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
  – Constructing kernels
  – SVD and kernel SVD
Number of support vectors

linearly separable data*: \( \#SV \leq d+1 = \text{VC-dim} \)

Minimize \( \frac{1}{2} w^T w \)

subject to \( y_n(w^T x_n + b) \geq 1 \)
\( n = 1, 2, \ldots, N \)

\( w \in \mathbb{R}^d, b \in \mathbb{R} \)

* Could be much more in "degenerate cases"
#SV=2 is sometimes enough
Positive (Semi) Definite (PD/PSD) Matrices

(1) The symmetric $n \times n$ matrix $A$ is positive definite if and only if:

$$Y^T A Y > 0, \quad \forall Y \neq 0$$

(2) $A$ is positive definite $\iff \exists P$ s.t. $A = PP^T$, $|P| \neq 0$

Note: $A = PP^T \Rightarrow A$ is symmetric
Eigenvalues of PD Matrices

- Given the $n \times n$ symmetric matrix $A$, there are $n$ eigenvalues $\lambda$ and vectors $X \neq 0$ where

$$AX = \lambda X$$

$$[X_1 \quad X_2 \quad \cdots \quad X_n]^T A [X_1 \quad X_2 \quad \cdots \quad X_n] = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- If $\lambda_i > 0 \; \forall i \in [1, n] \iff A$ is positive definite

- If $\lambda_i \geq 0 \; \forall i \in [1, n] \iff A$ is positive semi-definite

$A$ is positive definite $\Rightarrow |A| > 0$
Positive (Semi) Definite (PD/PSD) Matrices

The symmetric $n \times n$ matrix $A$ is positive definite if and only if:

1. $Y^T A Y > 0$, $\forall Y \neq 0$

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

3. $\lambda_i > 0 \quad \forall i \in [1, n]$
The dual formulation

\[
\min \frac{1}{2} \alpha^T \begin{bmatrix}
y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \cdots & y_1 y_N x_1^T x_N \\
y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \cdots & y_2 y_N x_2^T x_N \\
\vdots & \vdots & \ddots & \vdots \\
y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \cdots & y_N y_N x_N^T x_N \\
\end{bmatrix} \alpha + (-1^T) \alpha
\]

subject to \( y^T \alpha = 0 \)

linear constraint

\[
0 \leq \alpha \leq C
\]

lower bounds \quad upper bounds
The dual formulation

$$\min_{\alpha} \frac{1}{2} \alpha^T \begin{bmatrix} y_1 y_1 x_1^T x_1 & y_1 y_2 x_1^T x_2 & \cdots & y_1 y_N x_1^T x_N \\ y_2 y_1 x_2^T x_1 & y_2 y_2 x_2^T x_2 & \cdots & y_2 y_N x_2^T x_N \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 x_N^T x_1 & y_N y_2 x_N^T x_2 & \cdots & y_N y_N x_N^T x_N \end{bmatrix} \alpha + (-1^T) \alpha$$

$$M = A^T A$$

Linear
Support Vector Machine

$$f(x) = \sum_i \alpha_i y_i (x_i^T x) + b$$

$$w^T x + b = 0$$

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)

\[
\begin{bmatrix}
    x_1 \\
    x_2 
\end{bmatrix}
\]

nonlinear boundary

\[x_1^2 + x_2^2 = C\]
Nonlinear SVMs

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

\[ \Phi: x \rightarrow \varphi(x) \]
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?

\[
\begin{align*}
\text{Minimize } & \frac{1}{2} w^T w \\
\text{subject to } & y_n (w^T \phi(x_n) + b) \geq 1 \\
& n = 1, 2, ..., N \\
& w \in \mathbb{R}^q, b \in \mathbb{R}
\end{align*}
\]
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?

