Introduction to Machine Learning

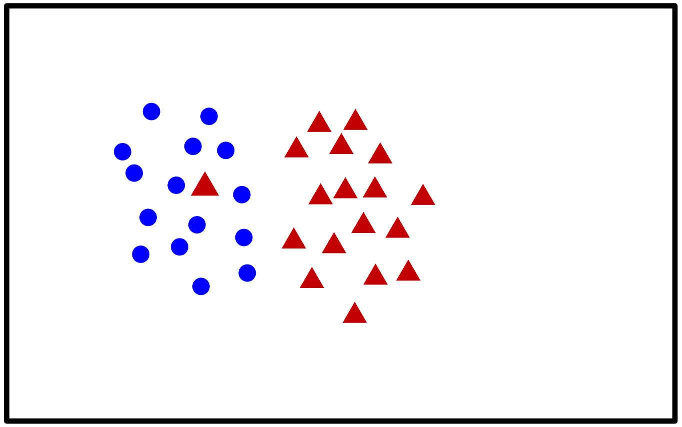
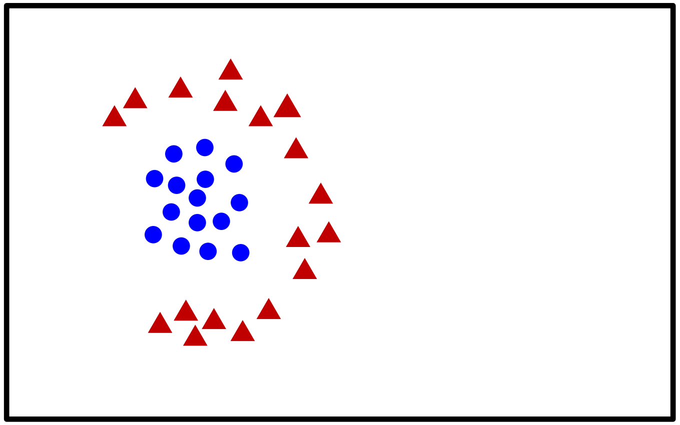
Eran Halperin, Yishay Mansour, Lior Wolf

2013-2014

Lecture 7: Kernels

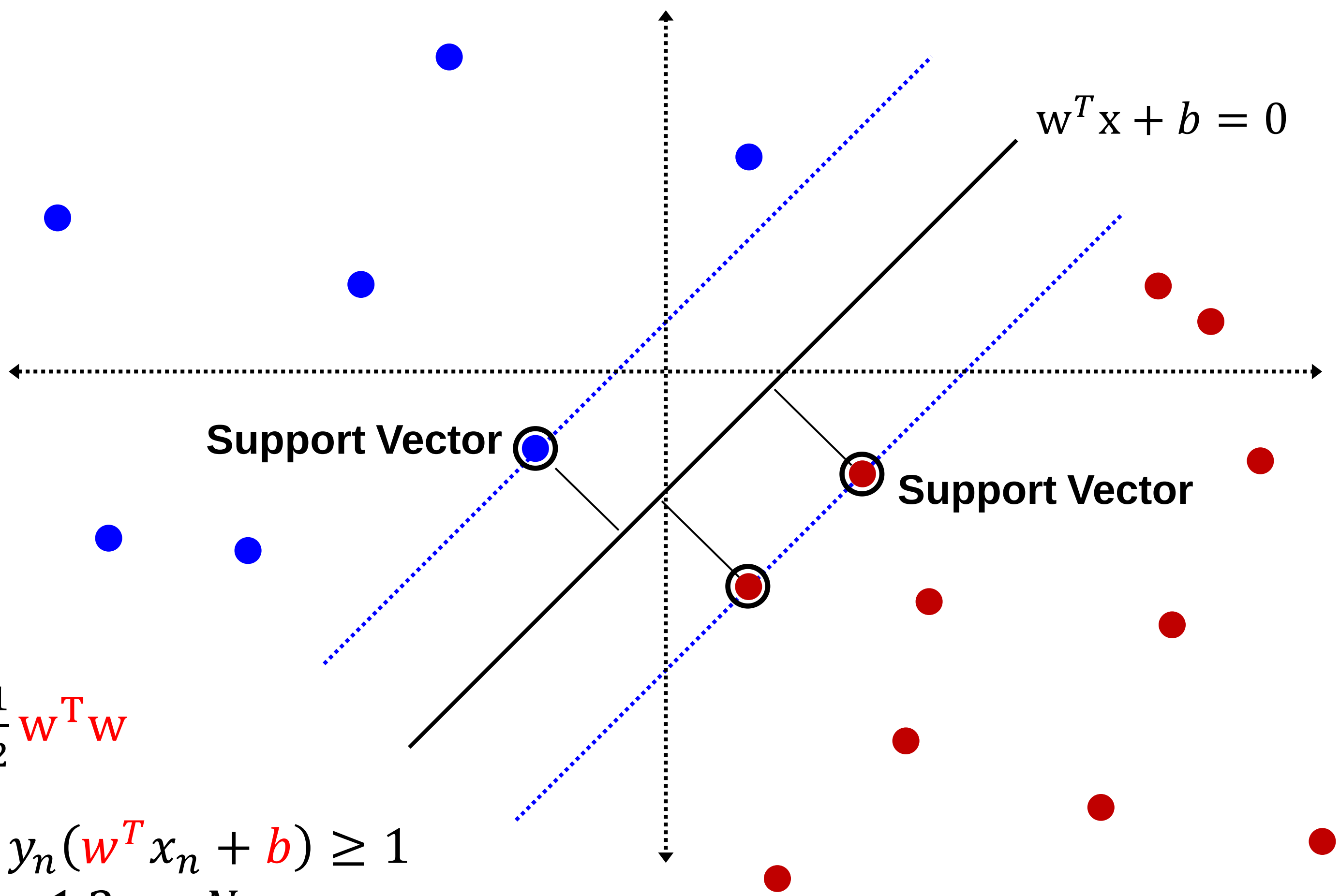
Outline

- Following discussions from last class
  - How many support vectors are there anyhow?
  - Positive definite matrices
- Support Vector machine (SVM) classifier
  - The kernel trick
  - Which kernels to use
  - Contructing kernels
  - SVD and kernel SVD

Number of support vectors

linearly separable data*: #SV $\leq$ d+1 = VC-dim



Minimize $\frac{1}{2} \mathrm{w}^{\mathrm{T}} \mathrm{w}$

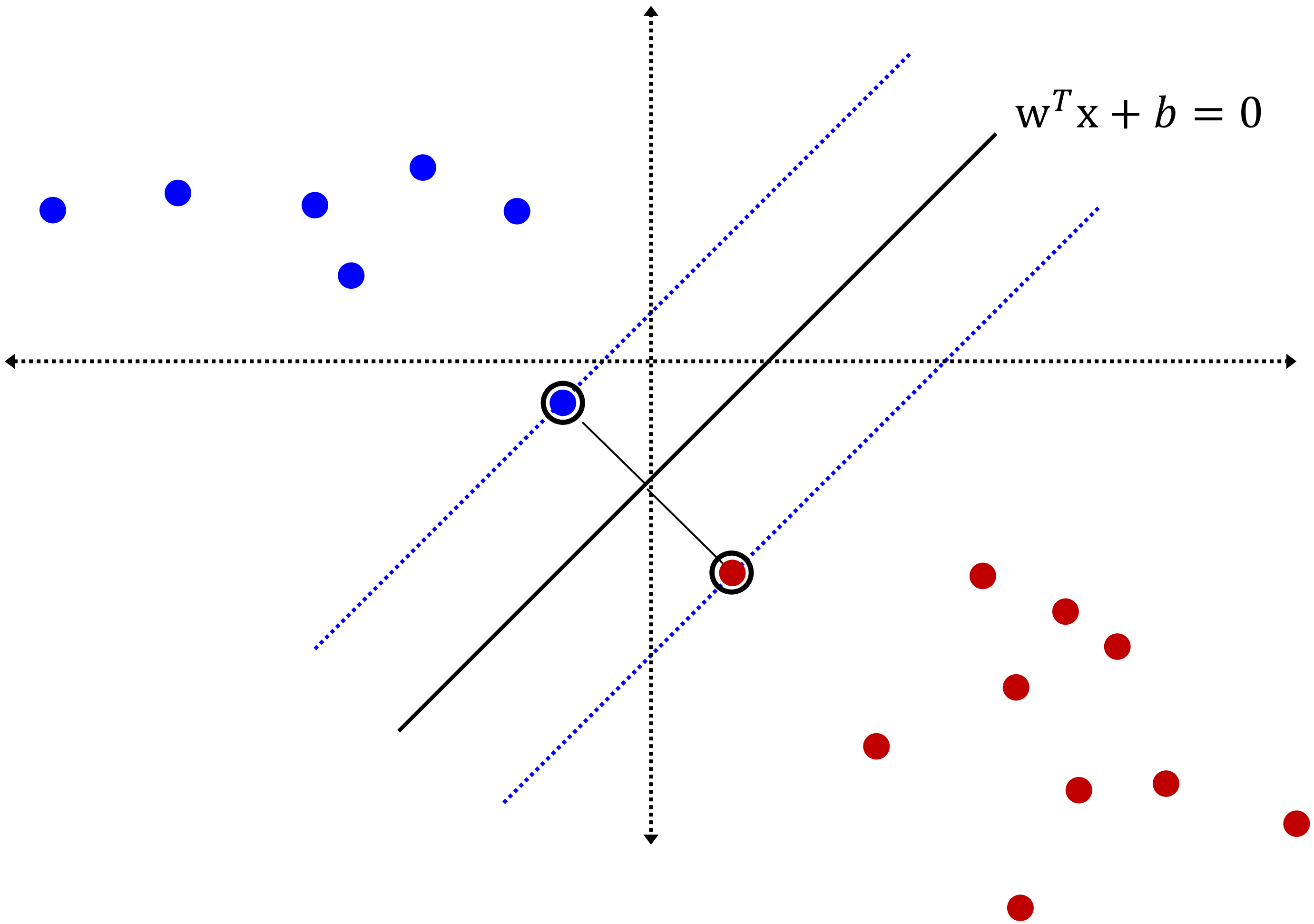
subject to $y_n(w^T x_n + b) \geq 1$

$n = 1,2, \dots, N$

$\mathrm{w} \in \mathrm{R}^d, b \in \mathrm{R}$

* Could be much more in ``degenerate cases''

