Foundations of Machine Learning

Kernel Classifiers

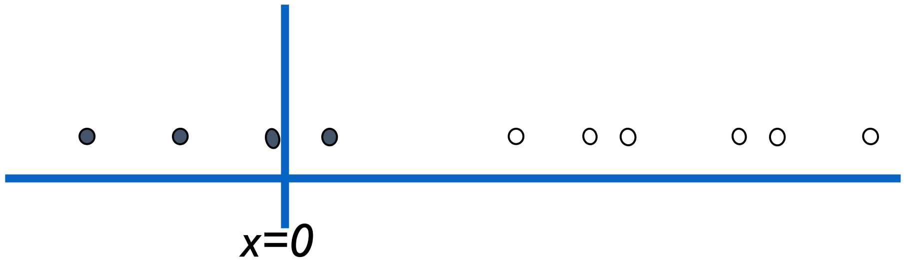
Sep 2022

Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

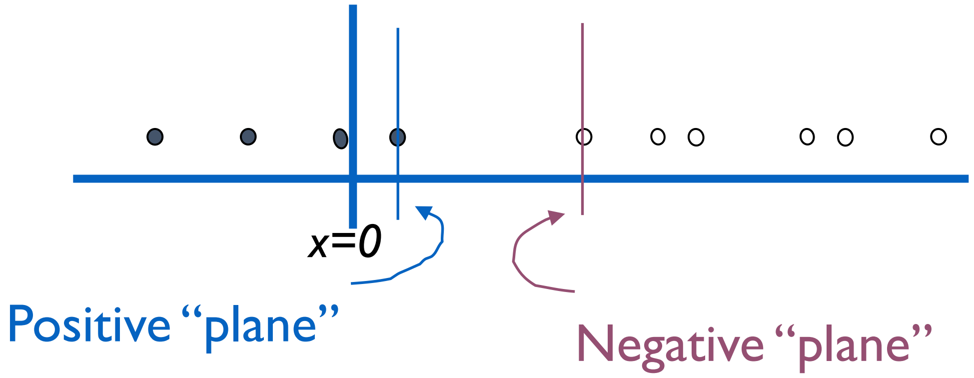
Assume we are in 1-dimension

What would SVMs do with this data?