\[
\min_{\alpha} \frac{1}{2} \alpha^T \begin{bmatrix}
y_1 y_1 \phi(x_1)^T \phi(x_1) & y_1 y_2 \phi(x_1)^T \phi(x_2) & \cdots & y_1 y_N \phi(x_1)^T \phi(x_N) \\
y_2 y_1 \phi(x_2)^T \phi(x_1) & y_2 y_2 \phi(x_2)^T \phi(x_2) & \cdots & y_2 y_N \phi(x_2)^T \phi(x_N) \\
\vdots & \vdots & \ddots & \vdots \\
y_N y_1 \phi(x_N)^T \phi(x_1) & y_N y_2 \phi(x_N)^T \phi(x_2) & \cdots & y_N y_N \phi(x_N)^T \phi(x_N) 
\end{bmatrix} \alpha + \begin{bmatrix} -1^T \end{bmatrix} \alpha
\]
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?

\[ f(x) = \sum_i \alpha_i y_i (\phi(x_i)^T \phi(x)) + b \quad f(x) = w^T \phi(x) + b \]

Dual decision rule \hspace{1cm} Primal decision rule
Where is the $\phi$ “feature” space?

Dual optimization:

$$\min_{\alpha} \frac{1}{2} \alpha^T \begin{bmatrix} y_1 y_1 \phi(x_1)^T \phi(x_1) & y_1 y_2 \phi(x_1)^T \phi(x_2) & \cdots & y_1 y_N \phi(x_1)^T \phi(x_N) \\ y_2 y_1 \phi(x_2)^T \phi(x_1) & y_2 y_2 \phi(x_2)^T \phi(x_2) & \cdots & y_2 y_N \phi(x_2)^T \phi(x_N) \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 \phi(x_N)^T \phi(x_1) & y_N y_2 \phi(x_N)^T \phi(x_2) & \cdots & y_N y_N \phi(x_N)^T \phi(x_N) \end{bmatrix} \alpha + (-1^T) \alpha$$

subject to $y^T \alpha = 0$, $0 \leq \alpha \leq C$

$w$?

$$w = \sum_{i \in SV} \alpha_i y_i \phi(x_i)$$

$f(x) = \sum_i \alpha_i y_i (\phi(x_i)^T \phi(x)) + b$

$b$?

for support vectors we have:

$$f(x) = \sum_i \alpha_i y_i (\phi(x_i)^T \phi(x)) + b = y$$
The “Kernel trick”

**Linear SVM**

\[f(x) = \sum_i \alpha_i y_i (x_i^T x) + b\]

**Non-linear SVM**

\[f(x) = \sum_i \alpha_i y_i (\phi(x_i)^T \phi(x)) + b\]

Define the “kernel function” \(K\)

\[K(x', x'') = \phi(x')^T \phi(x'')\]

then

\[f(x) = \sum_i \alpha_i y_i K(x_i, x) + b\]
Where is the $\phi$ “feature” space?

Dual optimization:

$$\min_{\alpha} \frac{1}{2} \alpha^T \begin{bmatrix} y_1 y_1 \phi(x_1)^T \phi(x_1) & y_1 y_2 \phi(x_1)^T \phi(x_2) & \ldots & y_1 y_N \phi(x_1)^T \phi(x_N) \\ y_2 y_1 \phi(x_2)^T \phi(x_1) & y_2 y_2 \phi(x_2)^T \phi(x_2) & \ldots & y_2 y_N \phi(x_2)^T \phi(x_N) \\ \vdots & \vdots & \ddots & \vdots \\ y_N y_1 \phi(x_N)^T \phi(x_1) & y_N y_2 \phi(x_N)^T \phi(x_2) & \ldots & y_N y_N \phi(x_N)^T \phi(x_N) \end{bmatrix} \alpha + (-1^T) \alpha$$

subject to

$$y^T \alpha = 0 \quad 0 \leq \alpha \leq C$$

$$w = \sum_{i \in SV} \alpha_i y_i \phi(x_i) \quad f(x) = \sum_i \alpha_i y_i (\phi(x_i)^T \phi(x)) + b$$

$$f(x) = \sum_i \alpha_i y_i (\phi(x_i)^T \phi(x)) + b = y$$
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: 2\(^{nd}\) order polynomial kernel in 2d

