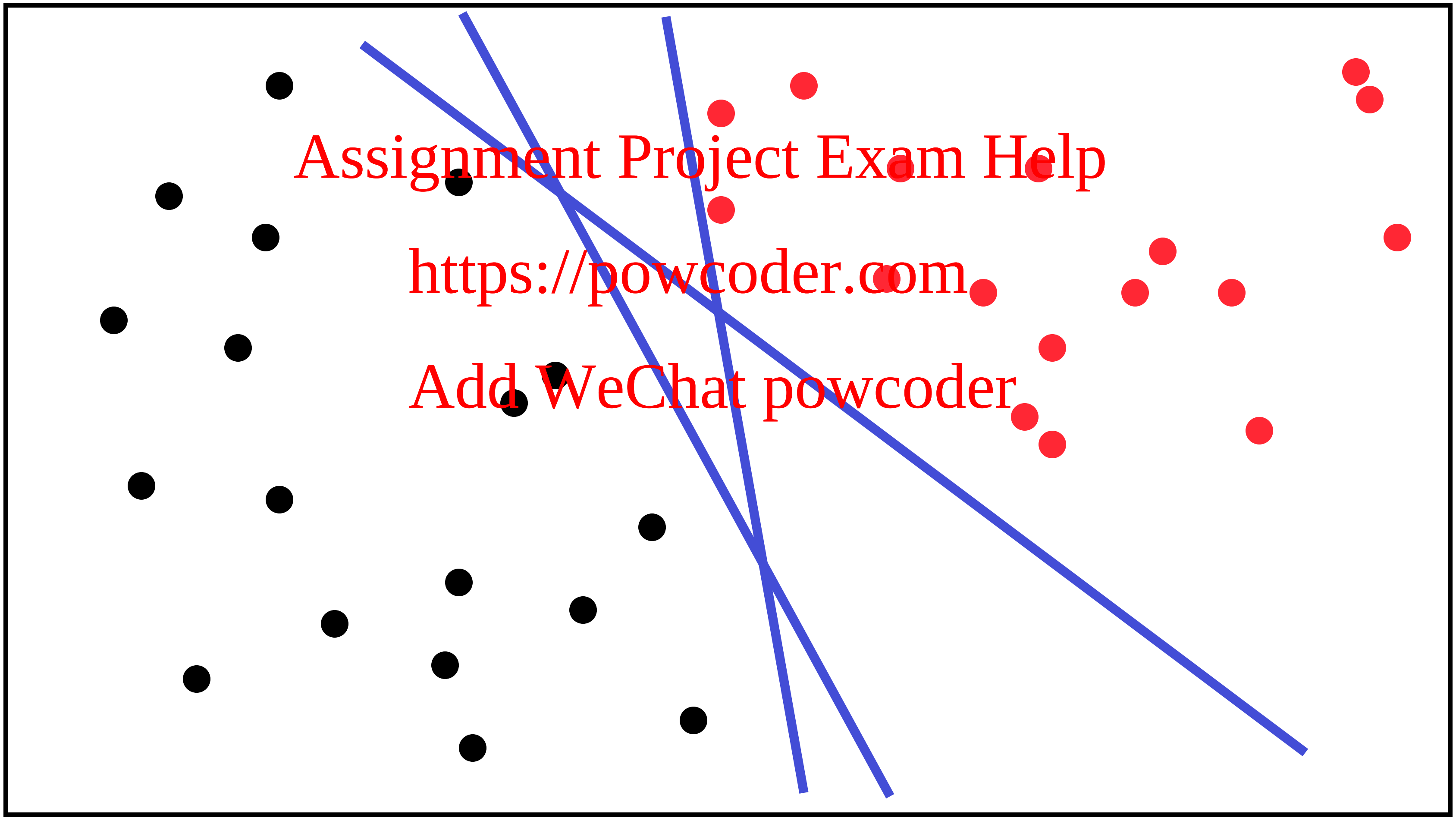
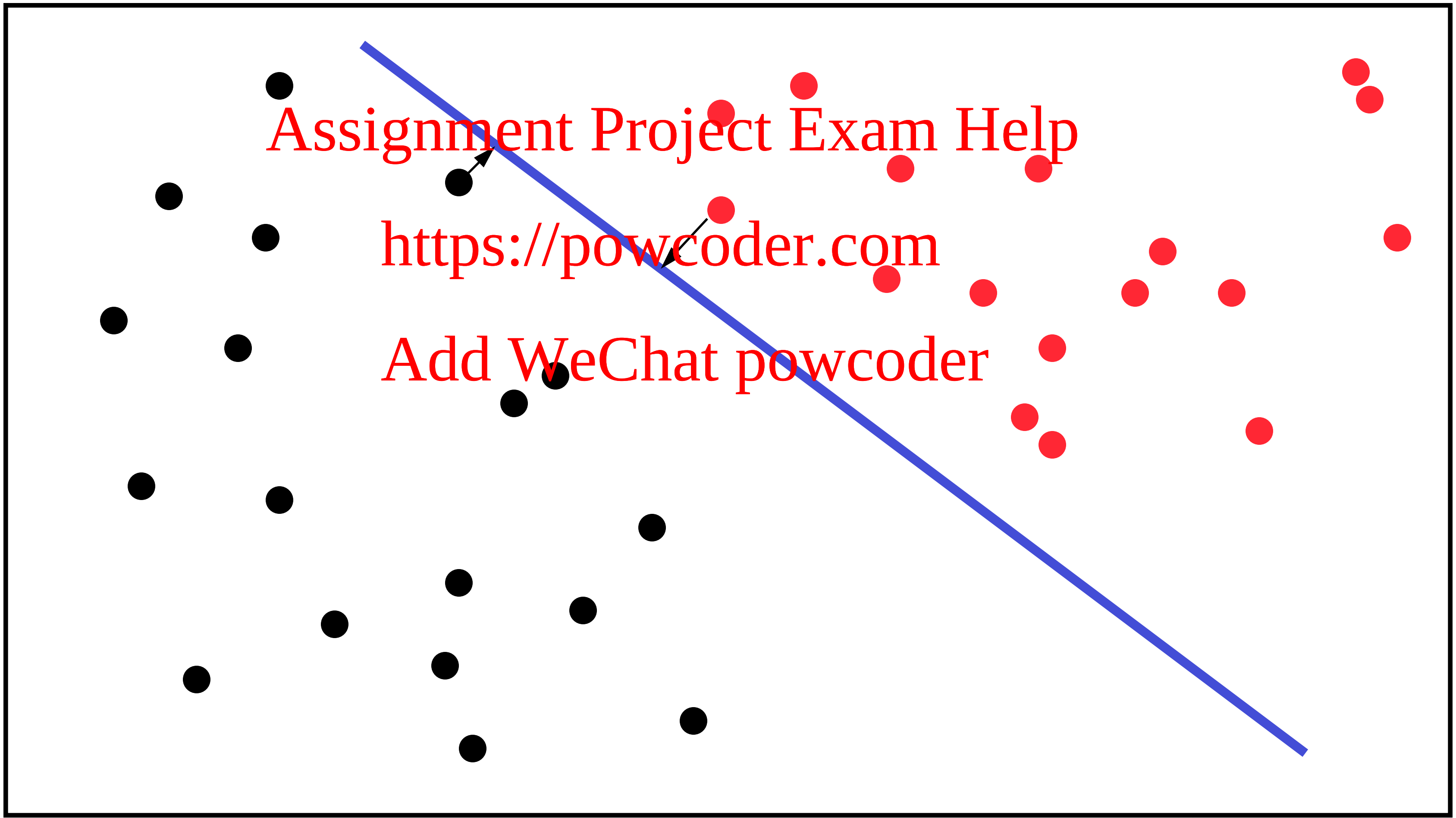
Which Separator?



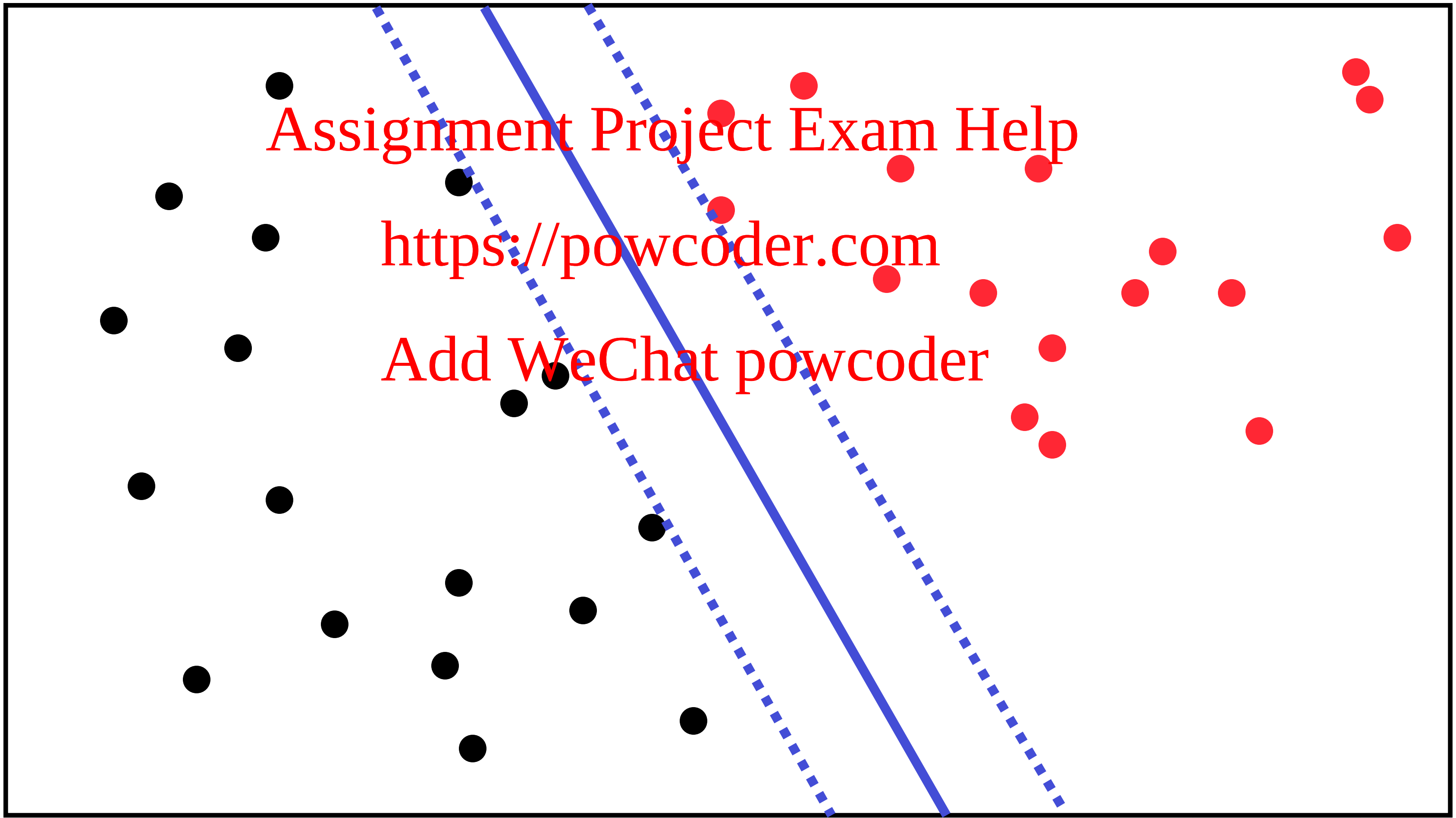
Which Separator?

