Announcements

Assignments

- HW7 (written + programming)
  - Due Tue 3/31, 11:59 pm
- HW8 (written + programming)
  - Out this week
  - Due Tue 4/7, 11:59 pm

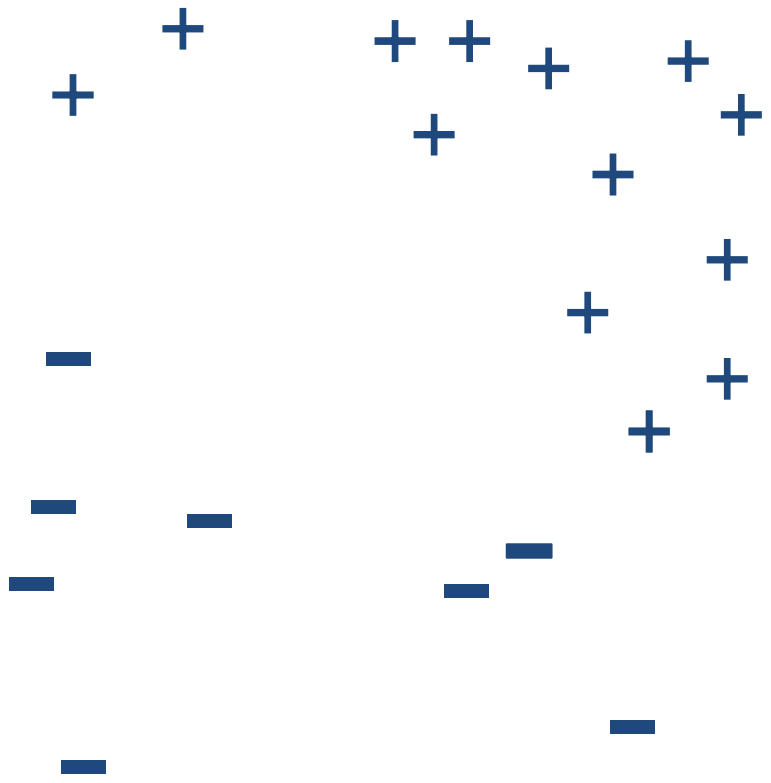
Introduction to Machine Learning

Support Vector Machines

Instructor: Pat Virtue

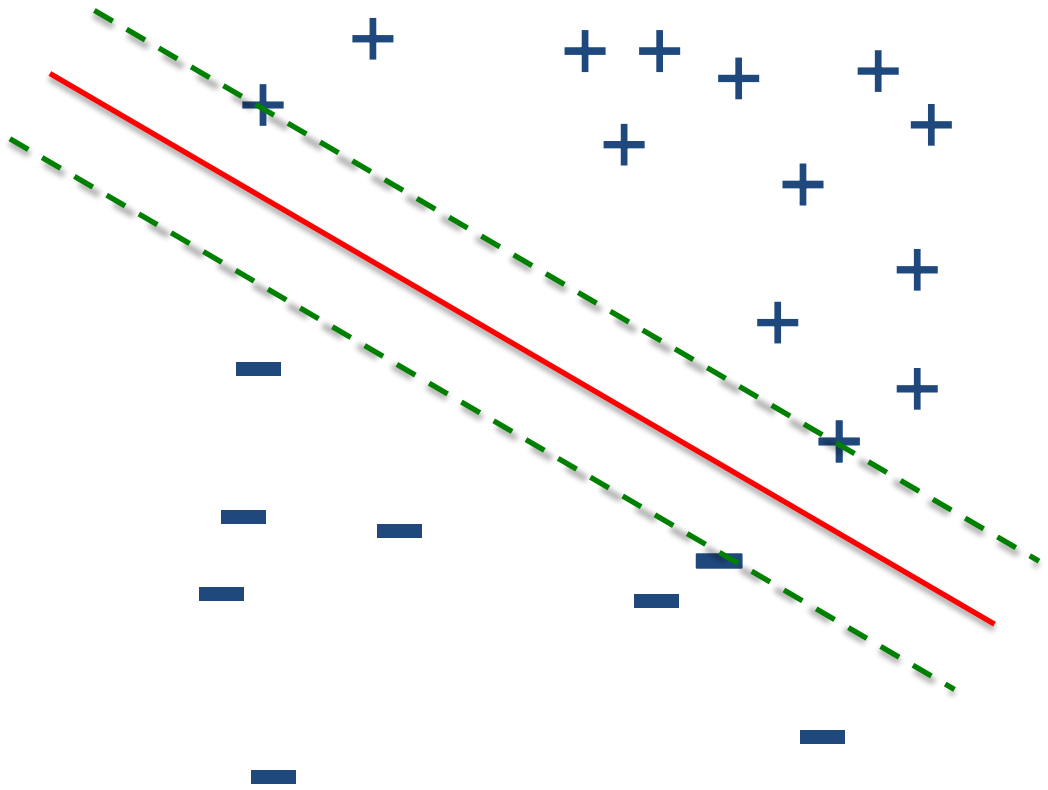
Support Vector Machines

Find linear separator with maximum margin



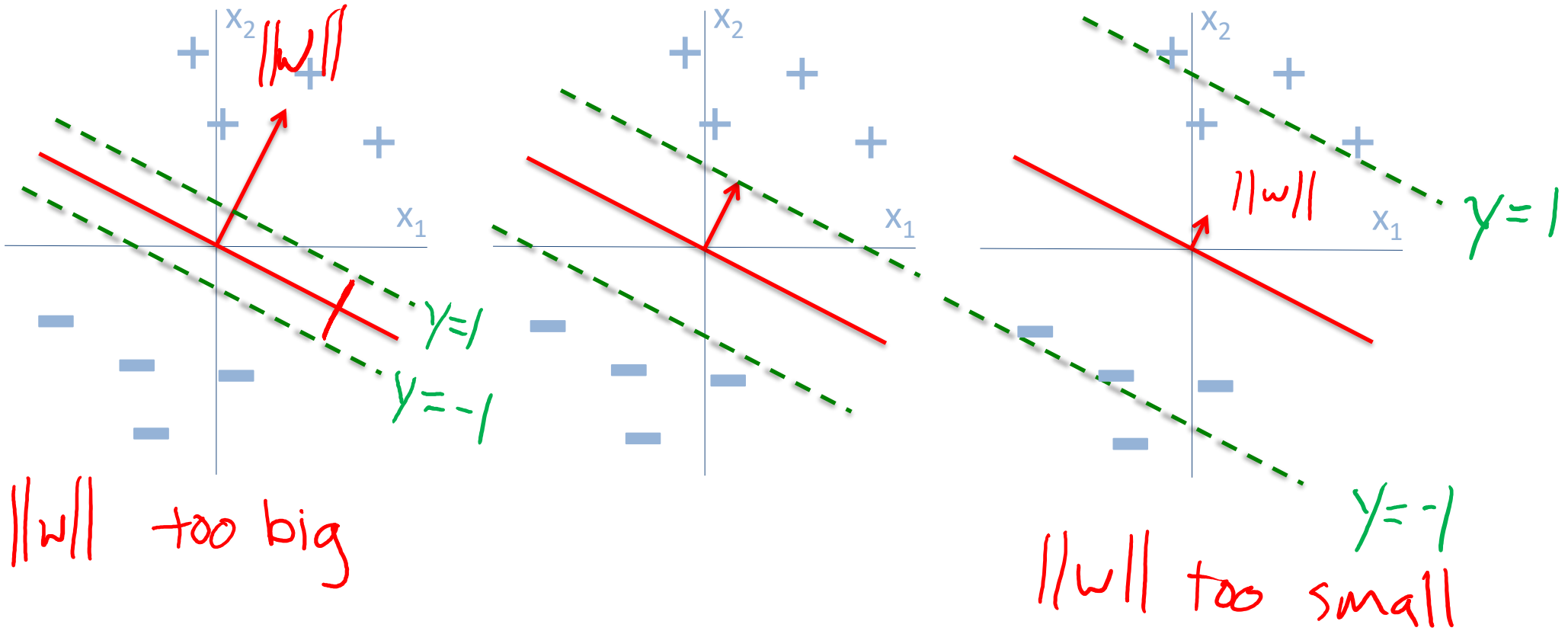
Support Vector Machines

Find linear separator with maximum margin



Support Vector Machines

Find linear separator with maximum margin



Linear Separability

Data

$$\mathcal{D} = \left\{\boldsymbol{x}^{(i)}, y^{(i)}\right\}_{i=1}^{N} \quad \boldsymbol{x} \in \mathbb{R}^M, \;\; y \in \{-1, +1\}$$

Linearly separable iff:

$$\exists\, \boldsymbol{w}, b \qquad s.t. \quad \boldsymbol{w}^T \boldsymbol{x}^{(i)} + b > 0 \quad \text{if} \quad y^{(i)} = +1 \quad \text{and}$$
$$\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b < 0 \quad \text{if} \quad y^{(i)} = -1$$