\[
K(x, x') = (1 + x^t x')^2 = (1 + x_1 x'_1 + x_2 x'_2)^2 = \\
= 1 + x_1^2 x'_1^2 + x_2^2 x'_2^2 + 2x_1 x'_1 + 2x_2 x'_2 + 2x_1 x'_1 x_2 x'_2
\]

\[
K(x, x') = \begin{bmatrix}
1 \\
x_1^2 \\
x_2^2 \\
\sqrt{2}x_1 \\
\sqrt{2}x_2 \\
\sqrt{2}x_1 x_2
\end{bmatrix}^T \begin{bmatrix}
1 \\
x'_1^2 \\
x'_2^2 \\
\sqrt{2}x'_1 \\
\sqrt{2}x'_2 \\
\sqrt{2}x'_1 x'_2
\end{bmatrix} = \phi(x)^T \phi(x)
\]
**Popular kernels**

\[ K(x', x'') = \phi(x')^T \phi(x'') \]

<table>
<thead>
<tr>
<th>Name</th>
<th>Params</th>
<th>Kernel equation ( K(x', y') )</th>
<th>Non-linear mapping ( \phi(x) )</th>
<th>( x )</th>
<th>((1+ (x'))^D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td></td>
<td>[ (1 + (x')^D) ]</td>
<td></td>
<td>( x )</td>
<td>((1+ (x'))^D)</td>
</tr>
<tr>
<td>Polynomial</td>
<td></td>
<td>[ \exp(-|x - x'|^2/(2\sigma^2)) ]</td>
<td>( \text{Infinite dimensional vector} )</td>
<td>( x )</td>
<td>((1+ (x'))^D)</td>
</tr>
<tr>
<td>Gaussian == RB</td>
<td>( \sigma )</td>
<td>[ \exp(-|x - x'|^2/(2\sigma^2)) ]</td>
<td>( \text{Infinite dimensional vector} )</td>
<td>( x )</td>
<td>((1+ (x'))^D)</td>
</tr>
</tbody>
</table>

- **Linear**
  - \( x \)
  - \((1+ (x'))^D\)
- **Polynomial**
  - \( x \)
  - \((1+ (x'))^D\)
- **Gaussian == RB**
  - \( x \)
  - \((1+ (x'))^D\)
## Popular kernels

\[ K(x', x'') = \phi(x')^T \phi(x'') \]

<table>
<thead>
<tr>
<th>K(x',x'')</th>
<th>params</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>(x')</td>
<td>Linear</td>
</tr>
<tr>
<td>Polynomial</td>
<td>(1+(x'))</td>
<td>D</td>
</tr>
<tr>
<td>Gaussian==RB</td>
<td>( \alpha )</td>
<td>Infinite dimensional vector</td>
</tr>
</tbody>
</table>

### Polynomial

All polynomials up to degree D in the elements of the vector x

Infinite dimensional vector

\[
K(x, y) = \left( \sum_{i=1}^{n} x_iy_i + 1 \right)^2 = \sum_{i=1}^{n} x_i^2y_i^2 + \sum_{i=2}^{n} \sum_{j=1}^{i-1} \sqrt{2}x_iy_i\sqrt{2}x_jy_j + \sum_{i=1}^{n} \sqrt{2}x_i\sqrt{2}y_i + 1
\]

\[
\varphi(x) = \langle x_n^2, \ldots, x_1^2, \sqrt{2}x_nx_{n-1}, \ldots, \sqrt{2}x_n, \sqrt{2}x_{n-1}x_{n-2}, \ldots, \sqrt{2}x_{n-1}x_1, \ldots, \sqrt{2}x_2x_1, \sqrt{2}x_n, \ldots, \sqrt{1}x_1, 1 \rangle
\]

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

\[ K(x', x'') = \phi(x')^T \phi(x'') \]