Maximize the margin to closest points



Which Separator?

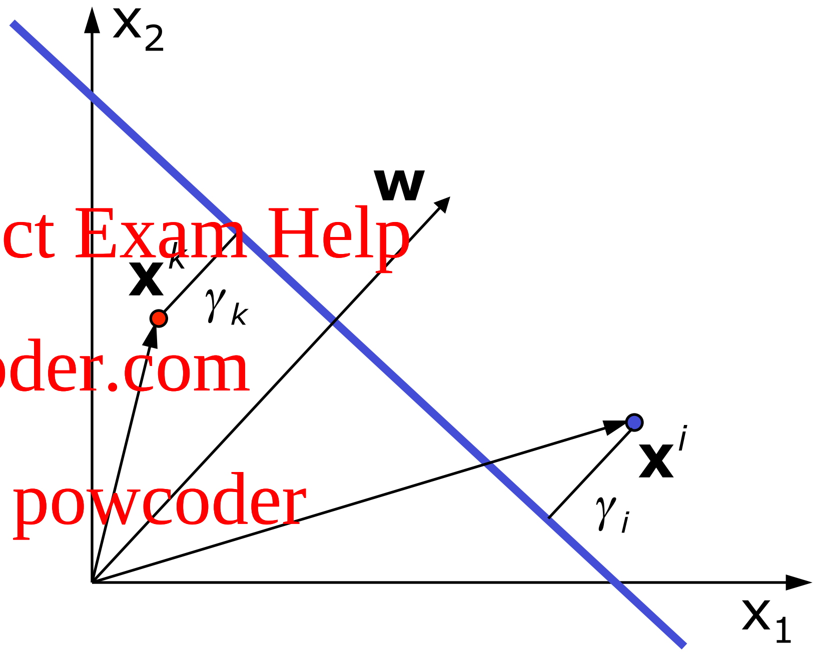
Maximize the margin to closest points



Margin of a point

$$\gamma^i \equiv y^i(\mathbf{w} \cdot \mathbf{x}^i + b)$$

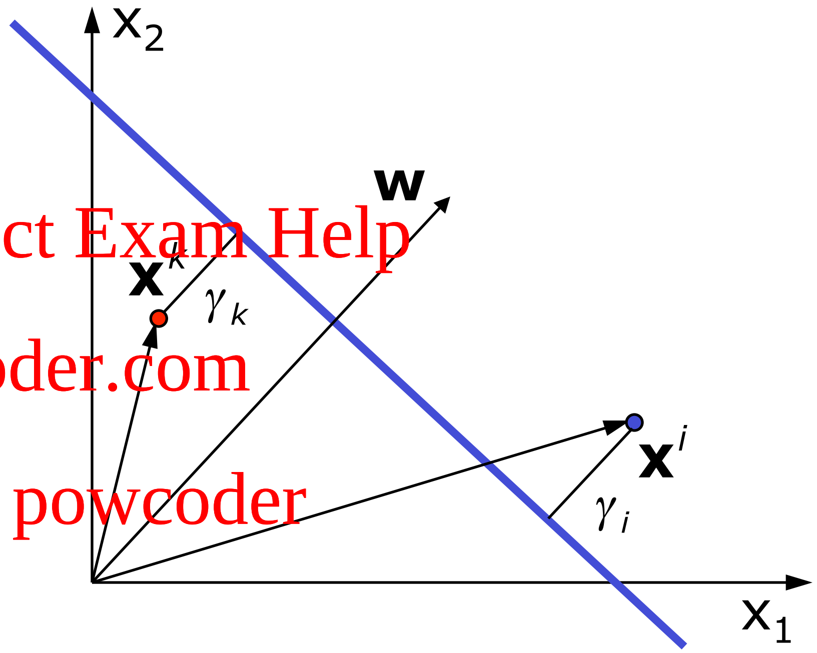
- proportional to perpendicular distance of point $\mathbf{x}^i$ to hyperplane



Margin of a point

$$\gamma^i \equiv y^i(\mathbf{w}\cdot\mathbf{x}^i + b)$$

- proportional to perpendicular distance of point $\mathbf{x}^i$ to hyperplane
- geometric margin is $\gamma^i / \|\mathbf{w}\|$



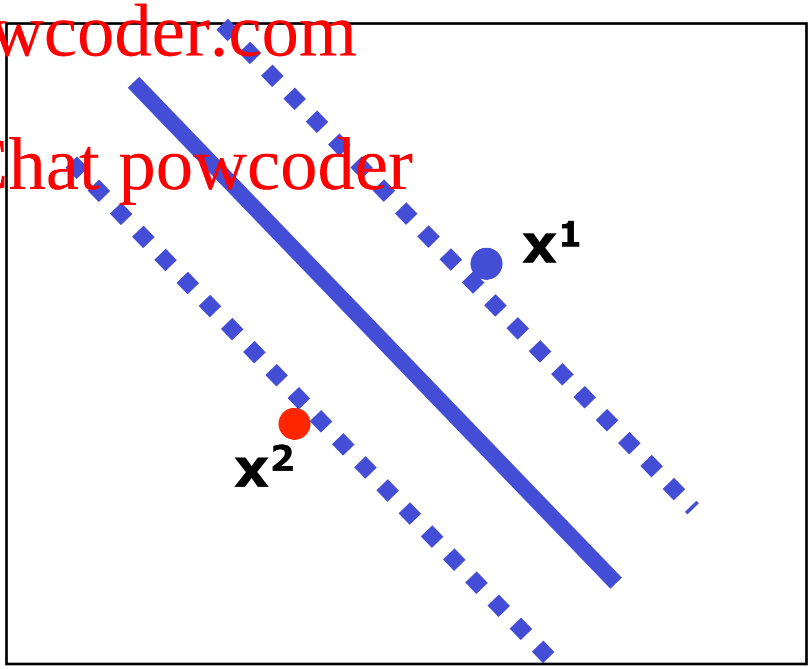
Margin

$$\gamma^i \equiv y^i(\mathbf{w}\cdot\mathbf{x}^i + b)$$

- Scaling **w** changes value of margin but not actual distances to separator (geometric margin)
- Pick the margin to closest positive and negative points to be 1

$$+1(\mathbf{w}\cdot\mathbf{x}^1 + b) = 1$$

$$-1(\mathbf{w}\cdot\mathbf{x}^2 + b) = 1$$



Margin

- Pick the margin to closest positive and negative points to be 1

$$+1(\mathbf{w}\cdot\mathbf{x}^1+b)=1$$

$$-1(\mathbf{w}\cdot\mathbf{x}^2+b)=1$$

- Combining these

$$\mathbf{w}\cdot(\mathbf{x}^1-\mathbf{x}^2)=2$$

- Dividing by length of **w** gives perpendicular distance between lines (2 **x** geometric margin)

$$\frac{\mathbf{w}}{\|\mathbf{w}\|}\cdot(\mathbf{x}^1-\mathbf{x}^2)=\frac{2}{\|\mathbf{w}\|}$$

Picking w to Maximize Margin

- Pick **w** to maximize geometric margin

$$\frac{2}{\|\mathbf{w}\|}$$

- or, equivalently, minimize

$$\|\mathbf{w}\| = \sqrt{\mathbf{w} \cdot \mathbf{w}}$$

- or, equivalently, minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} = \frac{1}{2}\sum_j w_j^2$$

Picking w to Maximize Margin

- Pick **w** to maximize geometric margin

$$\frac{2}{\|\mathbf{w}\|}$$

- or, equivalently, minimize

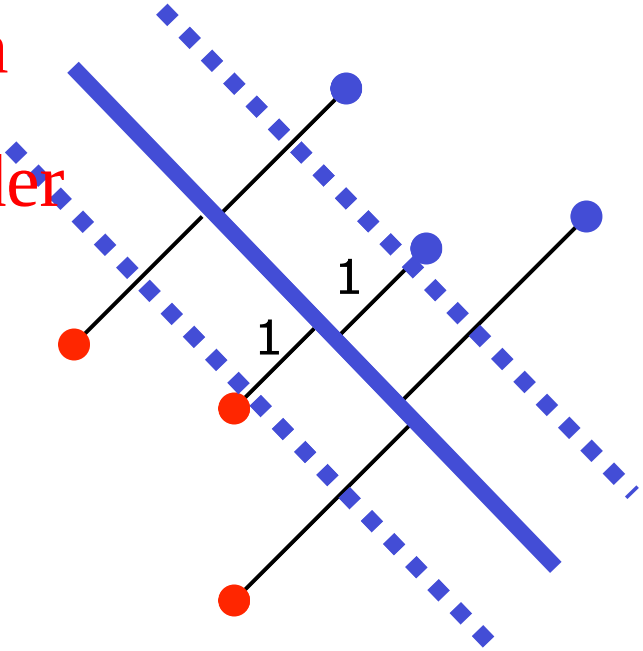
$$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} = \frac{1}{2}\sum_j w_j^2$$

- while classifying points correctly

$$y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1$$

- or, equivalently,

$$y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0$$



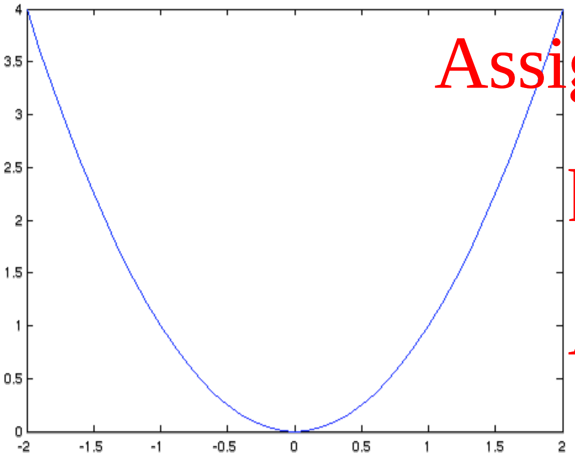
Constrained Optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0, \ \forall_i$$