#SV=2 is sometimes enough



Positive (Semi) Definite (PD/PSD) Matrices

(1) The *n*×*n* matrix ***A*** is positive definite if and only if:

$$\boldsymbol{Y}^T\boldsymbol{A}\boldsymbol{Y} > 0, \quad \forall \boldsymbol{Y} \neq \mathbf{0}$$

- The *n*×*n* matrix ***A*** is positive semi-definite if and only if:

$$\boldsymbol{Y}^T\boldsymbol{A}\boldsymbol{Y} \geq 0, \quad \forall \boldsymbol{Y} \neq \mathbf{0}$$

semi

(2) $\boldsymbol{A}$ is positive definite $\iff \exists \boldsymbol{P}$ s.t. $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{P}^T$, $\cancel{|\boldsymbol{P}| \neq \mathbf{0}}$

$\boldsymbol{A}$ is positive definite $\Rightarrow$ it is symmetric

Eigenvalues of PD Matrices

- Given the $n \times n$ matrix $\boldsymbol{A}$, there are n eigenvalues λ and vectors $\boldsymbol{X} \neq \boldsymbol{0}$ where

$$\boldsymbol{A}\boldsymbol{X} = \lambda \boldsymbol{X}$$

$$\begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \cdots & \boldsymbol{X}_n \end{bmatrix}^T \boldsymbol{A} \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \cdots & \boldsymbol{X}_n \end{bmatrix} = \mathsf{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- If $\lambda_i > 0 \;\; \forall i \in [1, n] \Leftrightarrow \boldsymbol{A}$ is positive definite
- If $\lambda_i \geq 0 \;\; \forall i \in [1, n] \Leftrightarrow \boldsymbol{A}$ is positive semi-definite

$\boldsymbol{A}$ is positive definite $\Rightarrow |\boldsymbol{A}| > 0$

Positive (Semi) Definite (PD/PSD) Matrices

The $n \times n$ matrix $\boldsymbol{A}$ is positive definite if and only if:

1. $\boldsymbol{Y}^T \boldsymbol{A} \boldsymbol{Y} > 0, \quad \forall \boldsymbol{Y} \neq \mathbf{0}$

2. $\exists \boldsymbol{P}$ s.t. $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{P}^T, |\boldsymbol{P}| \neq \mathbf{0}$

3. $\lambda_i \geq 0 \;\; \forall i \in [1, n]$

The dual formulation

$$\min_{\propto} \frac{1}{2} \propto^{\mathrm{T}} \underbrace{\begin{bmatrix} y_1 y_1 \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_1 & y_1 y_2 \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_2 & \dots & y_1 y_N \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_N \\ y_2 y_1 \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_1 & y_2 y_2 \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_2 & \dots & y_2 y_N \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_N \\ \dots & \dots & \dots & \dots \\ y_N y_1 \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_1 & y_N y_2 \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_2 & \dots & y_N y_N \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_N \end{bmatrix}}_{\text{M}} \propto + \underbrace{\left(-1^{\mathrm{T}}\right)}_{\text{linear}} \propto$$

subject to $\underbrace{y^{\mathrm{T}} \propto = 0}_{\text{linear constraint}}$

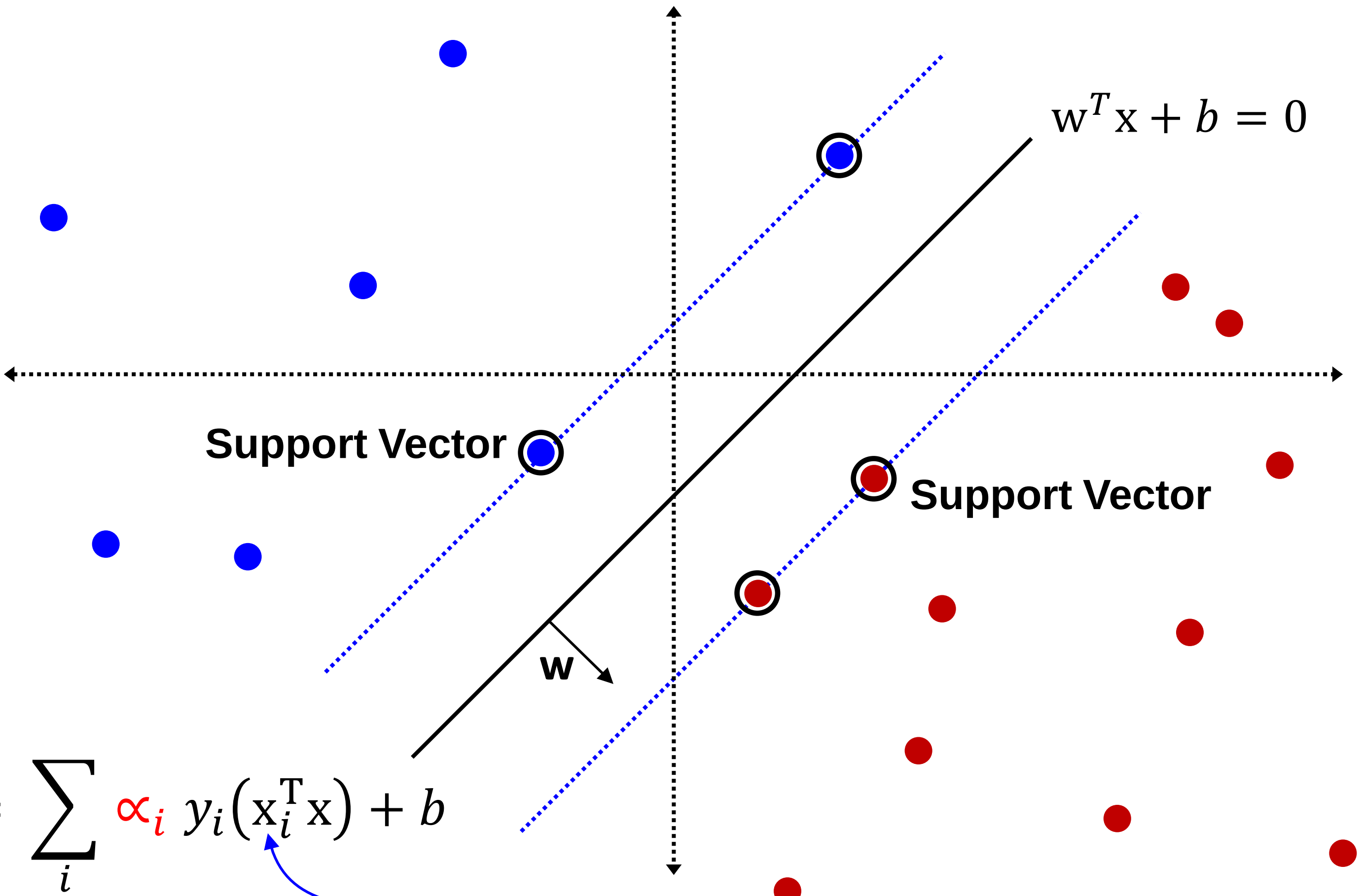
$$\underbrace{0}_{\text{lower bounds}} \quad \le \quad \propto \quad \le \quad \underbrace{C}_{\text{upper bounds}}$$

The dual formulation

$$\min_{\propto} \frac{1}{2} \propto^{\mathrm{T}} \underbrace{\begin{bmatrix} y_1 y_1 \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_1 & y_1 y_2 \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_2 & \dots & y_1 y_N \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_N \\ y_2 y_1 \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_1 & y_2 y_2 \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_2 & \dots & y_2 y_N \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_N \\ \dots & \dots & \dots & \dots \\ y_N y_1 \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_1 & y_N y_2 \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_2 & \dots & y_N y_N \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_N \end{bmatrix}}_{\mathrm{M=A^tA}} \propto + \underbrace{\left(-1^{\mathrm{T}}\right)}_{\text{linear}} \propto$$

$$\begin{bmatrix} y_1 y_1 \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_1 & y_1 y_2 \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_2 & \dots & y_1 y_N \mathrm{x}_1^{\mathrm{T}} \mathrm{x}_N \\ y_2 y_1 \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_1 & y_2 y_2 \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_2 & \dots & y_2 y_N \mathrm{x}_2^{\mathrm{T}} \mathrm{x}_N \\ \dots & \dots & \dots & \dots \\ y_N y_1 \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_1 & y_N y_2 \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_2 & \dots & y_N y_N \mathrm{x}_N^{\mathrm{T}} \mathrm{x}_N \end{bmatrix} = [y_1 \mathrm{x}_1, y_2\, \mathrm{x}_2, \dots, y_N\, \mathrm{x}_N]^t [y_1 \mathrm{x}_1, y_2\, \mathrm{x}_2, \dots, y_N\, \mathrm{x}_N]$$

Support Vector Machine



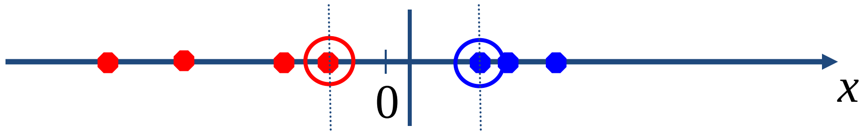
$$f(x) = \sum_i \alpha_i \, y_i(\mathrm{x}_i^{\mathrm{T}}\mathrm{x}) + b$$

support vectors

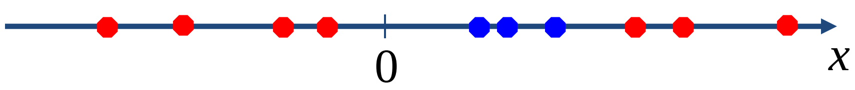
Slide credit: A. Zisserman

Nonlinear SVMs

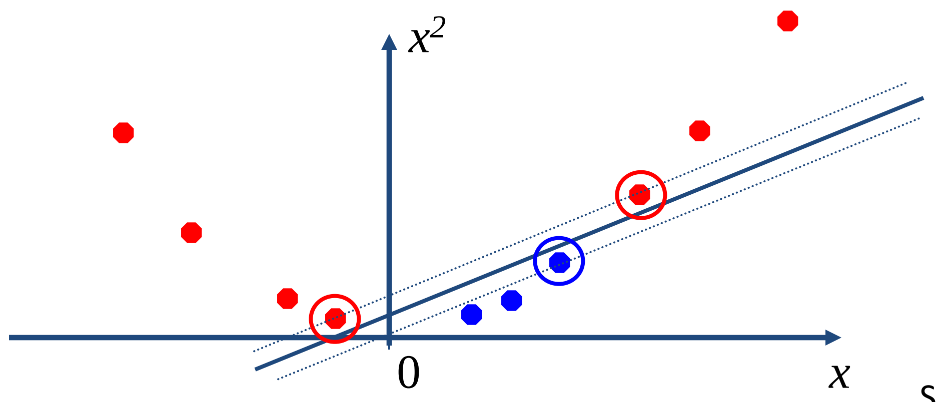
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