Slide credit: CMU MLD Matt Gormley

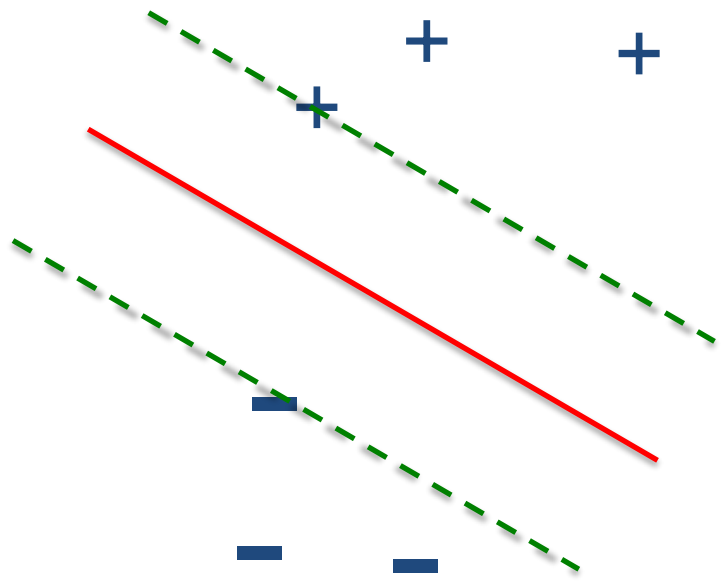
Support Vector Machines

$$width = \frac{\boldsymbol{w}^T}{\|\boldsymbol{w}\|_2}(\boldsymbol{x}_+ - \boldsymbol{x}_-)$$

Find linear separator with maximum margin

$$\max_{\boldsymbol{w},\boldsymbol{b}} \quad \text{"width"}$$

$$\text{s.t.} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 \quad \forall\, i$$



Support Vector Machines

$$width = \frac{w^T}{\|w\|_2}(x_+ - x_-)$$

Find linear separator with maximum margin

$$\max_{w,b} \quad \text{"width"}$$

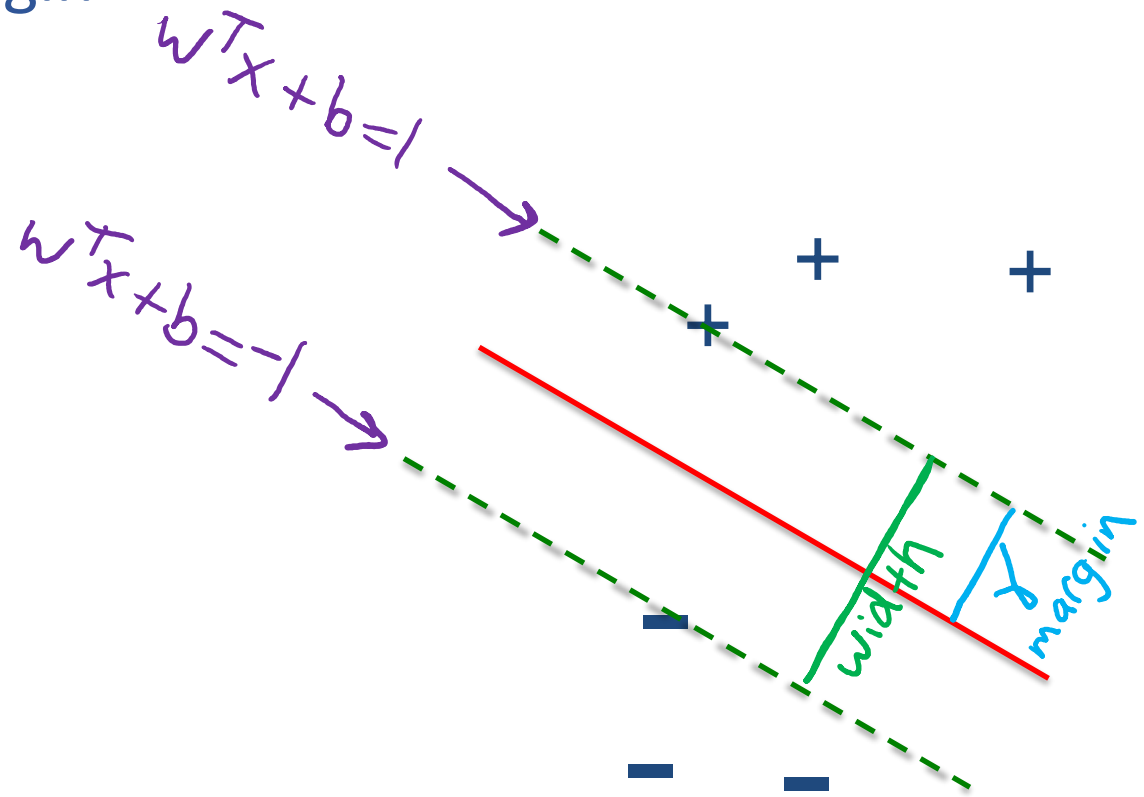
$$\text{s.t.} \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 \quad \forall\, i$$