Constrained optimization

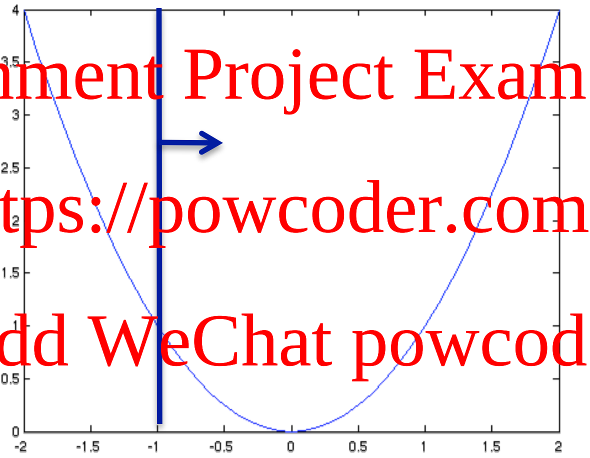
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

No Constraint



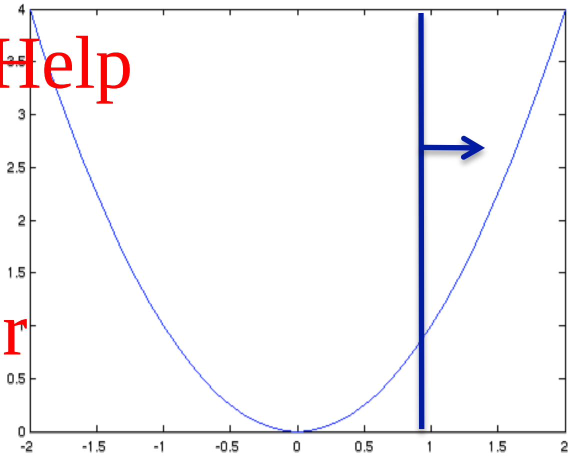
x*=0

x ≥ -1



x*=0

x ≥ 1

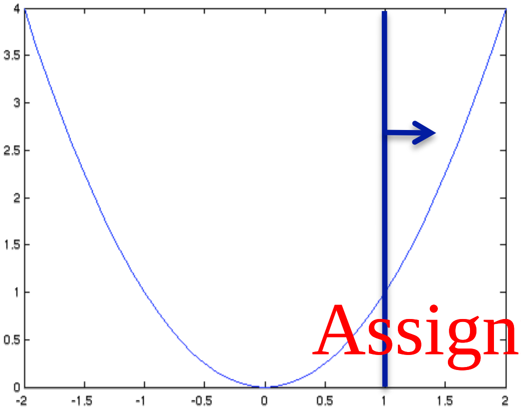


x*=1

How do we solve with constraints?

→ Lagrange Multipliers!!!

Lagrange multipliers – Dual variables



$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

Add Lagrange multiplier

Rewrite Constraint

Introduce Lagrangian (objective):

$$L(x,\alpha) = x^2 - \alpha(x-b)$$

We will solve:

$$\min_x \max_\alpha \; L(x,\alpha)$$
$$\text{s.t.} \quad \alpha \geq 0$$

Add new constraint

Why does this work at all???

- min is fighting max!
- x<b → (x-b)<0 → $\max_\alpha$-α(x-b) = ∞
  - min won't let that happen!!
- x>b, α>0→ (x-b)>0 → $\max_\alpha$-α(x-b) = 0, α*=0
  - min is cool with 0, and L(x, α)=x^2 (original objective)
- x=b → α can be anything, and L(x, α)=x^2 (original objective)
- Since min is on the outside, can force max to behave and constraints will be satisfied!!!

Constrained Optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } \; y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0, \;\; \forall_i$$

Convert to unconstrained optimization by incorporating the constraints as an additional term

$$\min_{\mathbf{w}} \left(\frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \right] \right) \quad \alpha_i \geq 0, \forall_i$$

Constrained Optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } \; y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0, \;\; \forall_i$$

Convert to unconstrained optimization by incorporating the constraints as an additional term

$$\min_{\mathbf{w}} \left(\frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \right] \right) \quad \alpha_i \geq 0, \forall_i$$

To minimize expression:

minimize first (original) term, and

maximize second (constraint) term

since $\alpha_i > 0$, encourages constraints to be satisfied

but we want least “distortion” of original term…

Constrained Optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } \ y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \ge 0, \ \ \forall_i$$

Convert to unconstrained optimization by incorporating the constraints as an additional term

$$\min_{\mathbf{w}} \left(\frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \right] \right) \quad \alpha_i \ge 0, \forall_i$$

Lagrange multipliers

To minimize expression:

minimize first (original) term, and

maximize second (constraint) term

since $\alpha_i > 0$, encourages constraints to be satisfied

but we want least “distortion” of original term…

Method of Lagrange multipliers

Maximizing the Margin

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i (\mathbf{w} \cdot \mathbf{x}^i + b) - 1 \right]$$

Maximizing the Margin

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i (\mathbf{w} \cdot \mathbf{x}^i + b) - 1 \right]$$

Minimized when: $\boxed{\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i}$ $\boxed{\sum_i \alpha_i y^i = 0}$

Maximizing the Margin

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i (\mathbf{w} \cdot \mathbf{x}^i + b) - 1 \right]$$

Minimized when: $\boxed{\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i}$ $\boxed{\sum_i \alpha_i y^i = 0}$

Substituting **w*** into L yields dual Lagrangian:

$$\boxed{L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{k=1}^{m} \alpha_i \alpha_k y_i y_k \mathbf{x}_i \mathbf{x}_k}$$

Only dot products of the feature vectors appear

Dual Lagrangian

$$\max_{\alpha} L(\alpha) \quad \text{subject to} \quad \sum_{i} \alpha_i y^i = 0 \quad \text{and} \quad \alpha_i \geq 0, \forall i$$

Dual Lagrangian

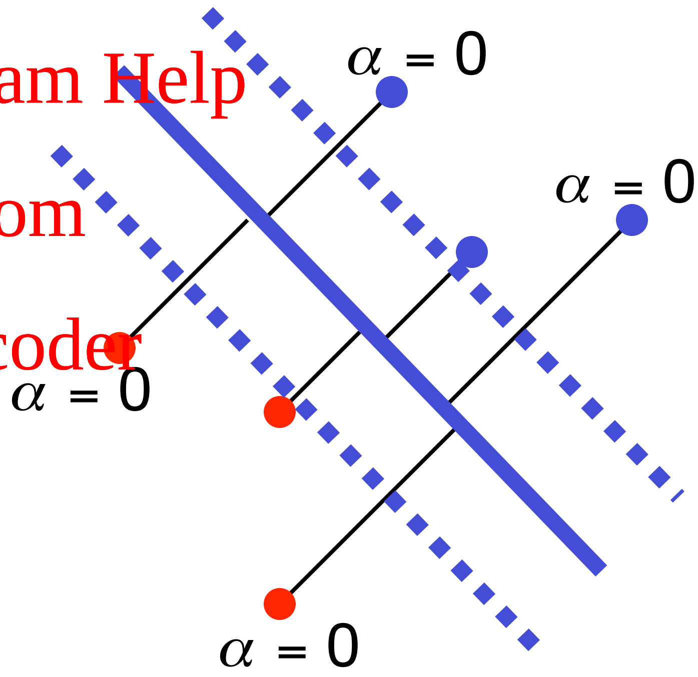
$$\max_{\alpha} L(\alpha) \quad \text{subject to} \quad \sum_{i} \alpha_i y^i = 0 \quad \text{and} \quad \alpha_i \geq 0, \forall i$$

In general, since $\alpha_i >= 0$, either

$\alpha_i = 0$: constraint is satisfied with no distortion at optimum w

or

$\alpha_i > 0$: constraint is satisfied with equality (in this case $\mathbf{x}^i$ is known as a support vector)



Dual Lagrangian

$$\max_{\alpha} L(\alpha) \quad \text{subject to } \sum_{i} \alpha_i y^i = 0 \quad \text{and } \alpha_i \geq 0, \forall i$$

In general, since $\alpha_i >= 0$, either

$\alpha_i = 0$: constraint is satisfied with no distortion at optimum w

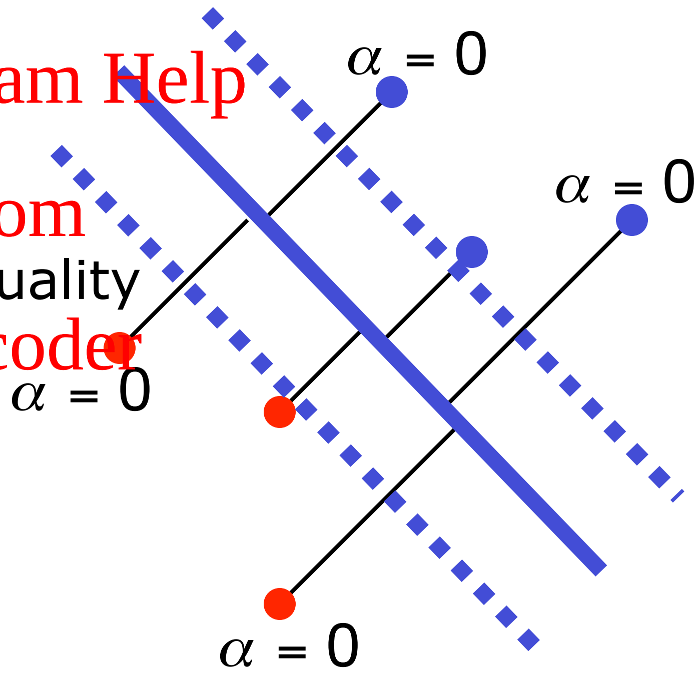
or

$\alpha_i > 0$: constraint is satisfied with equality

($\mathbf{x}^i$ is known as a support vector)

$$\boxed{\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i}$$

$$\boxed{b = 1/y^i - \mathbf{w}^* \mathbf{x}^i}$$



Dual Lagrangian

$$\max_{\alpha} L(\alpha) \quad \text{subject to} \quad \sum_{i} \alpha_i y^i = 0 \quad \text{and} \quad \alpha_i \geq 0, \forall i$$

