^d$, $h(x) \in \mathbb{R}^p$

Transformed data:

$h : \mathbb{R}^d \to \mathbb{R}^p$ maps original features to a rich, possibly high-dimensional space

in d=1: $$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_p(x) \end{bmatrix} = \begin{bmatrix} x \\ x^2 \\ \vdots \\ x^p \end{bmatrix}$$

for d>1, generate

$\{u_j\}_{j=1}^p \subset \mathbb{R}^d$ u_j random

$h_j(x) = (u_j^T x)^2$

$h_j(x) = \dfrac{1}{1 + \exp(u_j^T x)}$

$h_j(x) = \cos(u_j^T x)$

Creating Features

Transformed data:

ϕ: $h : \mathbb{R}^d \to \mathbb{R}^p$ maps original features to a rich, possibly high-dimensional space

in d=1: $$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ \vdots \\ h_p(x) \end{bmatrix} = \begin{bmatrix} x \\ x^2 \\ \vdots \\ x^p \end{bmatrix}$$

for d>1, generate

$$\{u_j\}_{j=1}^p \subset \mathbb{R}^d$$

$$h_j(x) = (u_j^T x)^2$$

$$h_j(x) = \frac{1}{1 + \exp(u_j^T x)}$$

$$h_j(x) = \cos(u_j^T x)$$

Feature space can get really large really quickly!

Degree-d Polynomials

input: $u \in \mathbb{R}^2$, (u_1, u_2)

degree-1: $\phi(u) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \in \mathbb{R}^2$

degree-2: $\phi(u) = \begin{pmatrix} u_1 \cdot u_1 \\ u_1 u_2 \\ u_2 \cdot u_1 \\ u_2 \cdot u_2 \end{pmatrix} \in \mathbb{R}^4$

degree-3:

$$\phi(u) = \begin{pmatrix} u_1 u_1 u_1 \\ u_1 u_1 u_2 \\ u_1 u_2 u_1 \\ u_1 u_2 u_2 \\ u_2 u_1 u_1 \\ u_2 u_1 u_2 \\ u_2 u_2 u_1 \\ u_2 u_2 u_2 \end{pmatrix} \in \mathbb{R}^8$$

if $u \in \mathbb{R}^p$

$\phi(u) = O(p^d)$ dim

exponential

$\Rightarrow n$: # of data

How do we deal with high-dimensional lifts/data?

A fundamental trick in ML: use kernels

A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a *kernel* for a map ϕ if $K(x, x') = \phi(x) \cdot \phi(x')$ for all x, x'.

So, if we can represent our algorithms/decision rules as dot products and we can find a kernel for our feature map then we can avoid explicitly dealing with ϕ(x).

Linear Regression as Kernels

Training $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$

$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$

solution regularized least square

$$\hat{w} = (X^T X + \lambda I_d)^{-1} X^T y \quad (d \times d,\ d \times 1)$$

$$= X^T (X X^T + \lambda I_n)^{-1} y \quad (d \times n,\ n \times n,\ n \times 1)$$

(linear algebra operations, e.g. SVD)

decision rule prediction

$x_{new} \in \mathbb{R}^d$

$$\hat{y}_{new} = \hat{w}^T x_{new}$$

$$= y^T (X X^T + \lambda I_n)^{-1} X x_{new} \quad (1 \times n,\ n \times 1)$$

α

$X x_{new} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix} \begin{bmatrix} x_{new} \end{bmatrix}$ ($n \times d$, d)

$X X^T \in \mathbb{R}^{n \times n}$

$i, j \in \{1, \ldots, n\}$ $\quad [X X^T]_{ij} = \langle x_i, x_j \rangle$

$[X x_{new}]_i = \langle x_i, x_{new} \rangle$

$\Rightarrow \langle \cdot, \cdot \rangle \to K(\cdot, \cdot)$

$\Rightarrow K(x_i, x_j)$

$K(x_i, x_{new})$

Dot-product of polynomials

$u \in \mathbb{R}^2$, $v \in \mathbb{R}^2$

$u \in \mathbb{R}^3$, $d = 2$

$(u_1, u_1, u_3)^T (u_1, u_1, u_3)$

$\Rightarrow q = 3^2$

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly d}$$

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

$K(u,v) = (\langle u, v \rangle)^2$

degree-d polynomial with $u \in \mathbb{R}^p$

if use explicit feature, # of operations $O(p^d)$

$K(u,v) = (\langle u, v \rangle)^d$, # of operations, $O(p + \log d)$

Dot-product of polynomials

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly d}$$

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \quad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

Feature space can get really large really quickly!