$$\text{width} = \frac{w^T}{\|w\|_2}(x_+ - x_-)$$

$$= \frac{1}{\|w\|_2}\left(w^T x_+ - w^T x_-\right)$$

$$= \frac{1}{\|w\|_2}\left((1-b) - (-1-b)\right)$$

$$= \frac{2}{\|w\|_2}$$



$w^T x_+ + b = 1$ $\qquad$ $w^T x_- + b = -1$

$w^T x_+ = 1 - b$ $\qquad$ $w^T x_- = -1 - b$

SVM Optimization

Quadratic program!

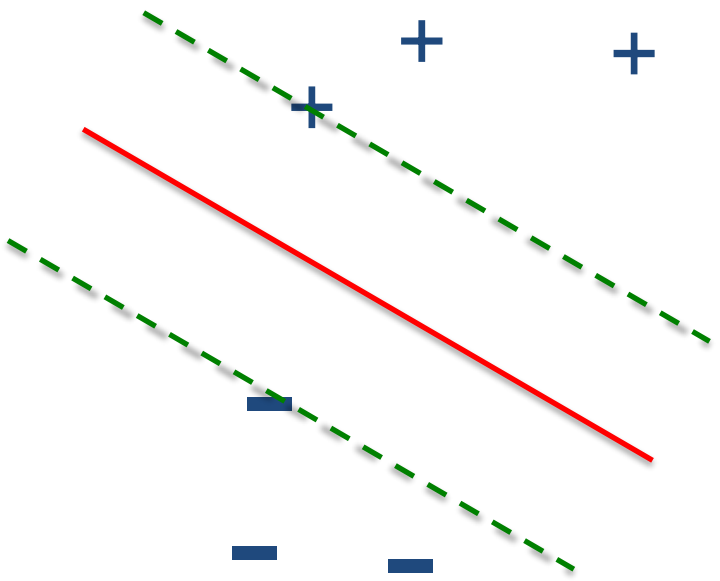
$$\min_{\boldsymbol{w},\boldsymbol{b}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t.} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 \quad \forall\, i$$

Quadratic Program

$$\min_{\boldsymbol{x}} \quad \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$

$$\text{s.t.} \quad A\boldsymbol{x} \preceq \boldsymbol{b}$$



Constrained Optimization

Linear Program

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T \boldsymbol{x}$$

$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}$$

Solvers

- Simplex
- Interior point methods

Quadratic Program

$$\min_{\boldsymbol{x}} \quad \boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x}$$

$$\text{s.t.} \quad A\boldsymbol{x} \preceq \boldsymbol{b}$$

Solvers

- Conjugate gradient
- Ellipsoid method
- Interior point methods

Constrained Optimization

Linear Program

$$\min_{\boldsymbol{x}} \quad \boldsymbol{c}^T \boldsymbol{x}$$
$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}$$

Quadratic Program

$$\min_{\boldsymbol{x}} \quad \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x}$$
$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}$$

Solvers

- Simplex
- Interior point methods

Special Case

- If $\boldsymbol{Q}$ is positive-definite, the problem is convex
- $\boldsymbol{Q}$ is positive-definite if:
  $\boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{v} > 0 \quad \forall\, \boldsymbol{v} \in \mathbb{R}^M \backslash \boldsymbol{0}$

Support Vector Machines

Next steps

- Different optimization formulation
  - Primal → dual
  - “Support vectors”
- Support non-linear classification
  - Feature maps
  - Kernel trick
- Support non-separable data
  - Hard-margin SVM → soft-margin SVM

Method of Lagrange Multipliers

Goal

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x})$$

$$\text{s.t.} \quad g(\boldsymbol{x}) \leq c$$

Step 1: Construct Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda(g(\boldsymbol{x}) - c)$$