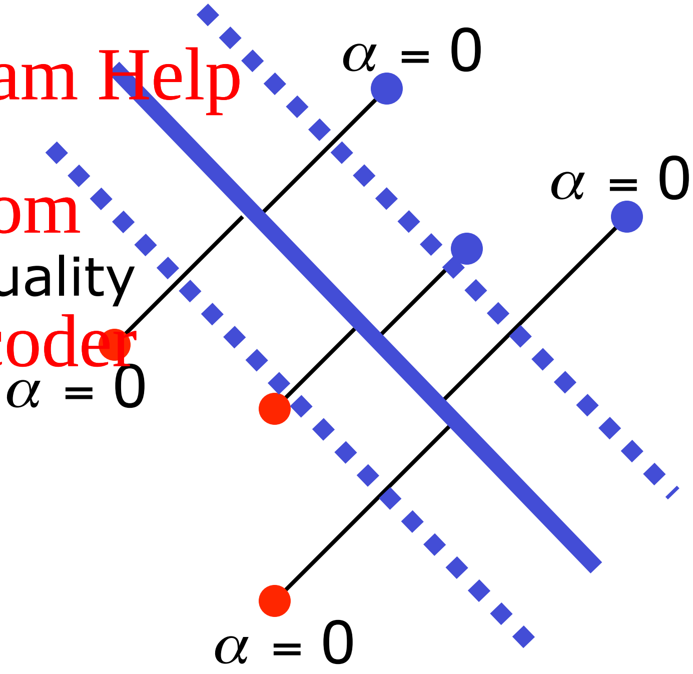
In general, since $\alpha_i >= 0$, either

$\alpha_i = 0$: constraint is satisfied with no distortion at optimum w

or

$\alpha_i > 0$: constraint is satisfied with equality (**x**^{i} is known as a support vector)

$$\boxed{\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i} \qquad \boxed{b = 1/y^i - \mathbf{w}^* \mathbf{x}^i}$$



- Has a unique maximum vector
- Can be found using quadratic programming or gradient ascent

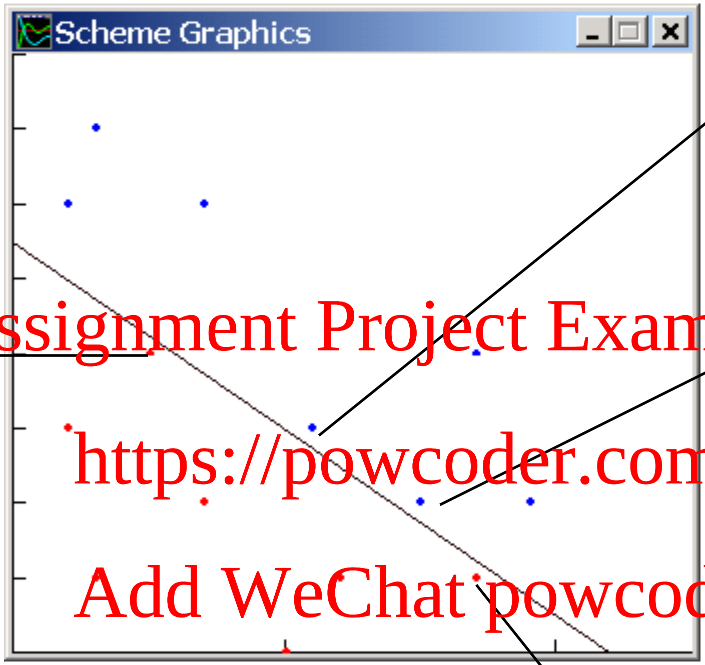
SVM Classifier

- Given unknown vector **u**, predict class (1 or -1) as follows:

$$h(\mathbf{u}) = sign\left(\sum_{i=1}^{k} \alpha_i y^i \mathbf{x}^i \cdot \mathbf{u} + b\right)$$

- The sum is over *k* support vectors

Bankruptcy Example



$\alpha_i y^i$ for support vectors are non-zero, all others are zero.

Key Points

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.

Key Points

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
- Exclusive reliance on dot products enables approach to non-linearly separable problems.

Key Points

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
- Exclusive reliance on dot products enables approach to non-linearly separable problems.
- The classifier depends only on the support vectors, not on all the training points.

Key Points

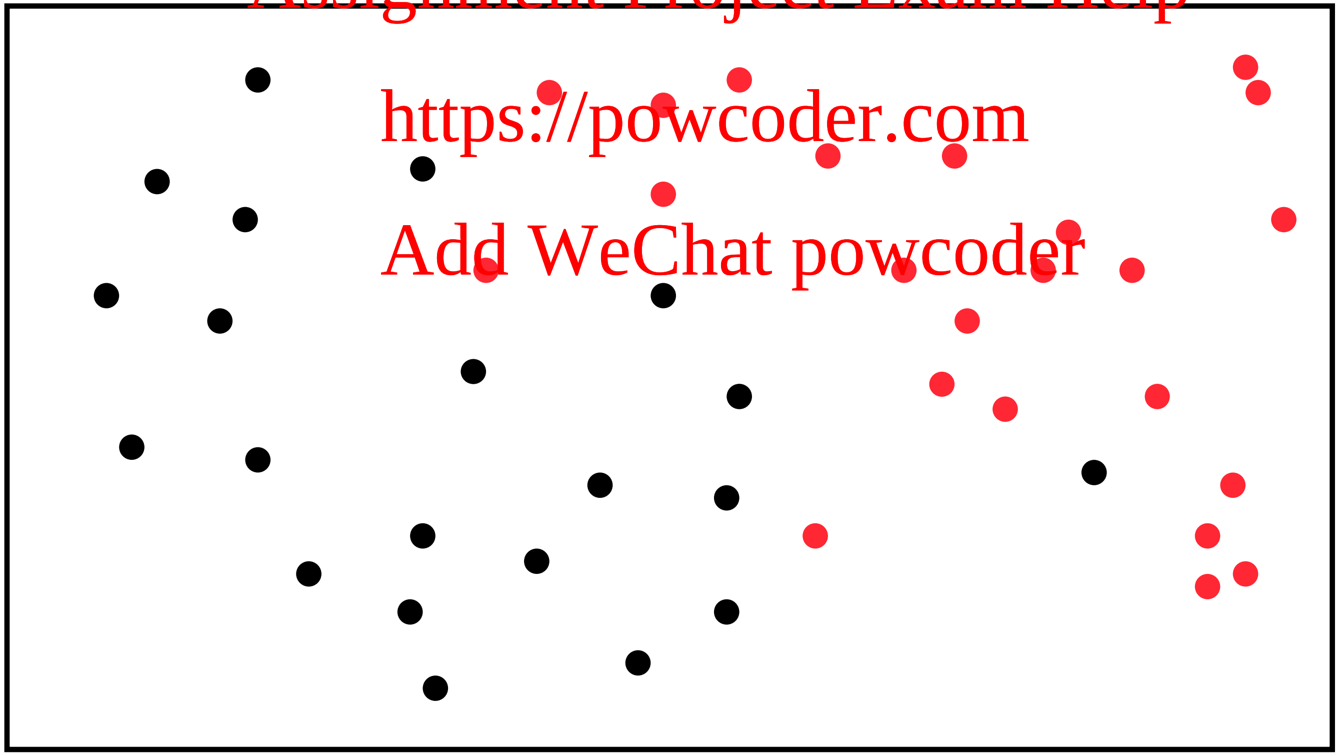
- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
- Exclusive reliance on dot products enables approach to non-linearly separable problems.
- The classifier depends only on the support vectors, not on all the training points.
- Max margin lowers hypothesis variance.

Key Points

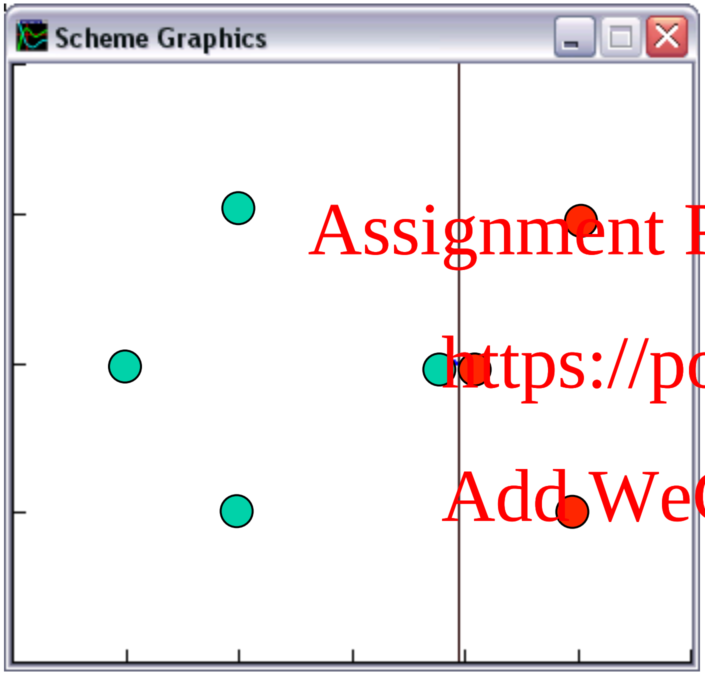
- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
- Exclusive reliance on dot products enables approach to non-linearly separable problems.
- The classifier depends only on the support vectors, not on all the training points.
- Max margin lowers hypothesis variance.
- The optimal classifier is defined uniquely – there are no "local maxima" in the search space
- Polynomial in number of data points and dimensionality

Not Linearly Separable?

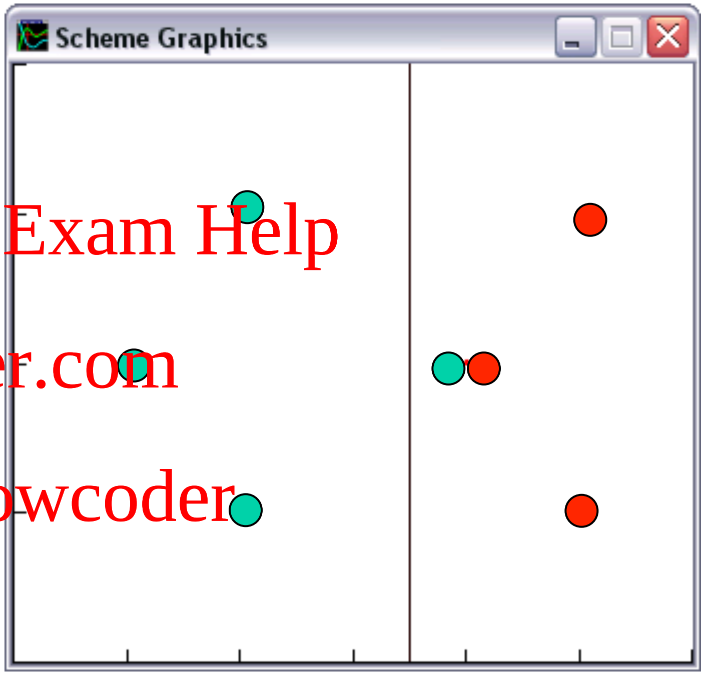
- Require $0 \leq \alpha_i \leq C$
- C specified by user; controls tradeoff between size of margin and classification errors
- C = **1** for separable case