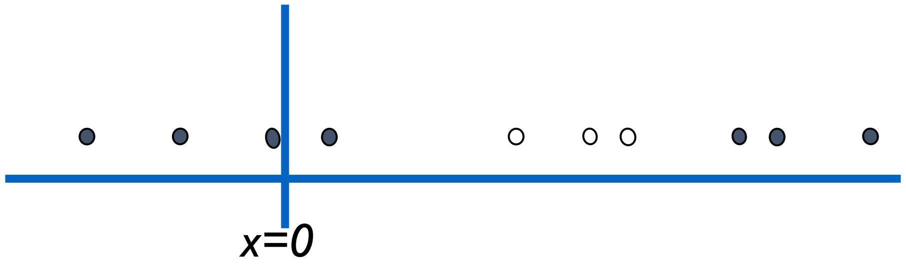


Assume we are in 1-dimension

Not a big surprise

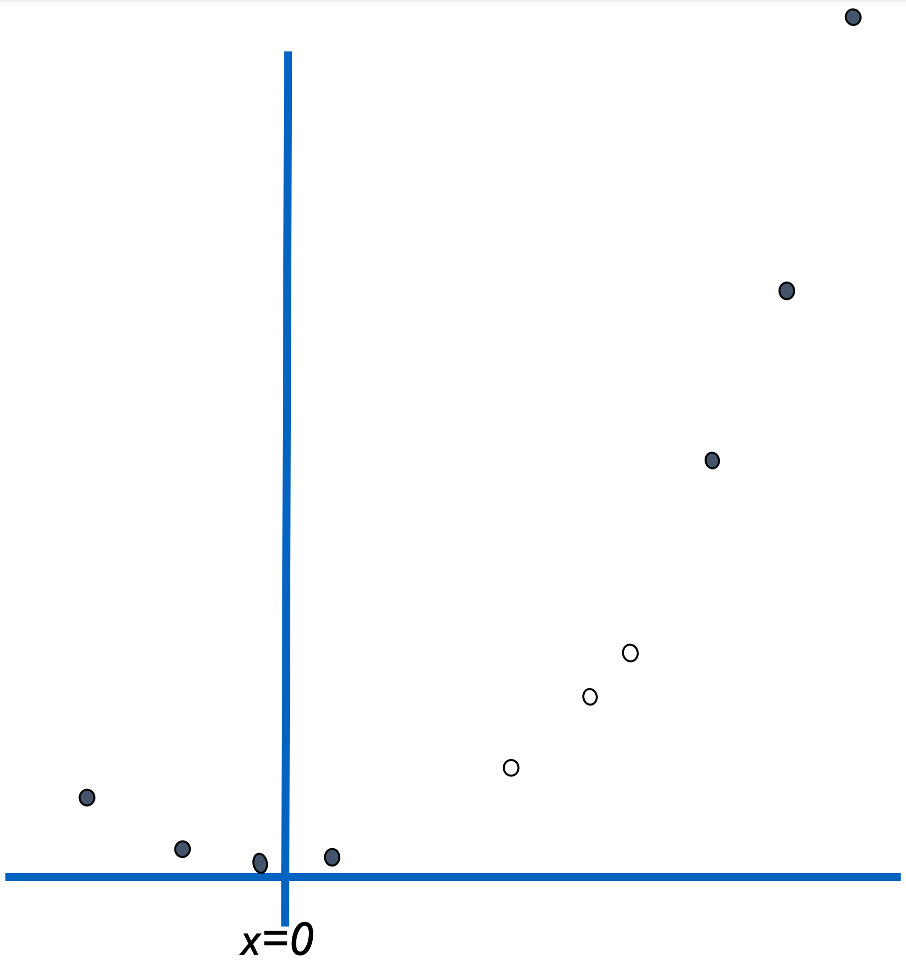


Harder 1-dimensional Dataset



What can be done about this?

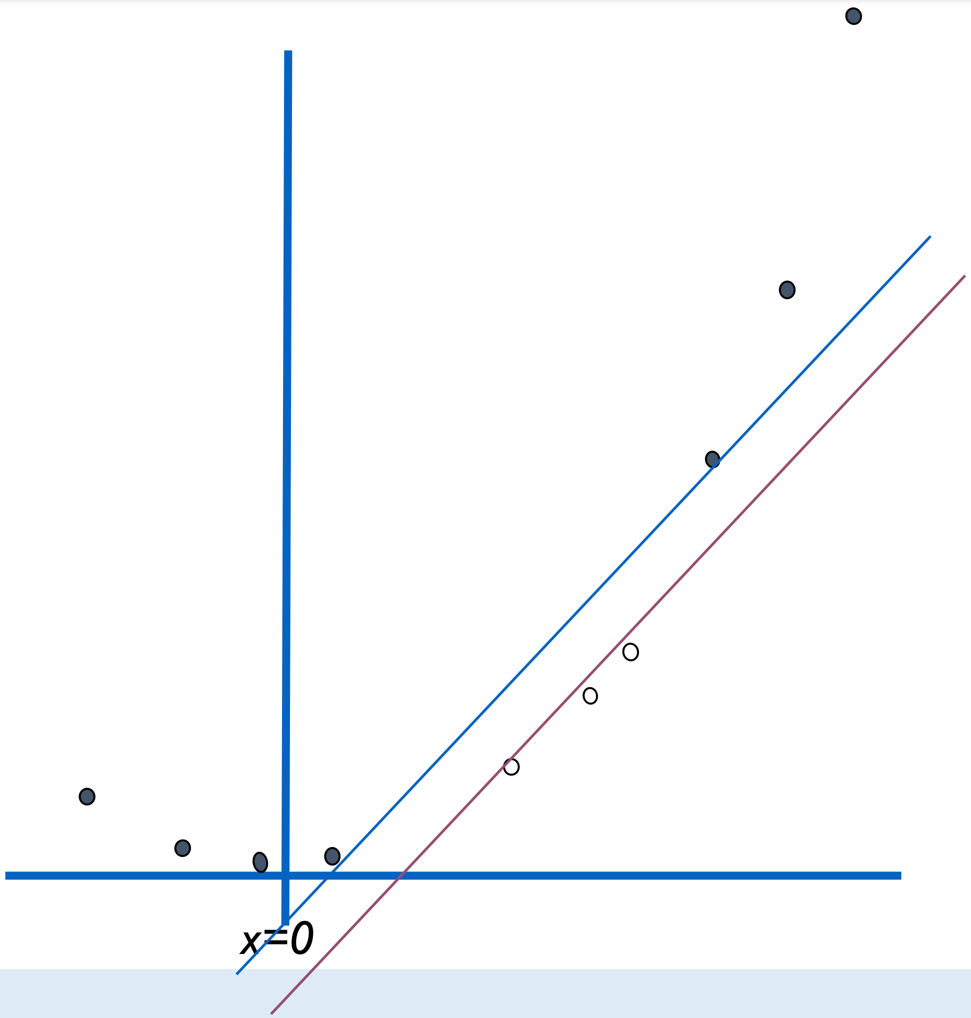
Harder 1-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, x_k^2)$$