- We can map it to a higher-dimensional space:



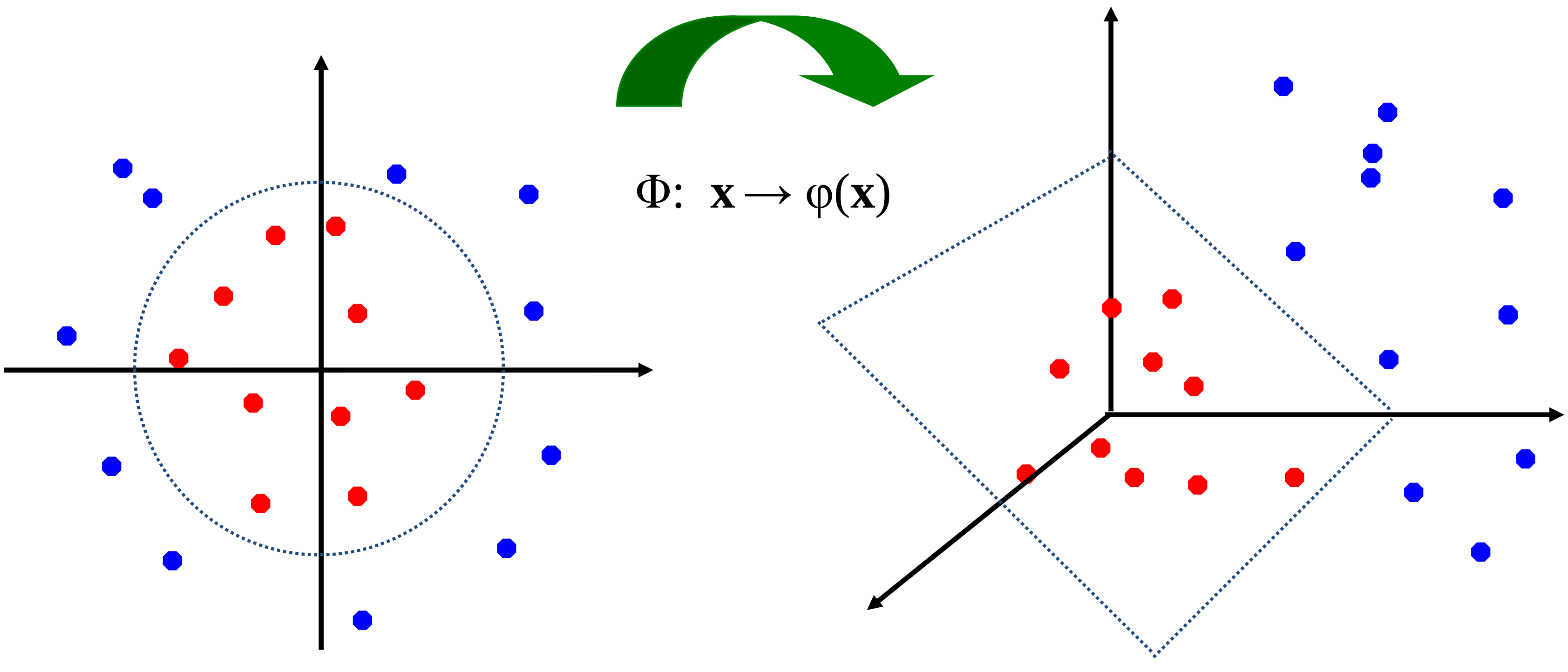
Slide credit: Andrew Moore

Another example (2D)

Slide credit: Jitendra Malik

Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

A potential problem

- If we map the input vectors into a very high-dimensional feature space, optimizing the SVM and even classification might become computationally intractable
  - The mathematics is the same
  - The vectors have a huge number of components
  - Taking the dot product of two vectors is very expensive
  - What would happen to the primal QP?

Minimize $\frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w}$

subject to $y_n(w^T\phi(x_n) + b) \geq 1$

$n = 1,2, \ldots, N$

$\mathrm{w} \in \mathrm{R}^{?}, b \in \mathrm{R}$

A potential problem

- If we map the input vectors into a very high-dimensional feature space, optimizing the SVM and even classification might become computationally intractable
  - The mathematics is the same
  - The vectors have a huge number of components
  - Taking the dot product of two vectors is very expensive
  - What would happen to the primal QP?
  - What would happen to the dual QP?

$$\min_{\propto} \frac{1}{2} \propto^{\mathrm{T}} \begin{bmatrix} y_1 y_1 \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_1) & y_1 y_2 \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_2) & \dots & y_1 y_N \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_\mathrm{N}) \\ y_2 y_1 \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_1) & y_2 y_2 \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_2) & \dots & y_2 y_N \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_\mathrm{N}) \\ \dots & \dots & \dots & \dots \\ y_N y_1 \phi(\mathrm{x}_\mathrm{N})^T \phi(\mathrm{x}_1) & y_N y_2 \phi(\mathrm{x}_\mathrm{N})^T \phi(\mathrm{x}_2) & \dots & y_N y_N \phi(\mathrm{x}_N)^T \phi(\mathrm{x}_\mathrm{N}) \end{bmatrix} \propto + \left(-1^{\mathrm{T}}\right) \propto$$

A potential problem

- If we map the input vectors into a very high-dimensional feature space, optimizing the SVM and even classification might become computationally intractable
  - The mathematics is the same
  - The vectors have a huge number of components
  - Taking the dot product of two vectors is very expensive
  - What would happen to the primal QP?
  - What would happen to the dual QP?
  - And during classification?

$$f(x) = \sum_i \propto_i y_i(\phi(\mathrm{x}_i)^T \mathrm{x}) + b$$

Dual decision rule

$$f(\mathrm{x}) = \mathrm{w}^T \mathrm{x} + b$$

Primal decision rule

Where is the ϕ “feature” space?

Dual optimization:

$$\min_{\propto} \frac{1}{2} \propto^{\mathrm{T}} \begin{bmatrix} y_1 y_1 \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_1) & y_1 y_2 \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_2) & \dots & y_1 y_N \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_\mathrm{N}) \\ y_2 y_1 \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_1) & y_2 y_2 \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_2) & \dots & y_2 y_N \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_\mathrm{N}) \\ \dots & \dots & \dots & \dots \\ y_N y_1 \phi(\mathrm{x}_\mathrm{N})^T \phi(\mathrm{x}_1) & y_N y_2 \phi(\mathrm{x}_\mathrm{N})^T \phi(\mathrm{x}_2) & \dots & y_N y_N \phi(\mathrm{x}_N)^T \phi(\mathrm{x}_\mathrm{N}) \end{bmatrix} \propto +(-1^{\mathrm{T}}) \propto$$

subject to $y^{\mathrm{T}} \propto = 0 \qquad 0 \leq \propto \leq C$

w?

$$w = \sum_{i \in SV} \propto_i y_i \phi(\mathrm{x}_i) \qquad f(x) = \sum_i \propto_i y_i (\phi(\mathrm{x}_i)^T \phi(\mathrm{x})) + b$$

b?

$$f(x) = \sum_i \propto_i y_i (\phi(\mathrm{x}_i)^T \phi(\mathrm{x})) + b = \mathrm{y}$$

The "Kernel trick"

Linear SVM

$$f(x) = \sum_i \propto_i y_i(\mathrm{x}_i^{\mathrm{T}}\mathrm{x}) + b$$

Non-linear SVM

$$f(x) = \sum_i \propto_i y_i(\phi(\mathrm{x}_i)^T\phi(\mathrm{x})) + b$$

Define the "kernel function" K

$$K(x', x'') = \phi(\mathrm{x}')^T\phi(\mathrm{x}'')$$

then

$$f(x) = \sum_i \propto_i y_i K(\mathrm{x}_i,^T \mathrm{x}) + b$$

Where is the ϕ "feature" space?

Dual optimization:

$$\min_{\propto} \frac{1}{2} \propto^{\mathrm{T}} \begin{bmatrix} y_1 y_1 \overbrace{\boxed{\phi(\mathrm{x}_1)^T \phi(\mathrm{x}_1)}}^{K} & y_1 y_2 \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_2) & \dots & y_1 y_N \phi(\mathrm{x}_1)^T \phi(\mathrm{x}_\mathrm{N}) \\ y_2 y_1 \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_1) & y_2 y_2 \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_2) & \dots & y_2 y_N \phi(\mathrm{x}_2)^T \phi(\mathrm{x}_\mathrm{N}) \\ \dots & \dots & \dots & \dots \\ y_N y_1 \phi(\mathrm{x}_\mathrm{N})^T \phi(\mathrm{x}_1) & y_N y_2 \phi(\mathrm{x}_\mathrm{N})^T \phi(\mathrm{x}_2) & \dots & y_N y_N \phi(\mathrm{x}_N)^T \phi(\mathrm{x}_\mathrm{N}) \end{bmatrix} \propto + (-1^{\mathrm{T}}) \propto$$

subject to $\quad y^{\mathrm{T}} \propto = 0 \qquad 0 \quad \leq \quad \propto \quad \leq \quad C$

w?