C Change

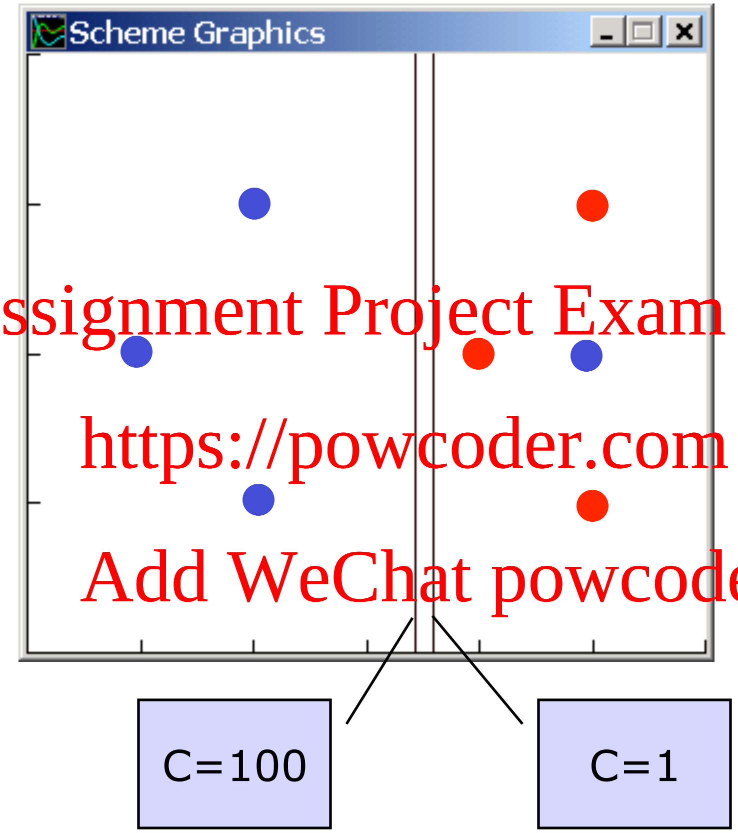


C=10



C=1

C Change



Example: Linearly Separable

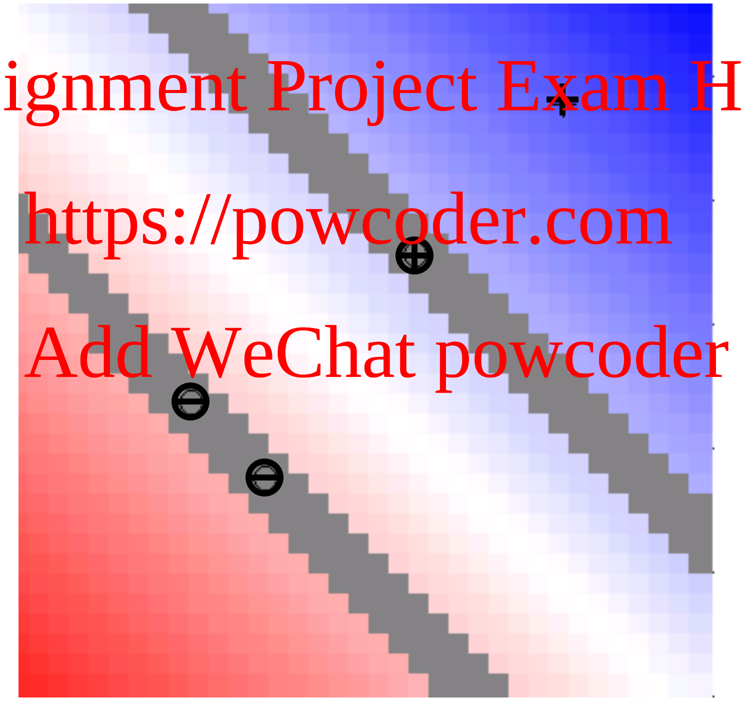


Image by Patrick Winston

Another example: Not linearly separable

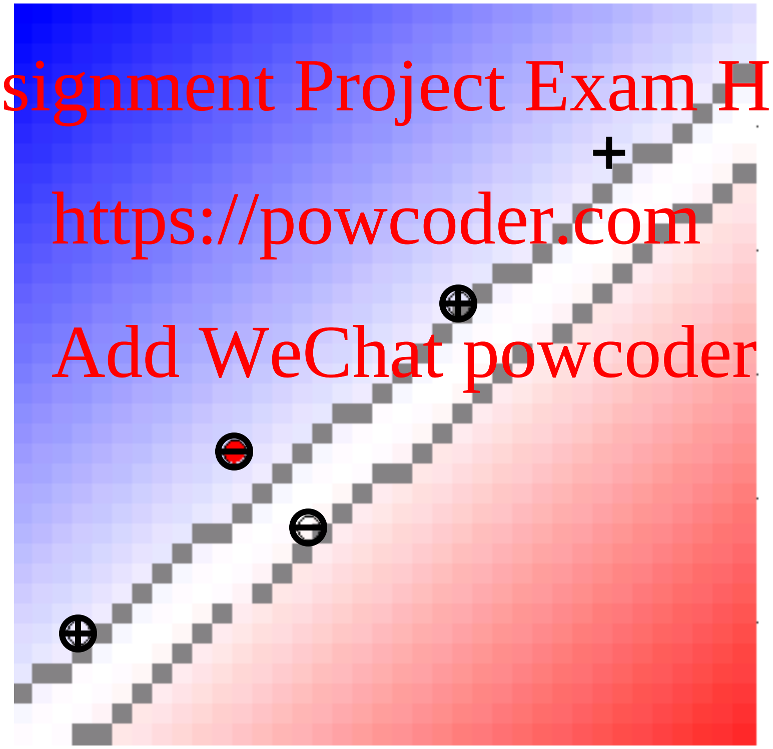
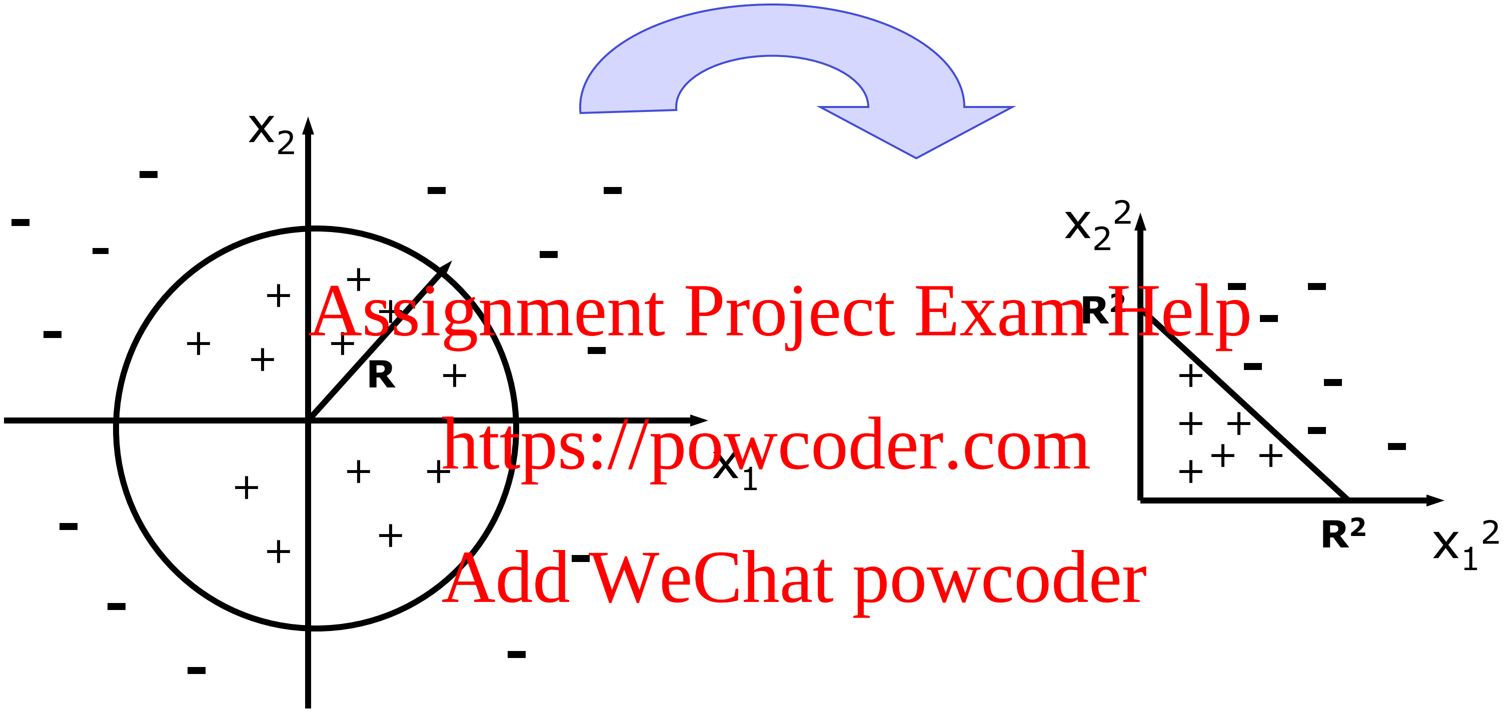


Image by Patrick Winston

Isn’t a linear classifier very limiting?

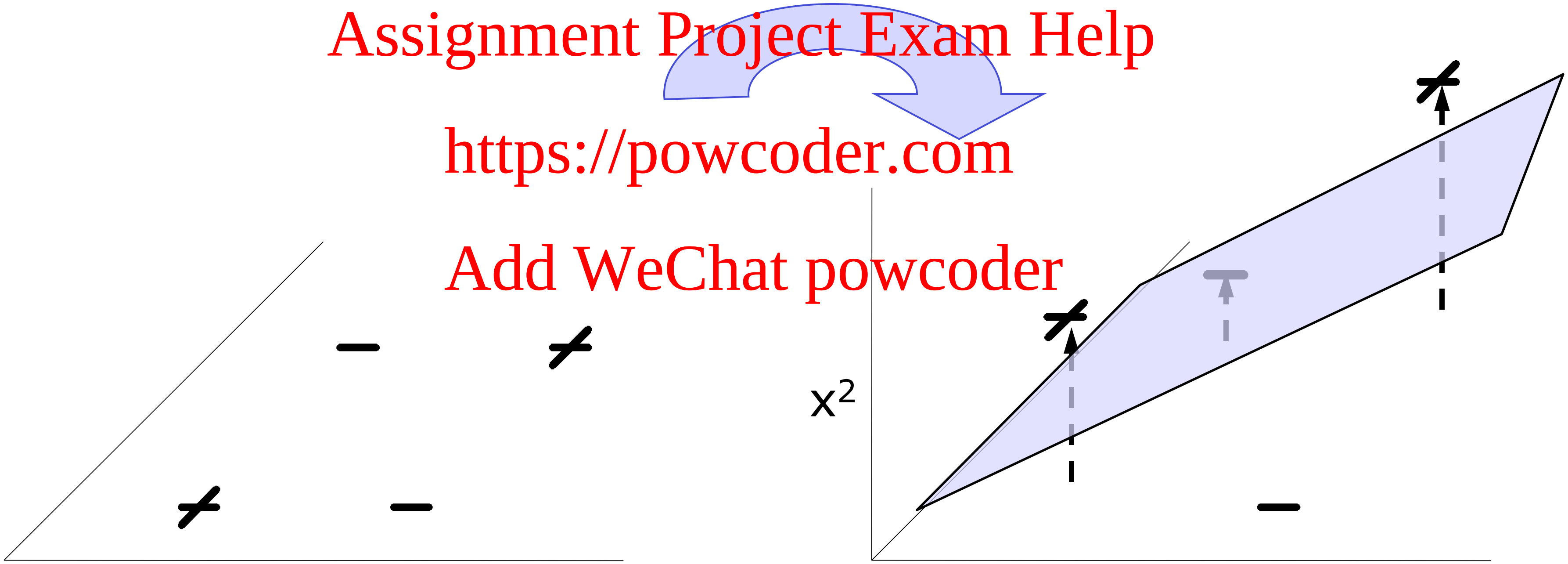


not linearly separable

linearly separable using squared value of features.