<table>
<thead>
<tr>
<th>Name</th>
<th>params</th>
<th>Kernel equation ( K(x, x') )</th>
<th>Non-linear mapping ( \phi(x) )</th>
<th>K(x', x'') Kernel</th>
<th>params</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td></td>
<td>( \phi(x') )</td>
<td></td>
<td>( (x') )</td>
<td></td>
<td>Linear</td>
</tr>
<tr>
<td>Polynomial</td>
<td>D</td>
<td>( 1 + (x')^D )</td>
<td>All polynomials up to degree D in the elements of the vector x</td>
<td>( (1 + (x'))^D )</td>
<td>D</td>
<td>Polynomial</td>
</tr>
<tr>
<td>Gaussian=RB</td>
<td>( \sigma )</td>
<td>( \exp(-|x-x'|^2/(2\sigma^2)) )</td>
<td>Infinite dimensional vector</td>
<td>( \exp(-|x-x'|^2/(2\sigma^2)) )</td>
<td>( \sigma )</td>
<td>Gaussian=RB</td>
</tr>
</tbody>
</table>

\[ \bar{K}_{ij} = K(x_i, x_j) \]

\[ \bar{K} = [\phi(x_1) \phi(x_2) \ldots \phi(x_N)]^T [\phi(x_1) \phi(x_2) \ldots \phi(x_N)] \]

\[ \text{rank}(\bar{K}) = \text{rank}([\phi(x_1) \phi(x_2) \ldots \phi(x_N)]) < \min(d', N) \]

\[ \text{take } \bar{K} = \begin{bmatrix} 1 & \epsilon & \epsilon \\ \epsilon & 1 & \epsilon \\ \epsilon & \epsilon & \ddots \end{bmatrix} \Rightarrow \text{rank}(\bar{K}) = N \]
\[ K(x', x'') = \exp(-\gamma \| x' - x'' \|^2) \]

\[ \gamma = \frac{1}{2\sigma^2} \]
$K(x', x'') = \exp(-\gamma \| x' - x'' \|^2)$
Proper kernels

• Symmetric

\[ K(x_i, x_j) = K(x_j, x_i) \]

• Positive semi definite kernel

\[ \forall N, \forall x_1, ..., x_N, \forall c \in \mathbb{R}^N, \quad c^T \overline{K} c \geq 0 \]

• But in practice, we don’t necessarily need PSD kernels..
Constructing proper kernels

- $K_3(x', x'') = K_1(x', x'') + K_2(x', x'')$
  
  $\phi_3(x) = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \end{bmatrix}$

- $K_3(x', x'') = K_1(x', x'') \ast K_2(x', x'')$

$$
(\phi_1(x')^t \phi_1(x''))(\phi_2(x'')^t \phi_2(x')) = \text{tr}(\phi_1(x')^t \phi_1(x'') \phi_2(x'')^t \phi_2(x')) = \\
\text{tr}((\phi_2(x') \phi_1(x')^t)(\phi_1(x'') \phi_2(x'')^t)) = \langle \text{VEC}(\phi_2(x') \phi_1(x')^t), \text{VEC}(\phi_2(x'') \phi_1(x'')^t) \rangle
$$
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: \[ D^2(h_1, h_2) = \sum_{i=1}^{N} (h_1(i) - h_2(i))^2 \]
- L-inf distance: \[ D(h_1, h_2) = \max_{1 \leq i \leq N} |h_1(i) - h_2(i)| \]
- \(\chi^2\) distance: \[ D(h_1, h_2) = \sum_{i=1}^{N} \frac{(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)} \]
- 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: \[ D^2(h_1, h_2) = (h_1 - h_2)^T S^{-1} (h_1 - h_2) \]
The Intersection Kernel

Histogram Intersection kernel between histograms $a, b$

$$K(a, b) = \sum_{i=1}^{n} \min(a_i, b_i)$$

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

Demonstration of Positive Definiteness

Histogram Intersection kernel between histograms $a, b$

$$K(a, b) = \sum_{i=1}^{n} \min(a_i, b_i)$$

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:

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

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

Decision function is

\[
h(x) = \sum_{j=1}^{\text{#sv}} \alpha^j \left( \sum_{i=1}^{\text{#dim}} \min(x_i, x_i^j) \right) + b
\]

= \sum_{i=1}^{\text{#dim}} \left( \sum_{j=1}^{\text{#sv}} \alpha^j \min(x_i, x_i^j) \right) + b