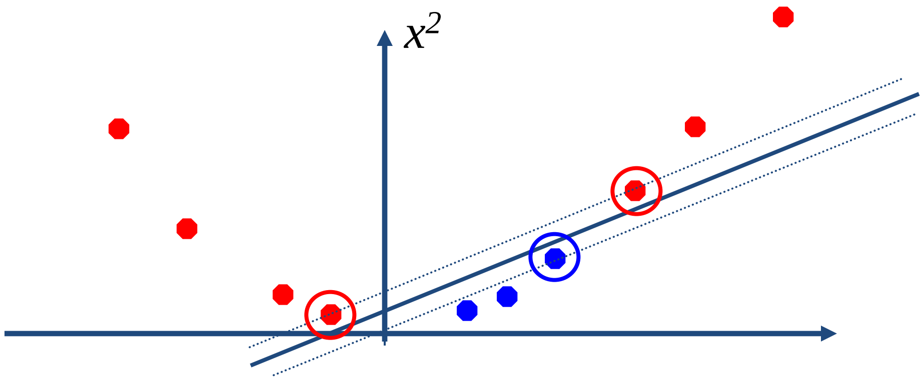
$$w = \sum_{i \in SV} \propto_i y_i \phi(\mathrm{x}_i) \qquad\qquad f(x) = \sum_i \propto_i y_i (\overbrace{\boxed{\phi(\mathrm{x}_i)^T \phi(\mathrm{x})}}^{K}) + b$$

b?

$$f(x) = \sum_i \propto_i y_i (\overbrace{\boxed{\phi(\mathrm{x}_i)^T \phi(\mathrm{x})}}^{K}) + b = \mathrm{y}$$

Nonlinear kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$



$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

Computing K(x,x') without explicitly computing $\phi(x)$

For example: 2nd order polynomial kernel in 2d

$$K(x, x') = (1 + x^t x')^2 = (1 + x_1 x_1' + x_2 x_2')^2 =$$
$$= 1 + x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_1' + 2x_2 x_2' + 2x_1 x_1' x_2 x_2'$$

$$\mathrm{K}(\mathrm{x}, \mathrm{x}') = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_1 x_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 \\ x'^2_1 \\ x'^2_2 \\ \sqrt{2}x'_1 \\ \sqrt{2}x'_2 \\ \sqrt{2}x'_1 x'_2 \end{bmatrix} = \phi(x)^T \phi(x)$$

Popular kernels

$$K(x', x'') = \phi(\mathrm{x}')^T \phi(\mathrm{x}'')$$

Name	params	Kernel eqation K(x',x'')	Non-linear mapping $\phi(x)$
Linear		$(x')^t x''$	x
Polinomial	D	$(1+(x')^t x'')^D$	All polynomials up to degree D in the elements of the vector x
Gaussian==RBF	σ	$\exp(-\|\|x'-x''\|\|^2/(2\sigma^2))$	Infinite dimensional vector

Popular kernels

$$K(x', x'') = \phi(\mathrm{x}')^T \phi(\mathrm{x}'')$$

Name	params	Kernel eqation K(x',x'')	Non-linear mapping $\phi(x)$
Linear		$(x')^t x''$	x
Polinomial	D	$(1+(x')^t x'')^D$	All polynomials up to degree D in the elements of the vector x
Gaussian==RBF	σ	$\exp(-\|\|x'-x''\|\|^2/(2\sigma^2))$	Infinite dimensional vector

$$K(x,y) = \left(\sum_{i=1}^{n} x_i y_i + 1\right)^2 = \sum_{i=1}^{n} x_i^2 y_i^2 + \sum_{i=2}^{n}\sum_{j=1}^{i-1} \sqrt{2} x_i y_i \sqrt{2} x_j y_j + \sum_{i=1}^{n} \sqrt{2} x_i \sqrt{2} y_i + 1$$

$$\varphi(x) = \langle x_n^2, \ldots, x_1^2, \sqrt{2} x_n x_{n-1}, \ldots, \sqrt{2} x_n x_1, \sqrt{2} x_{n-1} x_{n-2}, \ldots, \sqrt{2} x_{n-1} x_1, \ldots, \sqrt{2} x_2 x_1, \sqrt{2} x_n, \ldots, \sqrt{1} x_1, 1\rangle$$

Complexity does not depend on D! (take log multiply and exponent)

Popular kernels

$$K(x', x'') = \phi(\mathrm{x}')^T \phi(\mathrm{x}'')$$

Name	params	Kernel eqation K(x',x'')	Non-linear mapping $\phi(x)$
Linear		$(x')^t x''$	x
Polinomial	D	$(1+(x')^t x'')^D$	All polynomials up to degree D in the elements of the vector x
Gaussian==RBF	σ	$\exp(-\|\|x'-x''\|\|^2/(2\sigma^2))$	Infinite dimensional vector

$$\overline{\overline{K}}_{ij} = K\left(x_i, x_j\right)$$

$$\overline{\overline{K}} = [\phi(x_1)\ \phi(x_2)\ \ldots \phi(x_N)]^t\ [\phi(x_1)\ \phi(x_2)\ \ldots \phi(x_N)]$$

$$rank(\overline{\overline{K}}) = \mathrm{rank}([\phi(x_1)\ \phi(x_2)\ \ldots \phi(x_N)])$$

$$take\ \overline{\overline{K}} = \begin{bmatrix} 1 & \epsilon & \epsilon \\ \epsilon & 1 & \epsilon \\ \epsilon & \epsilon & \ddots \end{bmatrix}$$

Proper kernels

- Symmetric

$K(x_i, x_j) = K(x_j, x_i)$

- Positive definite kernel

$\forall N, \forall x_1, \dots, x_N, \forall c \in R^N, \qquad c^t \overline{\overline{\overline{K}}} c > 0$

- But in practice, we don’t necessarily need PSD kernels..

Constructing proper kernels

- $K_3(x',x'') = K_1(x',x'') + K_2(x',x'')$

$$\phi_3(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \end{bmatrix}$$

- $K_3(x',x'') = K_1(x',x'') * K_2(x',x'')$

$(\phi_1(x')^t\,\phi_1(x''))(\phi_2(x'')^t\,\phi_2(x')) = \mathrm{tr}(\phi_1(x')^t\,\phi_1(x'')\phi_2(x'')^t\,\phi_2(x'))=$

$\mathrm{tr}((\phi_2(x')\phi_1(x')^t)(\,\phi_1(x'')\phi_2(x'')^t)) = \langle \mathrm{VEC}(\phi_2(x')\phi_1(x')^t), \mathrm{VEC}(\phi_2(x'')\phi_1(x'')^t)\rangle$

Distances and kernels

- Suppose we want to apply kNN in kernel space

  In input space:

  $$||a-b||^2 = (a-b)^t(a-b) = a^t a - 2a^t b + b^t b$$

  Similarly, in "feature space"

  $$||\phi(a)-\phi(b)||^2 = (\phi(a)-\phi(b))^t(\phi(a)-\phi(b)) =$$
  $$= \phi(a)^t\phi(a) - 2\phi(a)^t\phi(b) + \phi(b)^t\phi(b) =$$

  = K(a,a) -2K(a,b)+K(b,b)

Generalized Gaussian kernel for histograms:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2 \right)$$

- L1 distance: $D(h_1, h_2) = \sum_{i=1}^{N} | h_1(i) - h_2(i) |$
- L2 distance: $D^2(h_1, h_2) = \sum_{i=1}^{N} \left(h_1(i) - h_2(i) \right)^2$
- L-inf distance: $D(h_1, h_2) = \max_{1 \le i \le N} | h_1(i) - h_2(i) |$
- χ^2 distance: $D \ (h_1, h_2) = \sum_{i=1}^{N} \frac{\left(h_1(i) - h_2(i) \right)^2}{h_1(i) + h_2(i)}$
- Hellinger distance: $D^2(h_1, h_2) = \sum_{i=1}^{N} \left(\sqrt{h_1(i)} - \sqrt{h_2(i)} \right)^2$
- Mahalanobis distance: $D^2(h_1, h_2) = \left(h_1 - h_2 \right)^T S^{-1} \left(h_1 - h_2 \right)$

The Intersection Kernel

Histogram Intersection kernel between histograms *a*, *b*

$$K(a,b) = \sum_{i=1}^{n} min(a_i, b_i) \qquad \begin{array}{r} a_i \geq 0 \\ b_i \geq 0 \end{array}$$

K small -> *a*, *b* are different
K large -> *a*, *b* are similar

Intro. by Swain and Ballard 1991 to compare color histograms.
Odone et al 2005 proved positive definiteness.

Slide credit: Jitendra Malik

Demonstration of Positive Definiteness

Histogram Intersection kernel between histograms *a*, *b*

$$K(a,b) = \sum_{i=1}^{n} min(a_i, b_i) \qquad \begin{matrix} a_i \geq 0 \\ b_i \geq 0 \end{matrix}$$

To see that $min(a_i, b_i)$ is positive definite,

represent *a*, *b* in "Unary", *n* is written as *n* ones in a row:

$$min(a_i, b_i) = \left\langle a_{i\,\mathrm{unary}}, b_{i\,\mathrm{unary}} \right\rangle$$

$$min(3,5) = \langle (1,1,1,0,0), (1,1,1,1,1) \rangle = 3$$

Slide credit: Jitendra Malik

The Trick

~~#support vectors x #dimensions~~

log(#support vectors) x #dimensions

Decision function is $\mathrm{sign}\,(h(x))$ where:

$$
\begin{aligned}
h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}
$$

Just sort the support vector values in each coordinate, and pre-compute

To evaluate, find position of x_i in the sorted support vector values x_i^j (cost: log #sv) look up values, multiply & add

Slide credit: Jitendra Malik

Singular Value Decomposition (SVD)

- Handy mathematical technique that has application to many problems
- Given any $m \times n$ matrix **A**, it decomposes it to three matrices **U**, **V**, and **W** such that

$$\mathbf{A} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T$$

U is $m \times n$ and orthonormal

W is $n \times n$ and diagonal

V is $n \times n$ and orthonormal

SVD

Matlab: [U,W,V]=svd(A,0)

$$\begin{pmatrix} \\ \\ & \mathbf{A} & \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ & \mathbf{U} & \\ \\ \\ \end{pmatrix} \begin{pmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_n \end{pmatrix} \begin{pmatrix} \\ & \mathbf{V} & \\ \\ \end{pmatrix}^{\mathrm{T}}$$

- The $w_i > 0$ are called the singular values of **A** and are sorted
- If **A** is singular, some of the w_i will be 0
- *rank*(**A**) = number of nonzero w_i
- SVD is unique (unless some w_i are equal)

SVD and Inverses

- $\mathbf{A}^{-1} = (\mathbf{V}^T)^{-1}\,\mathbf{W}^{-1}\,\mathbf{U}^{-1} = \mathbf{V}\,\mathbf{W}^{-1}\,\mathbf{U}^T$
  - Using fact that inverse = transpose for orthogonal matrices
  - Note: $\mathbf{W}^{-1}$ is also diagonal with elements one over those of $\mathbf{W}$

- Pseudoinverse: if $w_i=0$, set $1/w_i$ to 0 (!)
  - Defined for all (even non-square, singular, etc.) matrices
  - Equal to $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ if $\mathbf{A}^T\mathbf{A}$ invertible

- Solving $\mathbf{Ax}=\mathbf{b}$ by least squares
  $\mathbf{x}$=pinv($\mathbf{A}$)*$\mathbf{b}$

SVD and Eigenvectors

- Let **A**=**UWV**^{T}, and let x_i be i^{th} column of **V**
- Consider **A**^{T}**A**x_i:

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}x_i = \mathbf{V}\mathbf{W}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}}x_i = \mathbf{V}\mathbf{W}^2\mathbf{V}^{\mathrm{T}}x_i = \mathbf{V}\mathbf{W}^2\begin{pmatrix}0\\ \vdots\\ 1\\ \vdots\\ 0\end{pmatrix} = \mathbf{V}\begin{pmatrix}0\\ \vdots\\ {w_i}^2\\ \vdots\\ 0\end{pmatrix} = {w_i}^2 x_i$$

- So elements of **W** are sqrt(eigenvalues) and columns of **V** are eigenvectors of **A**^{T}**A**
- Similarly, the columns of U are the eigenvectors of **AA**^{T}**,** and **diag(W)** are sqrt(eigenvalues **AA**^{T})

Kernel SVD

Let A=$[\phi(x_1), \phi(x_2), \dots, \phi(x_N)]$

How do we compute the SVD decomposition U,W, V^T?