Important: Linear separator in transformed feature space maps into non-linear separator in original feature space

Not separable?
Try a higher dimensional space!



Not separable with 2D line

Separable with 3D plane

What you need

- To get into the new feature space, you use $\Phi(\mathbf{x}^i)$
- The transformation can be to a higher-dimensional feature space and may be non-linear in the feature values.

What you need

- To get into the new feature space, you use $\Phi(\mathbf{x}^i)$
- The transformation can be to a higher-dimensional feature space and may be non-linear in the feature values.
- Recall that SVM's only use dot products of the data, so
- To optimize classifier, you need $\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^k)$
- To run classifier, you need $\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{u})$
- So, all you need is a way to compute dot products in transformed space as a function of vectors in original space!

The “Kernel Trick”

- If dot products can be efficiently computed by

$$\Phi(\mathbf{x}^i)\cdot\Phi(\mathbf{x}^k) = K(\mathbf{x}^i,\mathbf{x}^k)$$

- Then, all you need is a function on low-dim inputs

$$K(\mathbf{x}^i,\mathbf{x}^k)$$

- You don’t need ever to construct high-dimensional

$$\Phi(\mathbf{x}^i)$$

Standard Choices For Kernels

- No change (linear kernel)

$$\Phi(\mathbf{x}^i)\cdot\Phi(\mathbf{x}^k) = K(\mathbf{x}^i,\mathbf{x}^k) = \mathbf{x}^i\cdot\mathbf{x}^k$$

Standard Choices For Kernels

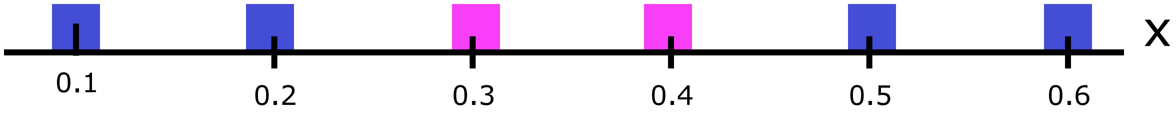
- No change (linear kernel)

$$\Phi(\mathbf{x}^i)\cdot\Phi(\mathbf{x}^k) = K(\mathbf{x}^i,\mathbf{x}^k) = \mathbf{x}^i\cdot\mathbf{x}^k$$

- Polynomial kernel (nth order)

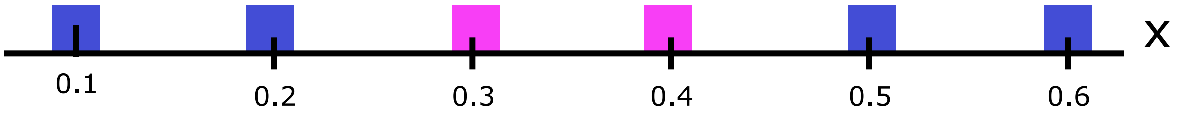
$$K(\mathbf{x}^i,\mathbf{x}^k) = (1+\mathbf{x}^i\cdot\mathbf{x}^k)^n$$

Polynomial Kernel Example
(one feature)



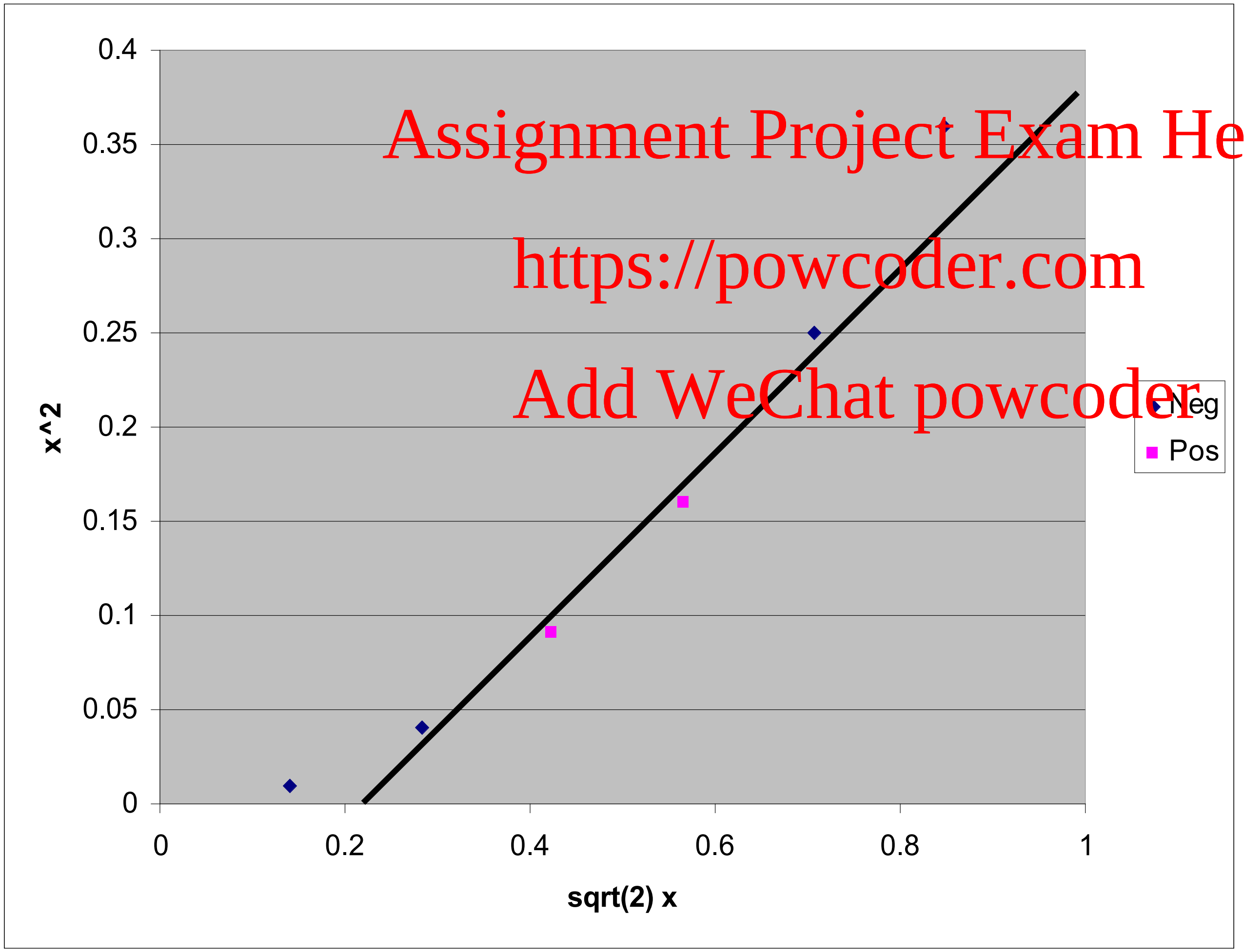
Not separable

Polynomial Kernel Example
(one feature)



Not separable

$$\Phi(x) = (x^2, \sqrt{2}x, 1)$$



Separable

$$\Phi(x) \cdot \Phi(z)$$
$$= x^2 z^2 + 2xz + 1$$
$$= (1 + xz)^2$$

Polynomial Kernel

- Polynomial kernel for n=2 and features **x**=[x$_1$ x$_2$]

$$K(\mathbf{x},\mathbf{z}) = (1 + \mathbf{x}\cdot\mathbf{z})^2$$

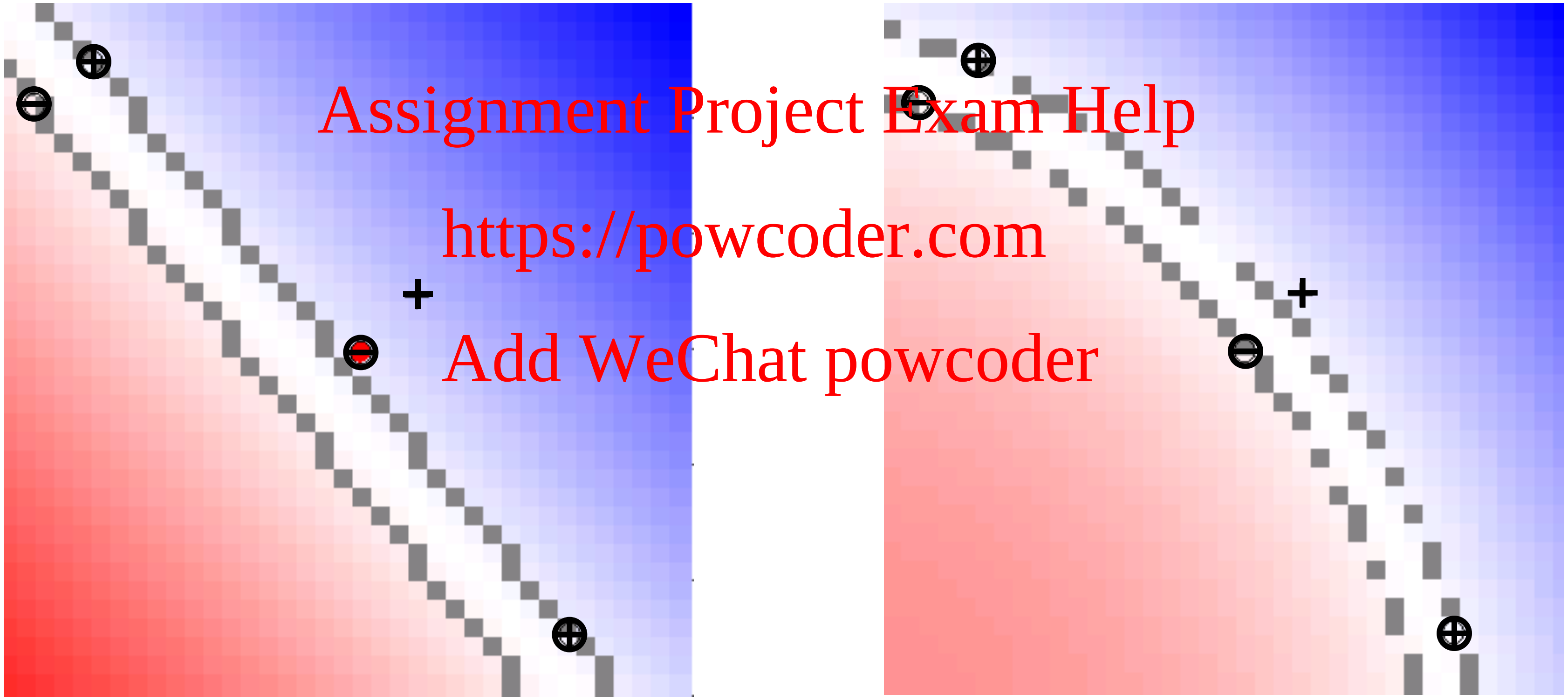
is equivalent to the following feature mapping:

$$\Phi(\mathbf{x}) = [x_1^2 \; x_2^2 \; \sqrt{2}x_1x_2 \; \sqrt{2}x_1 \; \sqrt{2}x_2 \; 1]$$