General d : Dimension of $\phi(u)$ is roughly p^d if $u \in \mathbb{R}^p$

Feature expansion can be written **implicitly** $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$

γ, r : hyper-parameter

Examples of Kernels

- **Polynomials of degree exactly d**

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

- **Polynomials of degree up to d**

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

- **Gaussian (squared exponential) kernel** (RBF Kernel)

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{2\sigma^2}\right)$$

- **Sigmoid**

$$K(u, v) = \tanh(\gamma \cdot u^T v + r)$$

$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

The Kernel Trick

Data $\{(x_i, y_i)\}_{i=1}^{n}$

α can depend on $\langle x_i, x_j \rangle$ and y

1. **Pick a kernel K** $\quad \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

2. **For a linear predictor, show** $w = \sum_{i=1}^{n} \alpha_i x_i \quad / \alpha \in \mathbb{R}^n$

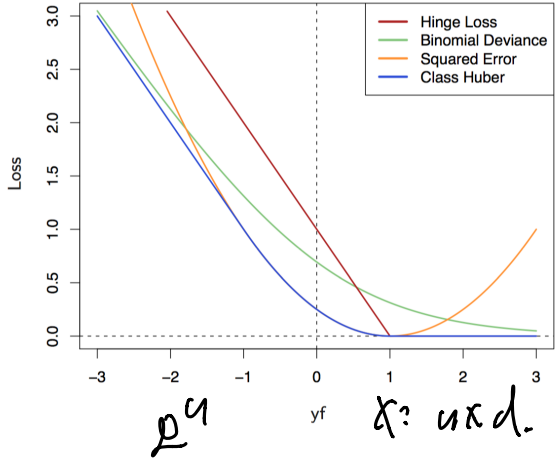
3. **Change loss function/decision rule to only access data through dot products**

$$w^T x_{new} = \sum_{i=1}^{n} \alpha_i x_i^T x_{new}$$

Substitute $K(x_i, x_j)$ for $x_i^T x_j$

$$\hat{y}_{new} = \sum_{i=1}^{n} \alpha_i K(x_i, x_{new})$$

Loss Functions



$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$

- Loss functions: $\sum_{i=1}^{n} \ell_i(w)$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

0/1 loss: $\ell_i(w) = \mathbb{I}[\text{sign}(y_i) \neq \text{sign}(x_i^T w)]$

Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$

SVM

$w = \alpha^T X$

$X_j^T w = \alpha^T X X_j$

$= \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle$

The Kernel Trick for regularized least squares

$\alpha \in \mathbb{R}^n$

$$\widehat{w} = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^{n} \alpha_i x_i$

$[y - K\alpha]_i := y_i - [K\alpha]_i$

$[K\alpha]_i = \sum_{j=1}^{n} \alpha_j K(x_i, x_j)$

$[K]_i = [K(x_i, x_1), \dots K(x_i, x_n)]$

$$\hat{\alpha} = \arg\min_{\alpha} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \alpha_j x_i^T x_j \right)^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j x_i^T x_j$$

$(\langle x_i, x_j \rangle \to K(x_i, x_j))$

$$\Rightarrow \arg\min_{\alpha} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} \alpha_j K(x_i, x_j) \right)^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j)$$

(Quadratic form)

$$= \arg\min_{\alpha} \| y - K\alpha \|_2^2 + \lambda \alpha^T K \alpha$$

$K \in \mathbb{R}^{n \times n}$, $[K]_{ij} = K(x_i, x_j)$

The Kernel Trick for regularized least squares

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^{n} \alpha_i x_i$

$$\begin{aligned}
\widehat{\alpha} &= \arg\min_{\alpha} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_j \langle x_j, x_i \rangle)^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle \\
&= \arg\min_{\alpha} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_j K(x_i, x_j))^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \\
&= \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha
\end{aligned}$$

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$