= \sum_{i=1}^{\text{#dim}} h_i(x_i) + b

\[
h_i(x_i) = \sum_{j=1}^{\text{#sv}} \alpha^j \min(x_i, x_i^j) + b
\]

= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left( \sum_{x_i^j \geq x_i} \alpha^j \right) x_i

where:

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 (cost: \( \log \text{#sv} \)) look up values, multiply \& add
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
  \[ A = U W V^T \]
  - $U$ is $m \times n$ and orthonormal
  - $W$ is $n \times n$ and diagonal
  - $V$ is $n \times n$ and orthonormal
SVD

Matlab: \([U, W, V] = \text{svd}(A, 0)\)

\[
\begin{pmatrix}
A \\
\end{pmatrix} = \begin{pmatrix}
U \\
\end{pmatrix}\begin{pmatrix}
w_1 & 0 & 0 \\
0 & w_2 & 0 \\
0 & 0 & w_n \\
\end{pmatrix}\begin{pmatrix}
V \\
\end{pmatrix}^T
\]

- 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.
- \(\text{rank}(A) = \text{number of nonzero } w_i\)
- SVD is unique (unless some \(w_i\) are equal)
SVD and Inverses

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

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

• Solving \( Ax = b \) by least squares
  \( x = \text{pinv}(A)^* b \)
SVD and Eigenvectors

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

$$A^TAx_i = VW^TU^TUWV^Tx_i = VW^2V^Tx_i = VW^2\begin{pmatrix} 0 \\ x_i \end{pmatrix} = V\begin{pmatrix} 0 \\ w_i^2 \end{pmatrix} = w_i^2x_i$$

- So elements of $W$ are $\sqrt{\text{eigenvalues}}$ and columns of $V$ are eigenvectors of $A^TA$
- Similarly, the columns of $U$ are the eigenvectors of $AA^T$, and $\text{diag}(W)$ are $\sqrt{\text{eigenvalues}}$ of $AA^T$
Kernel SVD

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

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) $A^{-1} = V W^{-1} U^T$
(2) kernel PCA
True label vs. classifier result

<table>
<thead>
<tr>
<th>Real label</th>
<th>classifier</th>
<th>Prediction=-1</th>
<th>Prediction=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>No disease</td>
<td></td>
<td>☺ True negative</td>
<td>X False positive</td>
</tr>
<tr>
<td>(D = -1)</td>
<td></td>
<td><strong>True negative</strong></td>
<td><strong>False positive</strong></td>
</tr>
<tr>
<td>Disease</td>
<td></td>
<td>X Miss</td>
<td>☺ True positive</td>
</tr>
<tr>
<td>(D = +1)</td>
<td></td>
<td></td>
<td><strong>True positive</strong></td>
</tr>
</tbody>
</table>

Original slide credit: Darlene Goldstein
Specific Example

Pts without the disease

Pts with disease

Test Result

\[ f(x) = w^t x + b \]
Threshold

Call these patients “negative”

Call these patients “positive”

Test Result

Original slide credit: Darlene Goldstein
Some definitions …

Call these patients “negative”

Call these patients “positive”

Test Result

without the disease
with the disease

True Positives

Original slide credit: Darlene Goldstein
Call these patients “negative”

Call these patients “positive”

Test Result

False Positives

without the disease

with the disease

Original slide credit: Darlene Goldstein
Call these patients “negative”

Call these patients “positive”

True negatives

without the disease
with the disease

Test Result

Original slide credit: Darlene Goldstein
Test Result

Call these patients “negative”

Call these patients “positive”

False negatives

without the disease
with the disease

Original slide credit: Darlene Goldstein
Moving the Threshold: right

Test Result

without the disease
with the disease

Original slide credit: Darlene Goldstein
Moving the Threshold: left

without the disease
with the disease

Test Result

Original slide credit: Darlene Goldstein
A good classifier:

A poor classifier:

ROC curve comparison
ROC curve extremes

Best Classifier:

The distributions don’t overlap at all

Worst Classifier:

The distributions overlap completely