- We can verify that:

$$\begin{aligned}\Phi(\mathbf{x})\cdot\Phi(\mathbf{z}) &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1\\ &= (1 + x_1z_1 + x_2z_2)^2\\ &= (1 + \mathbf{x}\cdot\mathbf{z})^2\\ &= K(\mathbf{x},\mathbf{z})\end{aligned}$$

Polynomial Kernel



Images by Patrick Winston

Standard Choices For Kernels

- No change (linear kernel)

$$\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^k) = K(\mathbf{x}^i, \mathbf{x}^k) = \mathbf{x}^i \cdot \mathbf{x}^k$$

- Polynomial kernel (nth order)

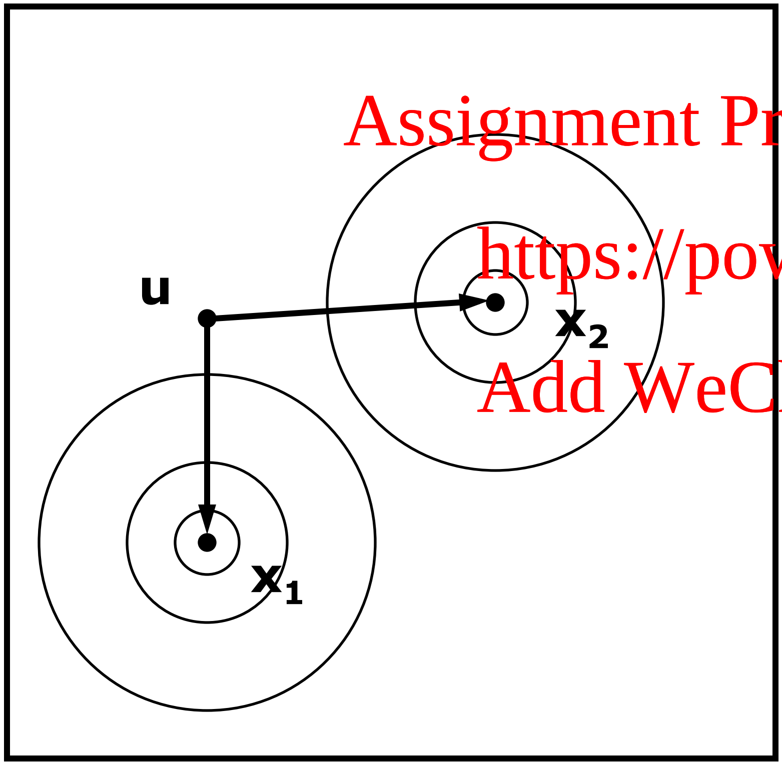
$$K(\mathbf{x}^i, \mathbf{x}^k) = (1 + \mathbf{x}^i \cdot \mathbf{x}^k)^n$$

- Radial basis kernel (σ is standard deviation)

$$K(\mathbf{x}^i, \mathbf{x}^k) = e^{\frac{-\left\|\mathbf{x}^i - \mathbf{x}^k\right\|^2}{2\sigma^2}} = e^{\frac{-(\mathbf{x}^i - \mathbf{x}^k)\cdot(\mathbf{x}^i - \mathbf{x}^k)}{2\sigma^2}}$$

Radial-basis kernel

- Classifier based on sum of Gaussian bumps with standard deviation σ, centered on support vectors.



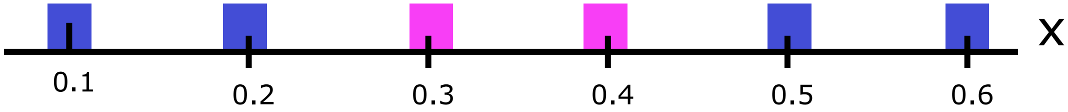
$$h(\mathbf{u}) = sign\left[h'(\mathbf{u})\right]$$

$$h'(\mathbf{u}) = \sum_{i=1}^{k} \alpha_i y^i K(\mathbf{x}^i, \mathbf{u}) + b$$

$$K(\mathbf{x}^i, \mathbf{u}) = e^{\frac{-\left\|\mathbf{x}^i - \mathbf{u}\right\|^2}{2\sigma^2}}$$