Mental framework – A is of size $\infty \times N$

AA^t is of size $\infty \times \infty$ and U is also $\infty \times N$

but A^tA is of size $N \times N$ and we can compute V and W

$A = UWV^t \rightarrow U = AVW^{-1}$

and we can compute, e.g., $U^t\phi(x) = W^{-1}V^tA^t\phi(x)$

$$= W^{-1}V^t \begin{bmatrix} k(x_1, x) \\ k(x_2, x) \\ \vdots \end{bmatrix}$$

Applications: (1) $\mathbf{A}^{-1} = \mathbf{V}\,\mathbf{W}^{-1}\,\mathbf{U}^T$

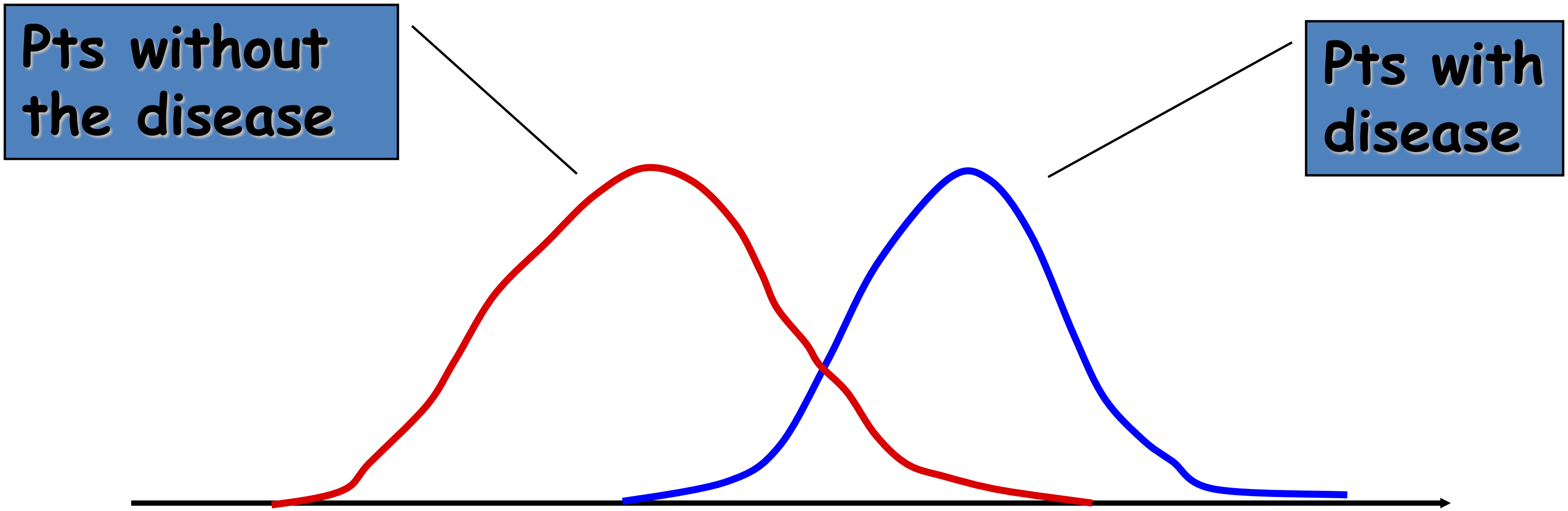
(2) kernel PCA

True label vs. classifier result

Real label \ classifier	Prediction=-1	Prediction=1
No disease (D = -1)	☺ True negative	X False positive
Disease (D = +1)	X Miss	☺ True positive