Step 2: Solve

$$\min_{\boldsymbol{x}} \quad \max_{\lambda \geq 0} \quad \mathcal{L}(\boldsymbol{x}, \lambda)$$

Find saddle point:

$$\nabla \mathcal{L}(\boldsymbol{x}, \lambda) \quad \text{s.t. } \lambda \geq 0$$

Equivalent to solving:

$$\nabla f(x) = \lambda \nabla g(x) \quad \text{s.t. } \lambda \geq 0$$

Slide credit: CMU MLD Matt Gormley

SVM Primal vs Dual

Construct Lagrangian

Primal

$$\min_{\boldsymbol{w},\boldsymbol{b}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t.} \quad y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 \quad \forall\, i$$

Lagrange Multipliers

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) \quad \text{s.t.} \quad g(\boldsymbol{x}) \leq c$$

Construct Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda(g(\boldsymbol{x}) - c)$$

Solve: $\min_{\boldsymbol{x}} \quad \max_{\lambda \geq 0} \quad \mathcal{L}(\boldsymbol{x}, \lambda)$

SVM Dual Optimization

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^t w \; - \sum_i^N \alpha_i [\, y^{(i)} \left(\boldsymbol{w}^T \boldsymbol{x}^{(i)} + b \right) - 1]$$

SVM Dual Optimization

Dual

$$\max_{\boldsymbol{\alpha}} \quad \sum_i^N \alpha_i - \frac{1}{2}\sum_i^N \sum_j^N \alpha_i \alpha_j y^{(i)} y^{(i)} \boldsymbol{x}^{(i)T} \boldsymbol{x}^{(j)}$$

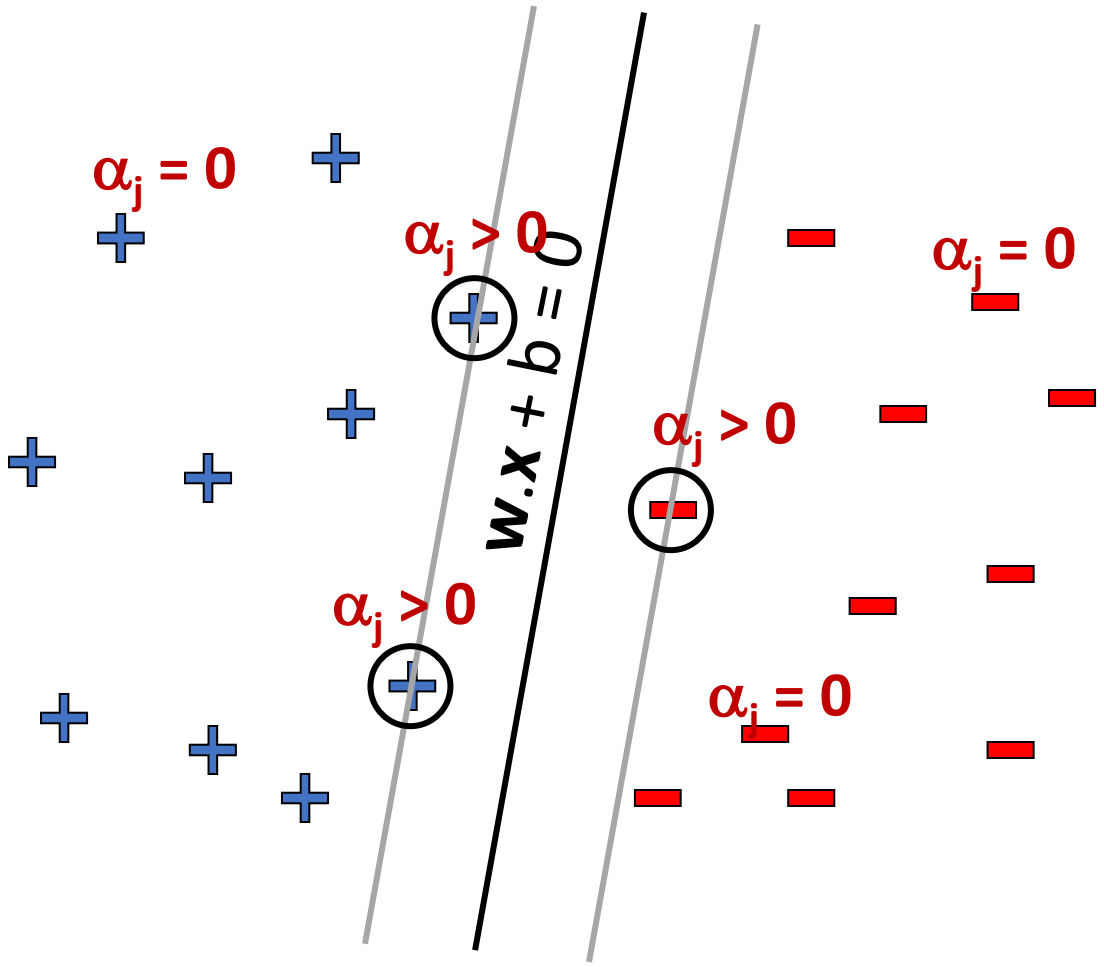
$$\text{s.t.} \quad \alpha_i \geq 0 \quad \forall\, i$$