Radial-basis kernel

$\sigma = 0.1$



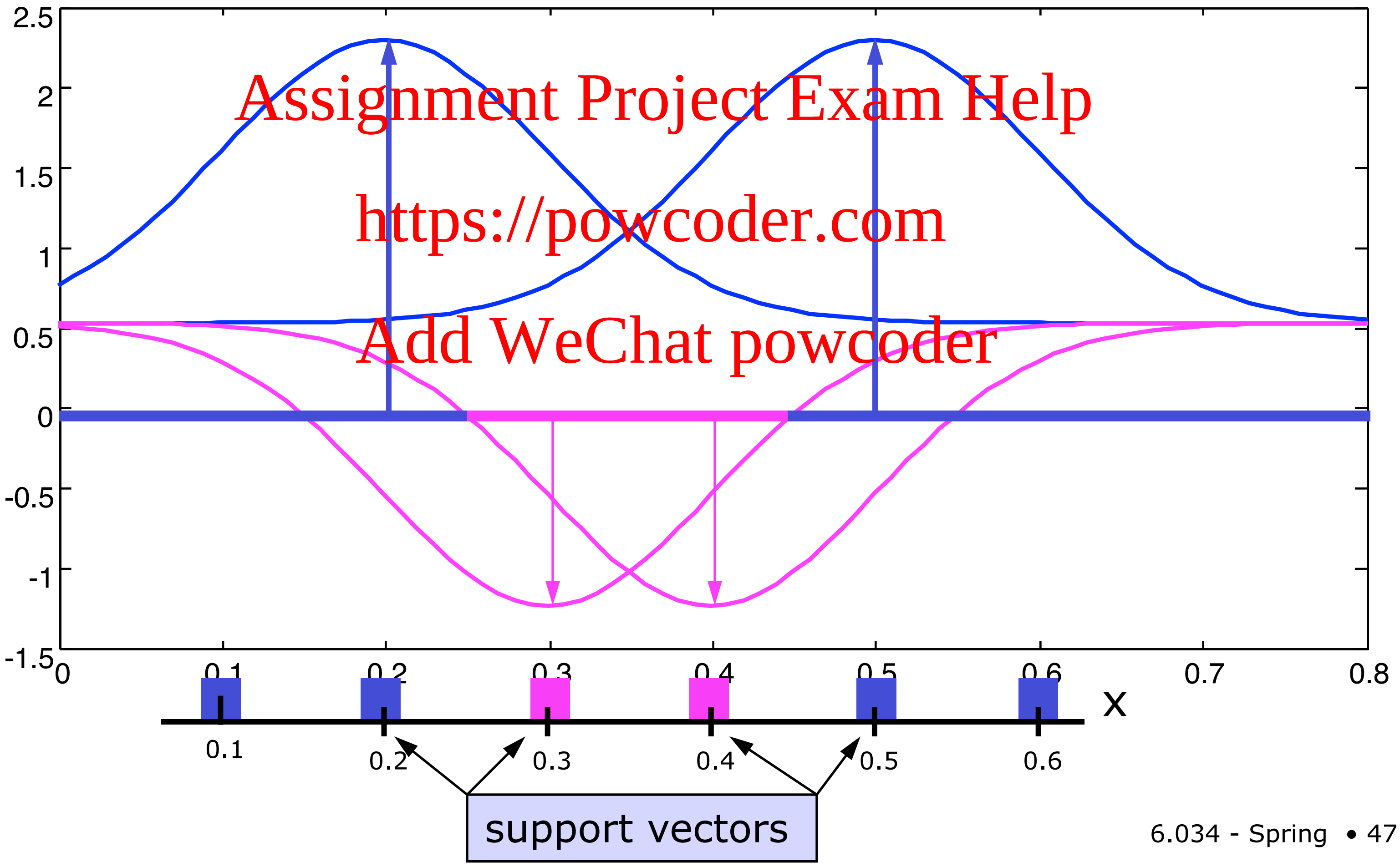
Radial-basis kernel

$y_1\alpha_1 = 1.76 \quad y_2\alpha_2 = -1.76$

$y_3\alpha_3 = 1.76 \quad y_4\alpha_4 = -1.76$

b = 0.525

σ = 0.1



support vectors

Radial-basis kernel

$y_1\alpha_1 = 1.76$ $y_2\alpha_2 = -1.76$

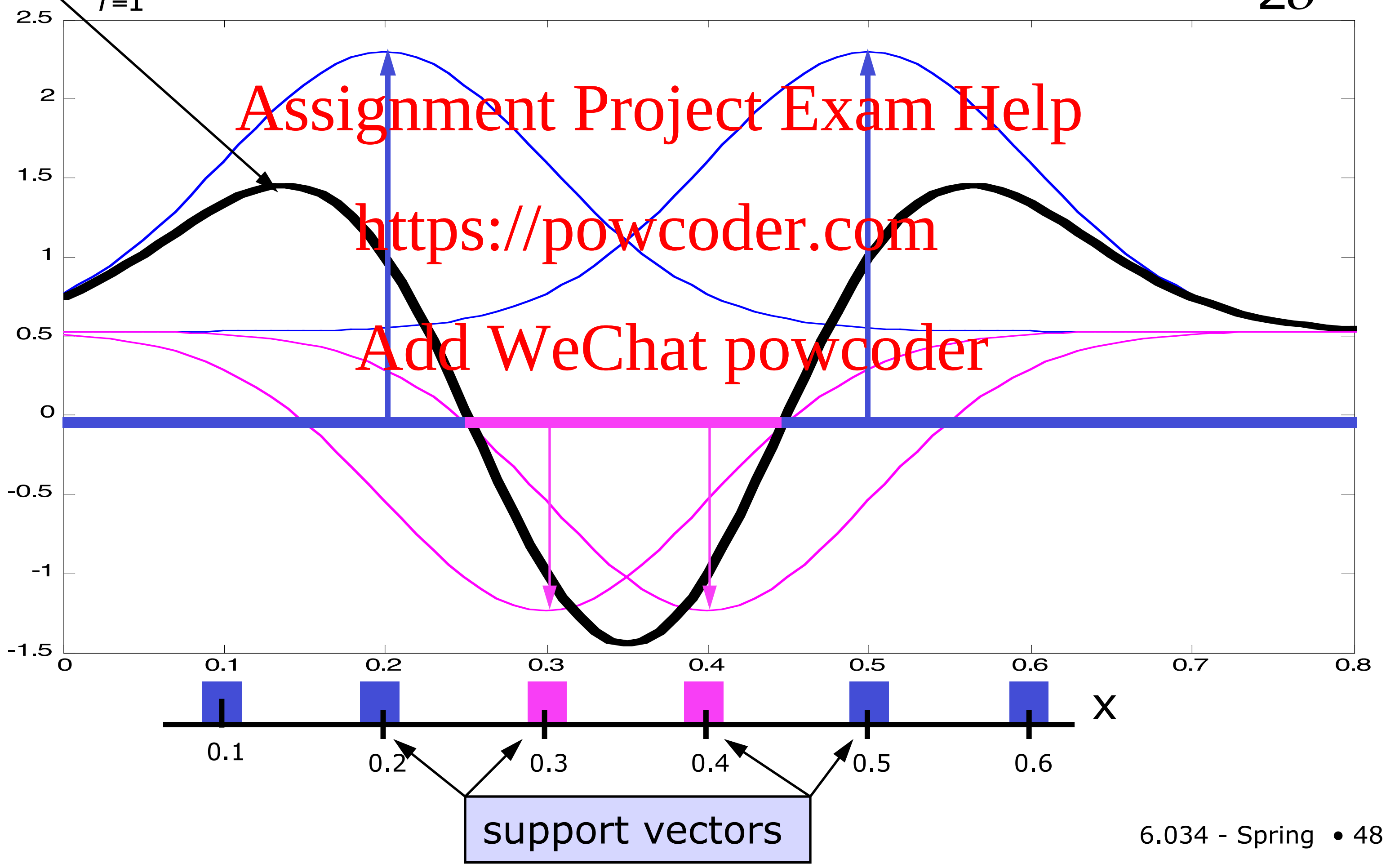
$y_3\alpha_3 = 1.76$ $y_4\alpha_4 = -1.76$

$b = 0.525$

$\sigma = 0.1$

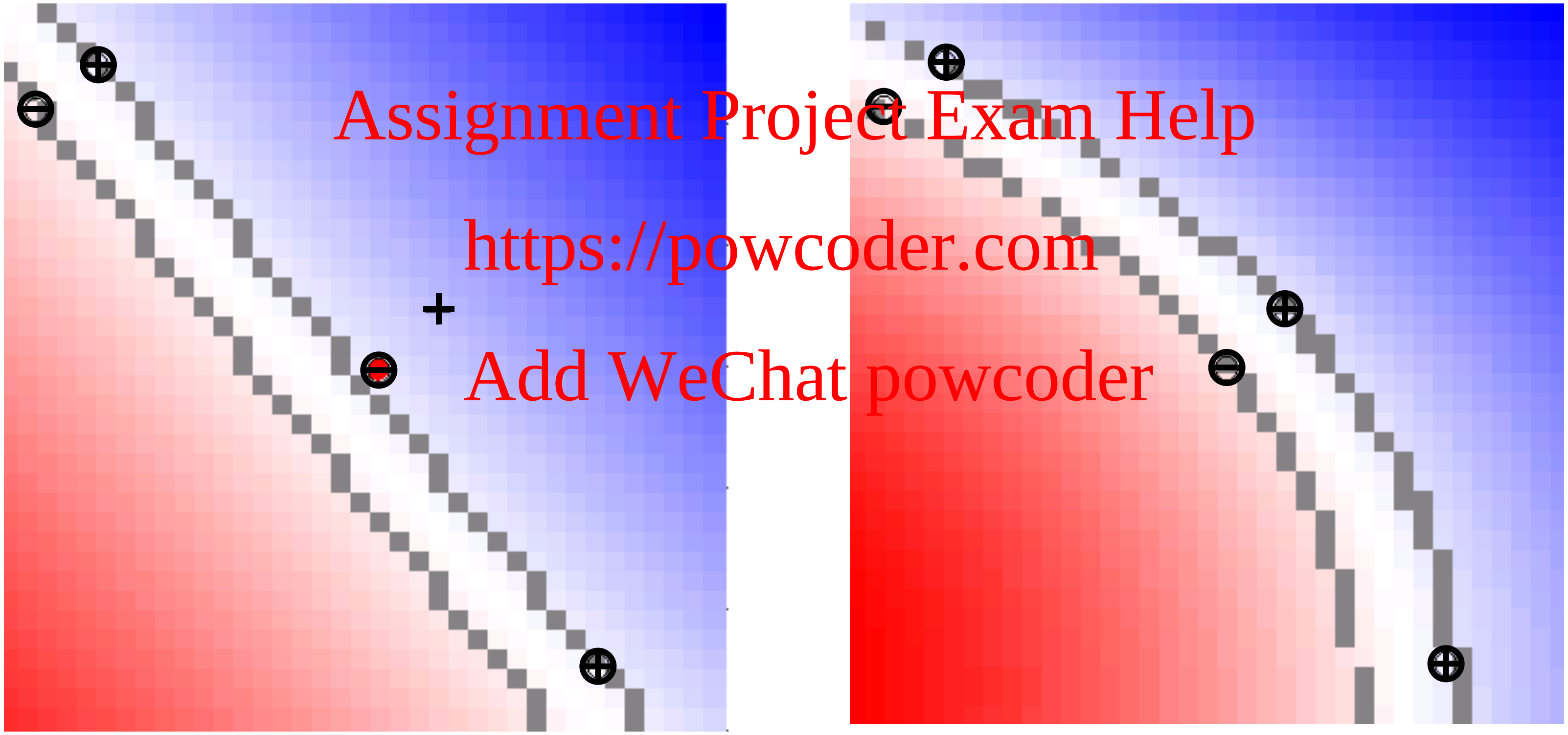
$$h'(\mathbf{u}) = \sum_{i=1}^{4} \alpha_i y^i K(\mathbf{x}^i, \mathbf{u}) + b$$

$$K(\mathbf{x}^i, \mathbf{u}) = e^{\frac{-\left\|\mathbf{x}^i - \mathbf{u}\right\|^2}{2\sigma^2}}$$



support vectors

Radial-basis kernel (large σ)



Images by Patrick Winston

Another radial-basis example (small σ)

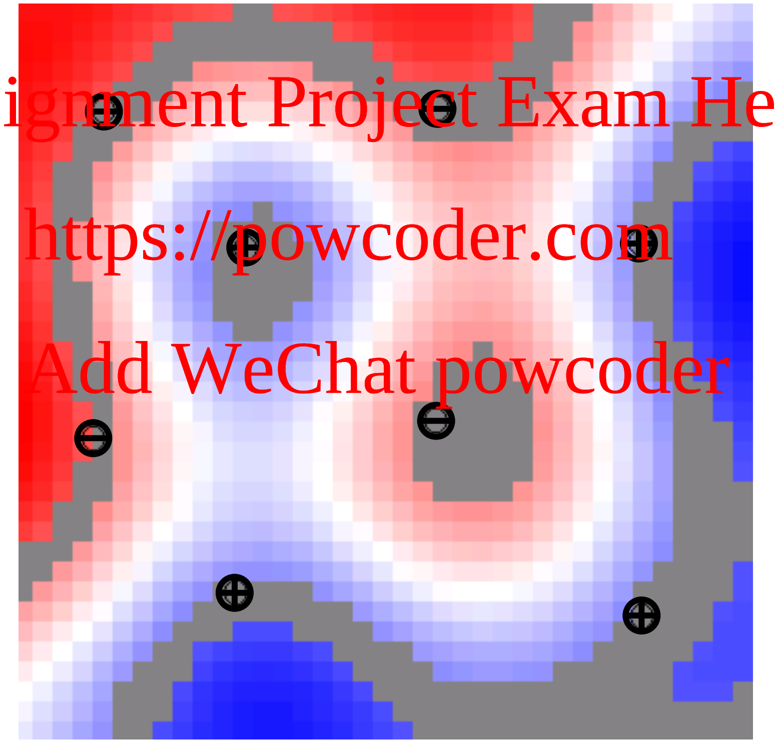


Image by Patrick Winston

Cross-Validation Error

- Does mapping to a very high-dimensional space lead to over-fitting?
- Generally, no, thanks to the fact that only the support vectors determine the decision surface.

Cross-Validation Error

- Does mapping to a very high-dimensional space lead to over-fitting?
- Generally, no, thanks to the fact that only the support vectors determine the decision surface.
- The expected leave-one-out cross-validation error depends on number of support vectors, not dimensionality of feature space.

$$\text{Expected CV error} \leq \frac{\text{Expected \# support vectors}}{\text{\# training samples}}$$

- If most data points are support vectors, a sign of possible overfitting, independent of the dimensionality of feature space.

Summary

- A single global maximum
  - Quadratic programming or gradient descent

Summary

- A single global maximum
  - Quadratic programming or gradient descent
- Fewer parameters
  - C and kernel parameters (n for polynomial, σ for radial basis kernel)

Summary

- A single global maximum
  - Quadratic programming or gradient descent
- Fewer parameters
  - C and kernel parameters (n for polynomial, σ for radial basis kernel)
- Kernel
  - Quadratic minimization depends only on dot products of sample vectors
  - Recognition depends only on dot products of unknown vector with sample vectors
  - Reliance on only dot products enables efficient feature mapping to higher-dimensional spaces where linear separation is more effective.

Real Data

- Wisconsin Breast Cancer Data
  - 9 features
  - C=1
  - 37 support vectors are used from 512 training data points
  - 12 prediction errors on training set (98% accuracy)
  - 96% accuracy on 171 held out points
  - Essentially same performance as nearest neighbors and decision trees
- Don't expect such good performance on every data set.

Success Stories

- Gene microarray data
  - outperformed all other classifiers
  - specially designed kernel
- Text categorization
  - linear kernel in >10,000 D input space
  - best prediction performance
  - 35 times faster to train than next best classifier (decision trees)
- Many others: http://www.clopinet.com/isabelle/Projects/SVM/applist.html