Original slide credit: Darlene Goldstein

Specific Example

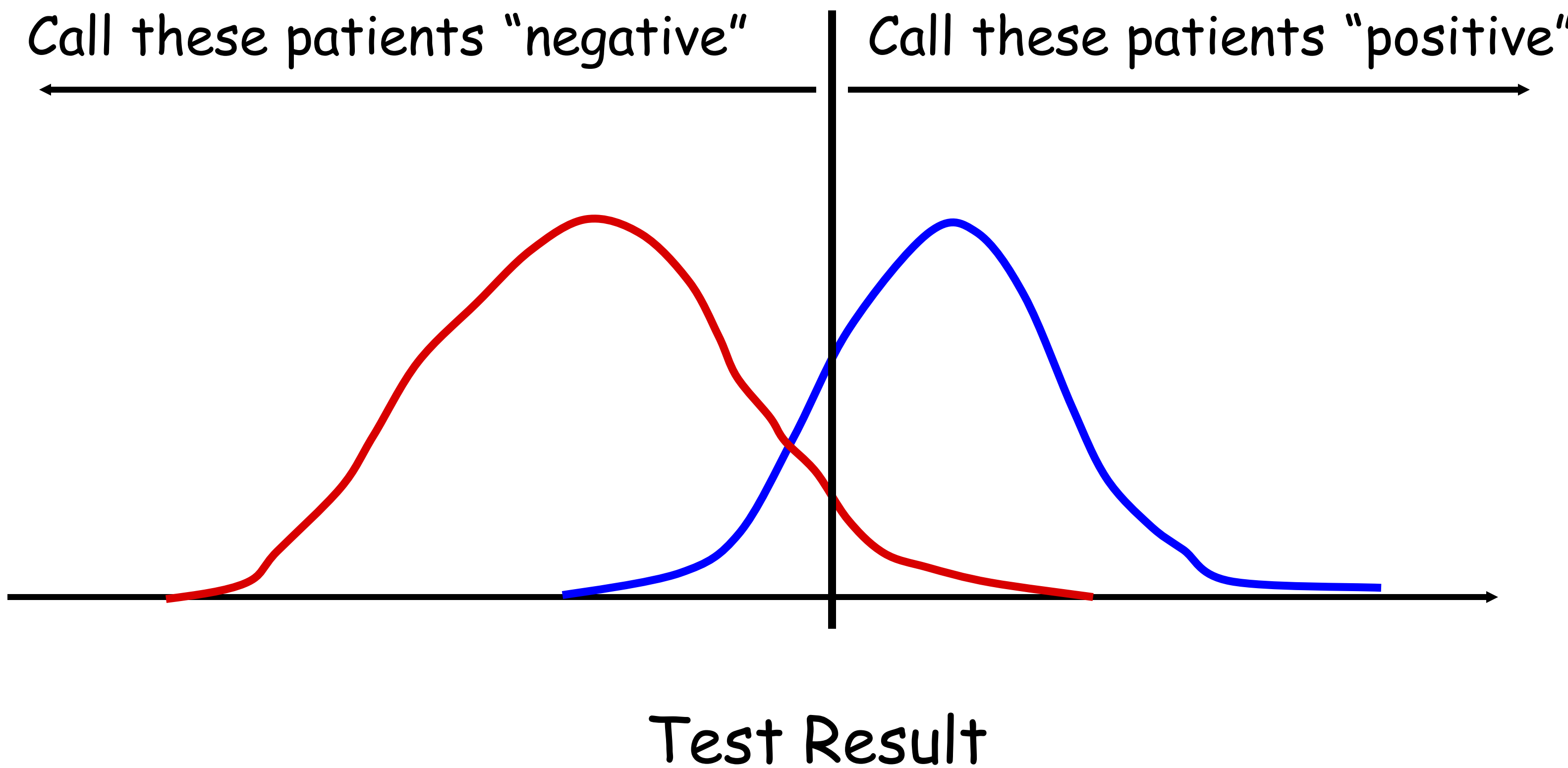


Test Result

$$f(\mathrm{x}) = \mathrm{w}^{\mathrm{t}}\mathrm{x} + b$$

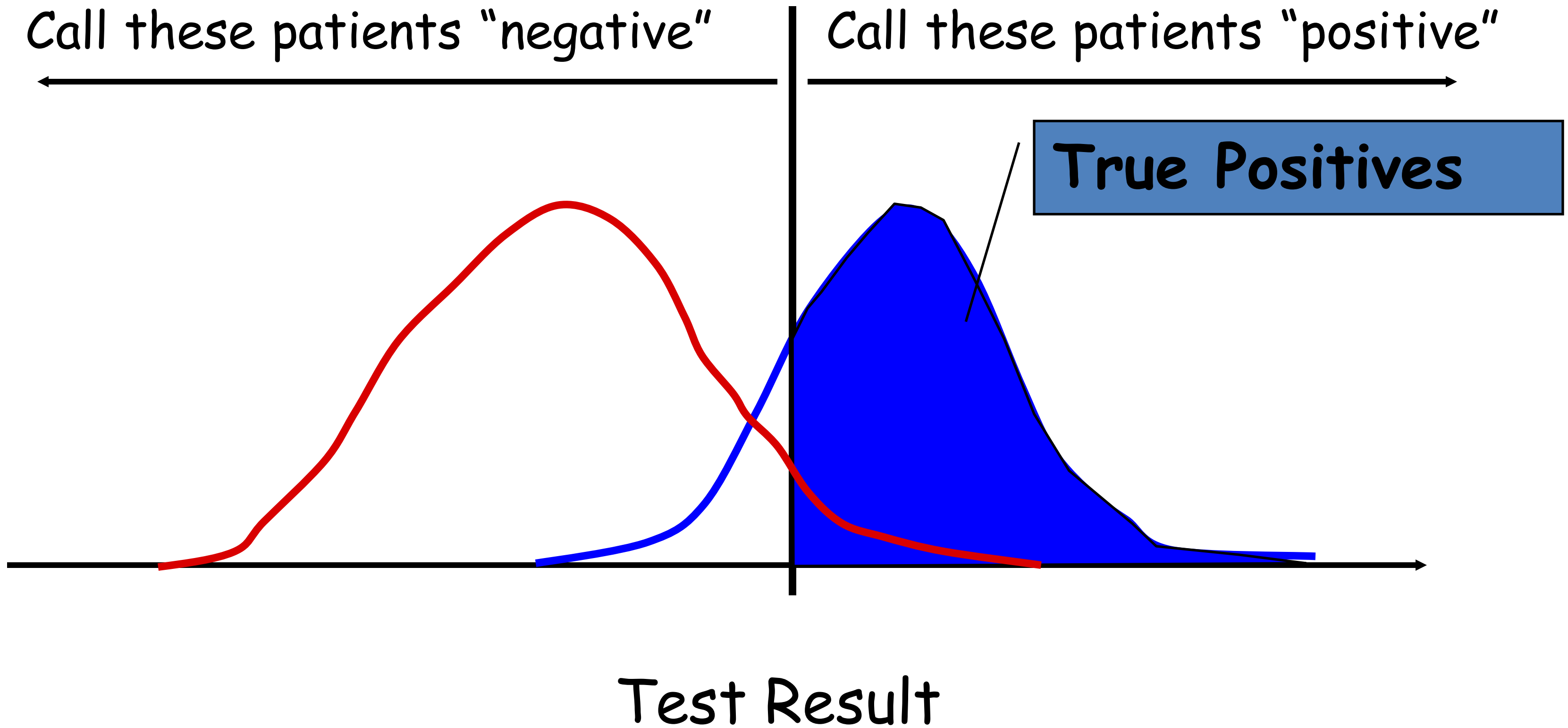
Original slide credit: Darlene Goldstein

Threshold



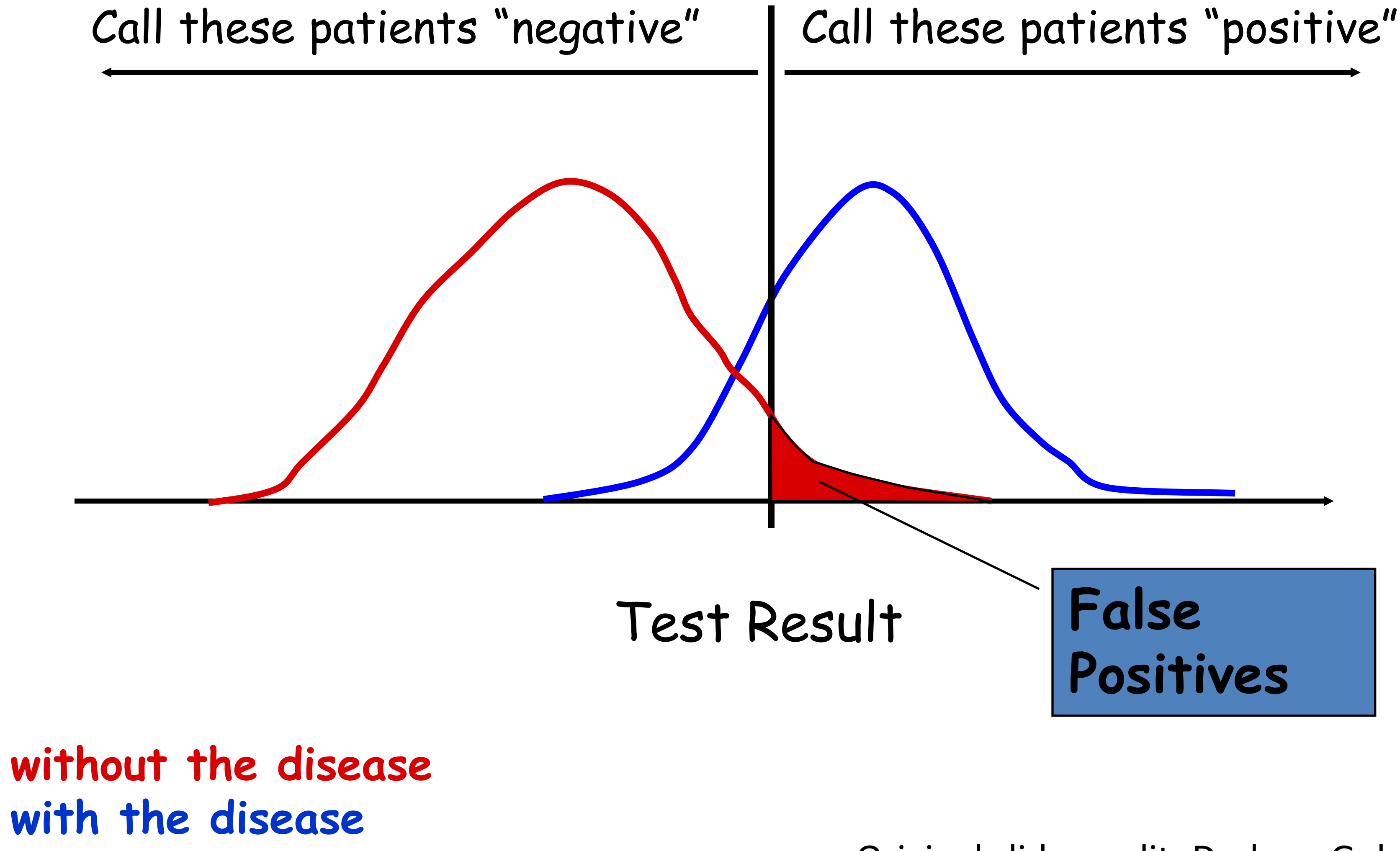
Original slide credit: Darlene Goldstein

Some definitions ...