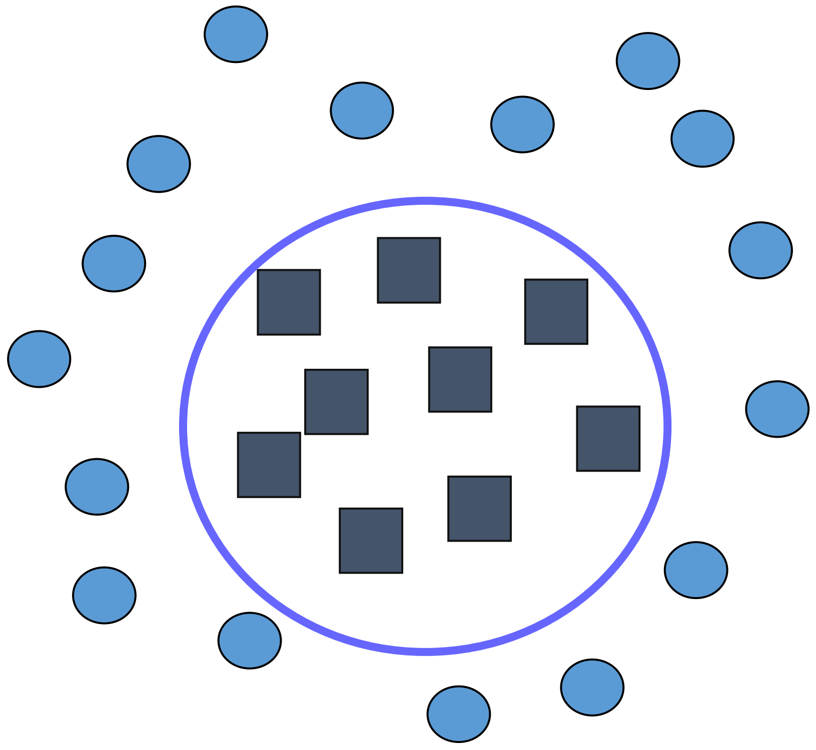
Harder 1-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, x_k^2)$$

Harder 2-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, y_k, x_k^2, y_k^2, x_k y_k)$$

Other Mapping Functions

$\mathbf{z}_k$ = (polynomial terms of $\boldsymbol{x}_k$ of degree 1 to q)

$\mathbf{z}_k$ = (radial basis functions of $\boldsymbol{x}_k$)

$$\mathbf{z}_k[j] = \varphi_j(\mathbf{x}_k) = \exp\left(-\frac{|\mathbf{x}_k - \mathbf{c}_j|^2}{2\sigma^2}\right)$$

$\mathbf{z}_k$ = (sigmoid functions of $\boldsymbol{x}_k$)

Recall: SVM Lagrangian Dual

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

subject to constraints: $0 \le \alpha_k \le c \quad \forall k \qquad \sum_{k=1}^{R} \alpha_k y_k = 0$

Once solved, we obtain w and b using:

$$\mathbf{w} = \frac{1}{2} \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

$$y_i (x_i \bullet w + b) - 1 = 0$$

$$b = -y_i (y_i (x_i \bullet w) - 1)$$

Then classify with:

***f*(x,w,*b*)** = *sign*(**w. x** + *b*)

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l))$

subject to constraints: $0 \leq \alpha_k \leq C \quad \forall k$ $\quad \sum_{k=1}^{R} \alpha_k y_k = 0$

Then compute:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

Then classify with:

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\ \mathbf{\Phi}(\mathbf{x}) + b)$

Most important change:

$\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix}$$

$$1 + \sum_{i=1}^{m} 2a_ib_i + \sum_{i=1}^{m} a_i^2b_i^2 + \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

Number of terms (assuming m input dimensions) = (m+2)-choose-2 = (m+2)(m+1)/2 = (approx) $m^2/2$

Quadratic Dot Products

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\boldsymbol{\Phi}(\mathbf{x}_k).\boldsymbol{\Phi}(\mathbf{x}_l))$

subject to constraints: $0 \leq \alpha_k \leq C$

Then compute:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \boldsymbol{\Phi}(\mathbf{x}_k)$$

We must do $R^2/2$ dot products to get this matrix ready

Assuming a quadratic polynomial kernel, each dot product requires $m^2/2$ additions and multiplications (where m is the dimension of x)

The whole thing costs $R^2 m^2 /4$.

Quadratic Dot Products

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) =$$

$$1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of interest, let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

Quadratic Dot Products

They're the same!
And this is only O(m) to compute!

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) =$$

$$1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of interest, let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2}\sum_{k=1}^{R}\sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l))$

subject to constraints: $0 \leq \alpha_k \leq C$

We must do $R^2/2$ dot products to get this matrix ready

Now, each dot product now only requires m additions and multiplications

Then compute:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

Most important change:

$\boldsymbol{x} \rightarrow \mathbf{\Phi}(\boldsymbol{x})$

Higher-Order Polynomials

Poly-nomial	f(x)	Cost to build Q_{kl} matrix traditionally	Cost if 100 dimensions	f(a).f(b)	Cost to build Q_{kl} matrix sneakily	Cost if 100 dimensions
Quadratic	All $m^2/2$ terms up to degree 2	$m^2 R^2 / 4$	$2{,}500\ R^2$	$(a.b+1)^2$	$m R^2 / 2$	$50\ R^2$
Cubic	All $m^3/6$ terms up to degree 3	$m^3 R^2 / 12$	$83{,}000\ R^2$	$(a.b+1)^3$	$m R^2 / 2$	$50\ R^2$
Quartic	All $m^4/24$ terms up to degree 4	$m^4 R^2 / 48$	$1{,}960{,}000\ R^2$	$(a.b+1)^4$	$m R^2 / 2$	$50\ R^2$

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l K(\mathbf{x}_k, \mathbf{x}_l)$

Kernel gram matrix

Subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\quad \sum_{k=1}^{R} \alpha_k y_k$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \boldsymbol{\Phi}(\mathbf{x}_k)$$

Then classify with:

$f(\mathbf{x},\mathbf{w},b) = sign(K(\mathbf{w}, \mathbf{x}) - b)$

Most important change:

$$\boldsymbol{\Phi}(\mathbf{x}_k).\boldsymbol{\Phi}(\mathbf{x}_l) \rightarrow K(\mathbf{x}_k, \mathbf{x}_l)$$

SVM Kernel Functions

- $K(\boldsymbol{a},\boldsymbol{b})=(\boldsymbol{a} \cdot \boldsymbol{b} + 1)^d$ is an example of a **kernel function** in SVM
- Beyond polynomials, there are other high-dimensional kernel functions such as:
  – Gaussian Kernel Function:

$$K(\mathbf{a},\mathbf{b}) = \exp\left(-\frac{(\mathbf{a}-\mathbf{b})^2}{2\sigma^2}\right)$$

  – Why is it an infinite-dimensional kernel?

Kernel Tricks

- Replacing dot product with a kernel function
- Not all functions are kernel functions
  - Need to be decomposable: $K(a,b) = \phi(a) \cdot \phi(b)$
- **Mercer's condition** To expand Kernel function $K(x,y)$ into a dot product, i.e. $K(x,y)=\Phi(x)\cdot\Phi(y)$, $K(x, y)$ has to be positive semi-definite function, i.e., for any function $f(x)$ whose $\int f^2(x)dx$ is finite, the following inequality holds:

$$\int dxdyf(x)K(x,y)f(y) \geq 0$$

How to choose a kernel function?

- Not easy! Remember – this depends on your data geometry
- If linear works, go with it
- RBF kernels are considered good in general, especially for images (and other smooth functions/data)
- For discrete data, chi-square kernel preferred of late (especially for histogram data)
- You can also do Multiple Kernel Learning
- Still not sure? Use cross-validation to select a kernel function from some basic options

An excellent resource: https://davejingtian.org/2010/09/10/kernel-functions-in-machine-learning-transfered/

Kernelizing other Methods

- The same kernel trick can also be applied to other methods including:
  - Kernel k-NN
  - Kernel Perceptron (we will see later)
  - Kernelized Linear Regression (we will see later)
  - Many more…

Readings

- PRML, Bishop, Chapter 7 (7.1-7.3)
- "Introduction to Machine Learning" by Ethem Alpaydin, 2nd edition, Chapters 3 (3.1-3.4), Chapter 13 (13.1-13.9)
- For kernel functions:
  - https://davejingtian.org/2010/09/10/kernel-functions-in-machine-learning-transfered/