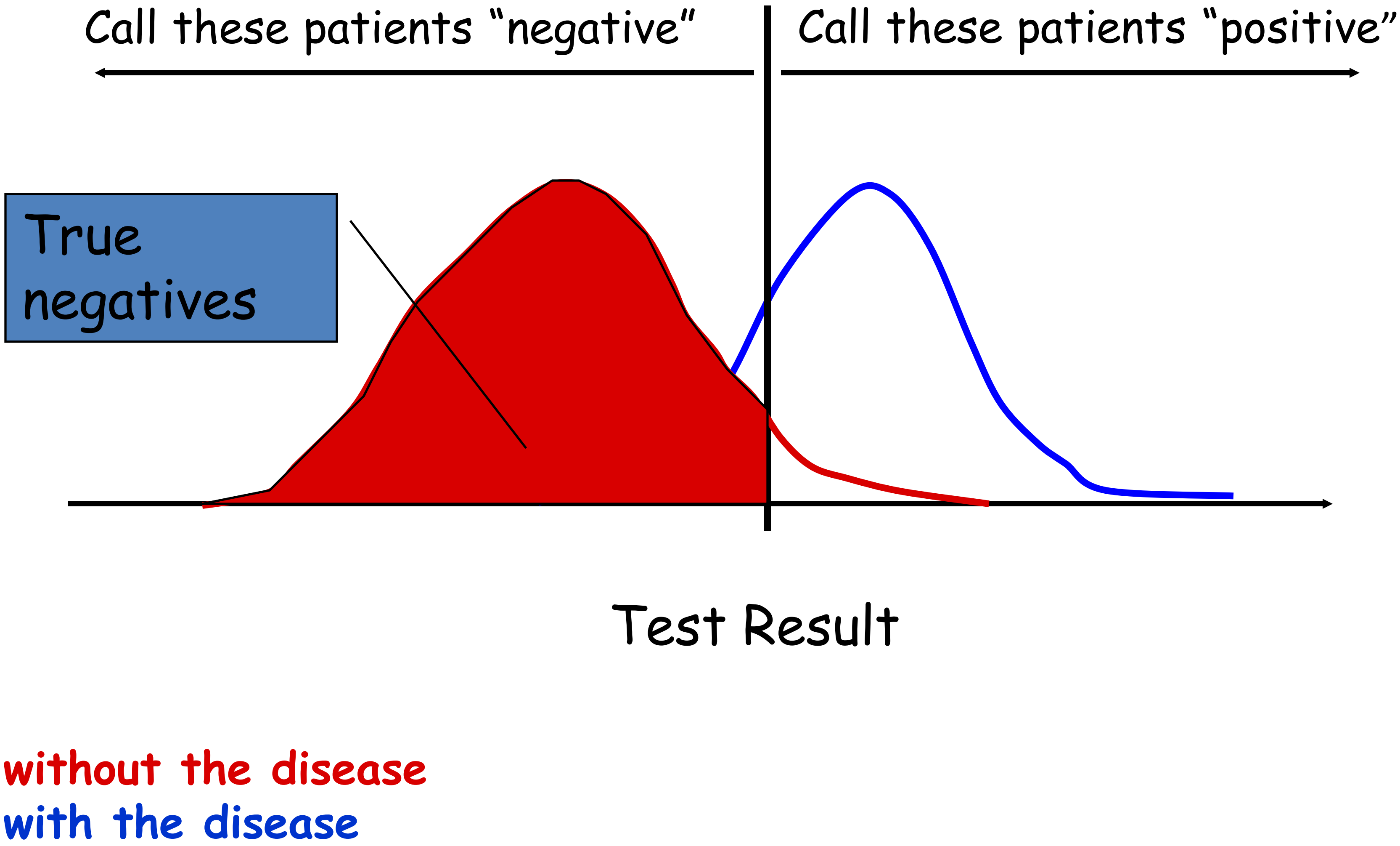


without the disease
with the disease

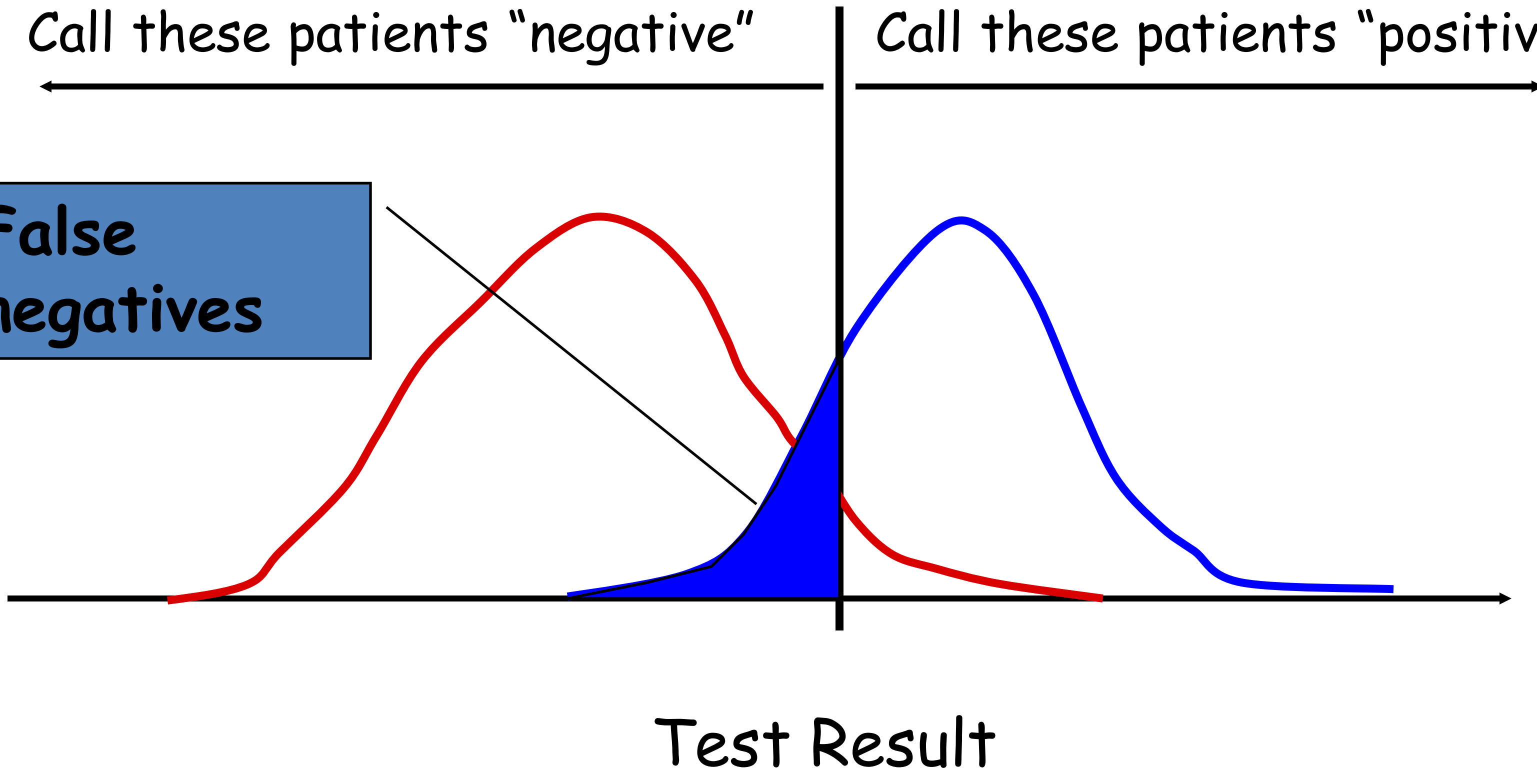
Original slide credit: Darlene Goldstein



Original slide credit: Darlene Goldstein



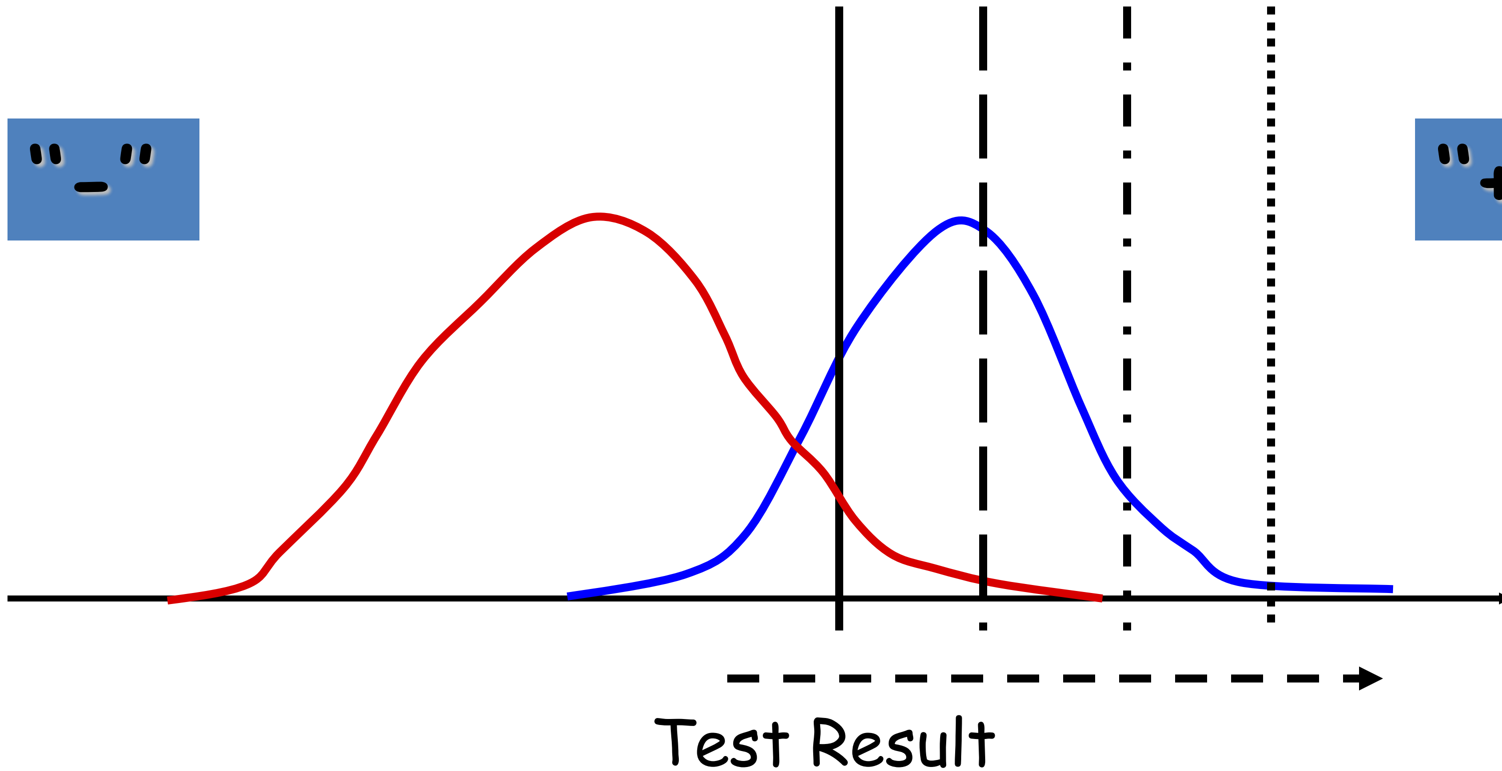
Original slide credit: Darlene Goldstein



without the disease
with the disease

Original slide credit: Darlene Goldstein

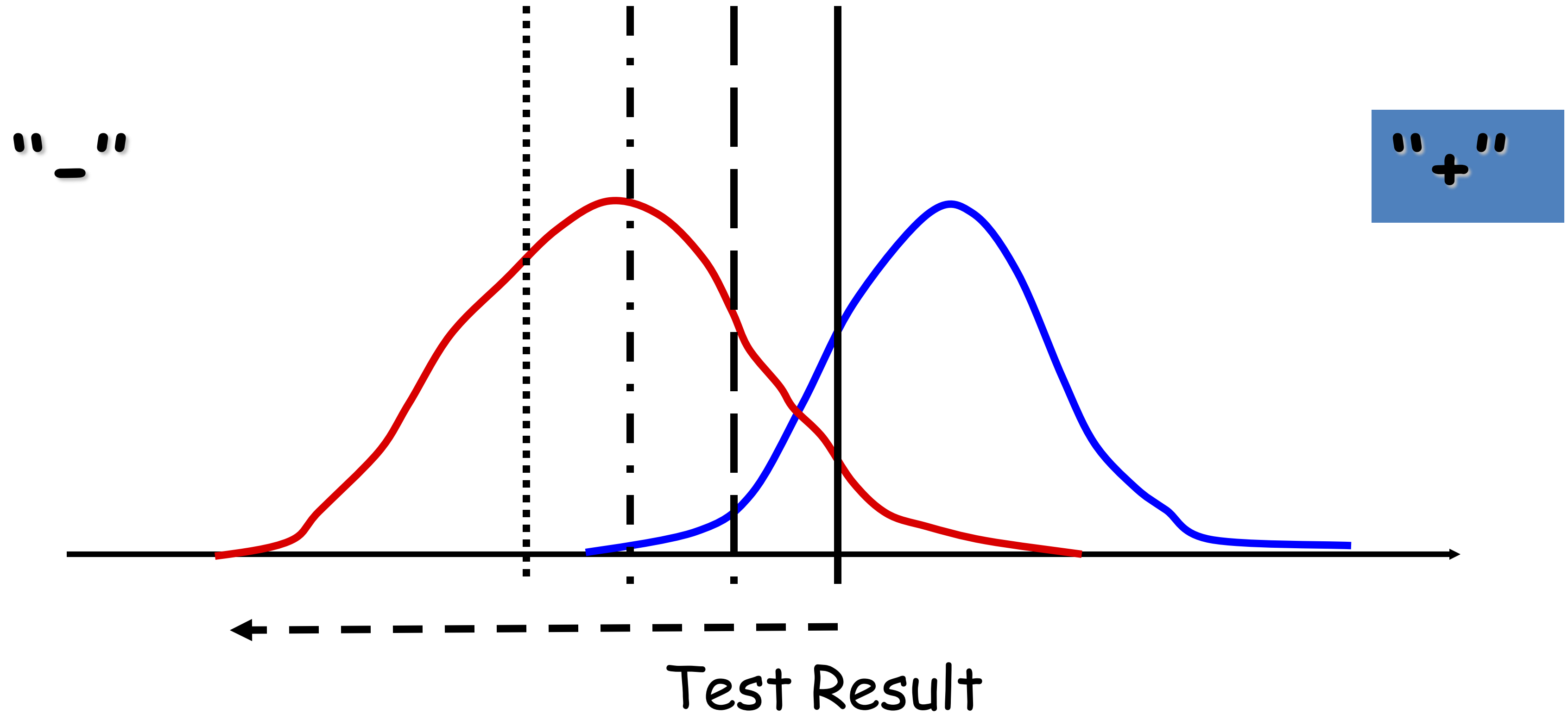
Moving the Threshold: right



without the disease
with the disease

Original slide credit: Darlene Goldstein

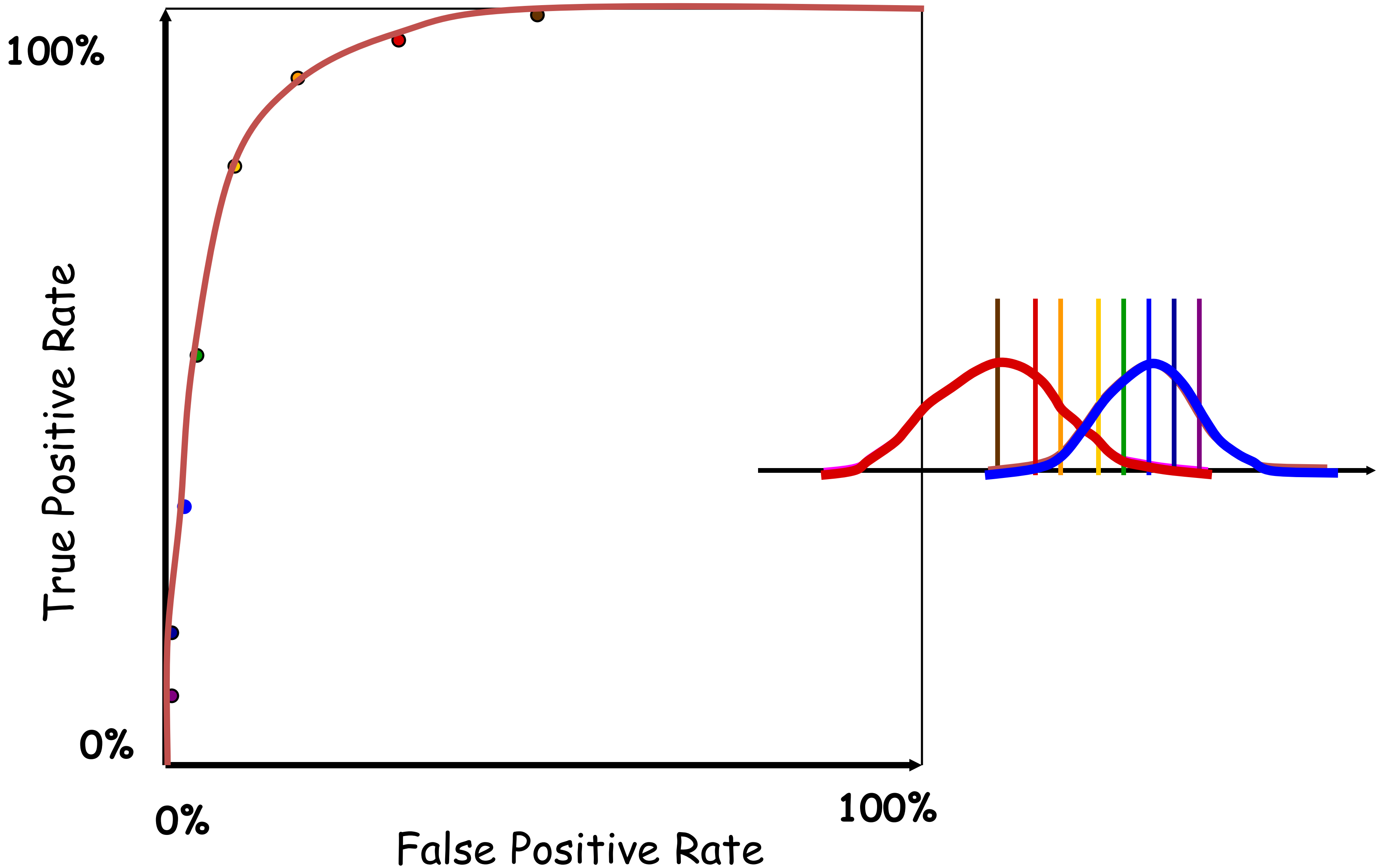
Moving the Threshold: left



without the disease
with the disease

Original slide credit: Darlene Goldstein

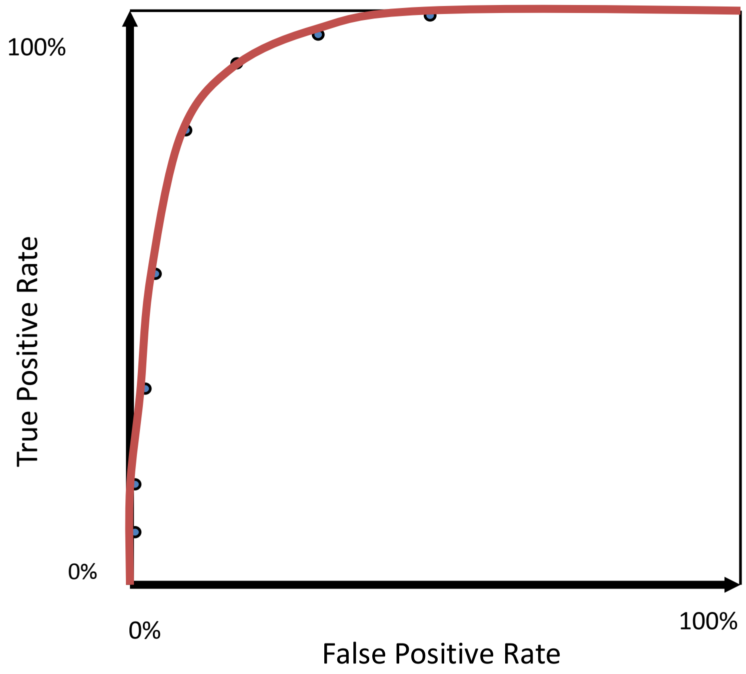
ROC curve



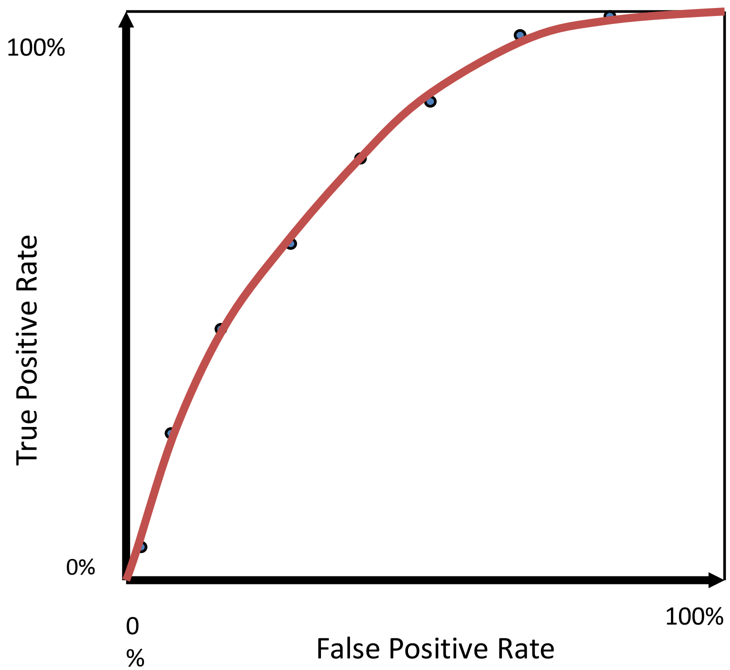
Original slide credit: Darlene Goldstein

ROC curve comparison

A good classifier:

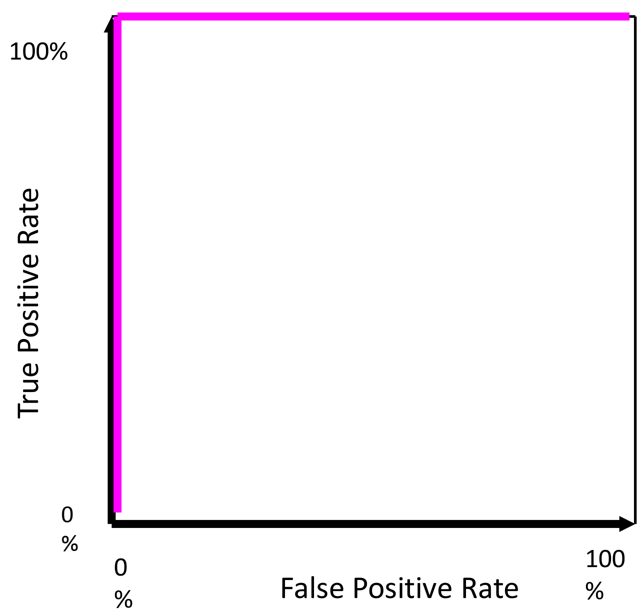


A poor classifier:



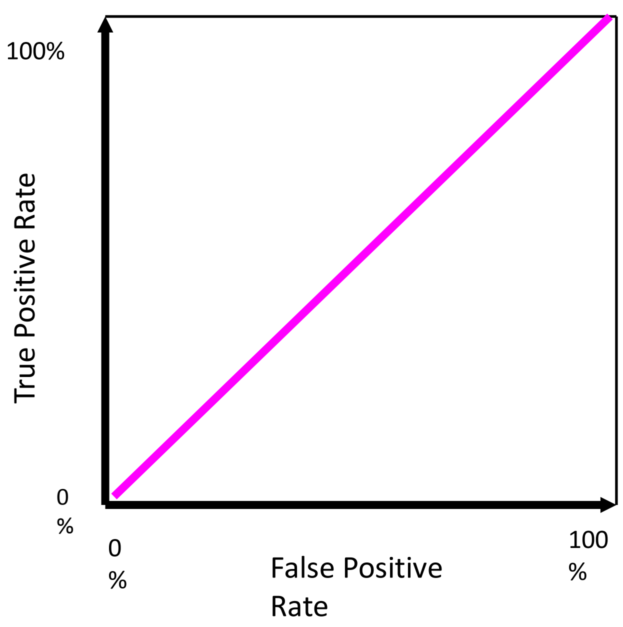
ROC curve extremes

Best Classifier:



The distributions don't overlap at all

Worst Classifier:



The distributions overlap completely