$$w = \sum_i^N \alpha_i y^{(i)} \boldsymbol{x}^{(i)}$$

$b = y^{(k)} - w^T x^{(k)}$ for any k where $\alpha_k > 0$

Prediction

Dual SVM: Sparsity of dual solution



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Only few α_js can be non-zero : where constraint is active and tight

$$(\mathbf{w}.\mathbf{x}_j + b)y_j = 1$$

Support vectors – training points j whose α_js are non-zero

Slide credit: CMU MLD Aarti Singh

Support Vector Machines

Next steps

- Different optimization formulation
  - Primal → dual
  - “Support vectors”
- Support non-linear classification
  - Feature maps
  - Kernel trick
- Support non-separable data
  - Hard-margin SVM → soft-margin SVM

Kernels: Motivation

Most real-world problems exhibit data that is not linearly separable.

Example: pixel representation for Facial Recognition:

Q: When your data is **not linearly separable**, how can you still use a linear classifier?

A: Preprocess the data to produce **nonlinear features**

Kernels: Motivation

- Motivation #1: Inefficient Features
  - Non-linearly separable data requires **high dimensional** representation
  - Might be **prohibitively expensive** to compute or store
- Motivation #2: Memory-based Methods
  - k-Nearest Neighbors (KNN) for facial recognition allows a **distance metric** between images -- no need to worry about linearity restriction at all

Kernel Methods

- **Key idea:**
  1. Rewrite the algorithm so that we only work with **dot products** $x^T z$ of feature vectors
  2. Replace the **dot products** $x^T z$ with a **kernel function** $k(x, z)$

- The kernel $k(x,z)$ can be **any** legal definition of a dot product:

  $$k(x, z) = \varphi(x)^T \varphi(z) \text{ for any function } \varphi: \mathcal{X} \rightarrow \mathbf{R}^D$$

  So we only compute the φ dot product **implicitly**

- This **"kernel trick"** can be applied to many algorithms:
  - classification: perceptron, SVM, …
  - regression: ridge regression, …
  - clustering: k-means, …

SVM: Kernel Trick

Hard-margin SVM (Primal)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad \forall i$$

- Suppose we do some feature engineering
- Our feature function is ϕ
- We apply ϕ to each input vector $\mathbf{x}$

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\phi\left(\mathbf{x}^{(i)}\right) + b) \geq 1, \quad \forall i$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^{(i)}y^{(j)}\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \quad \forall i = 1,\ldots,N$$

$$\sum_{i=1}^{N}\alpha_i y^{(i)} = 0$$

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y^{(i)}y^{(j)}\phi\left(\mathbf{x}^{(i)}\right)\cdot\phi\left(\mathbf{x}^{(j)}\right)$$

$$\text{s.t. } \alpha_i \geq 0, \quad \forall i = 1,\ldots,N$$

$$\sum_{i=1}^{N}\alpha_i y^{(i)} = 0$$

Slide credit: CMU MLD Matt Gormley

SVM: Kernel Trick

Hard-margin SVM (Lagrangian Dual)

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \boxed{\phi\left(\mathbf{x}^{(i)}\right) \cdot \phi\left(\mathbf{x}^{(j)}\right)}$$

$$\text{s.t. } \alpha_i \geq 0, \quad \forall i = 1, \ldots, N$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

We could replace the dot product of the two feature vectors in the transformed space with a function k(x,z)
where $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi\left(\mathbf{x}^{(i)}\right) \cdot \phi\left(\mathbf{x}^{(j)}\right)$

Slide credit: CMU MLD Matt Gormley

SVM: Kernel Trick

Hard-margin SVM (Lagrangian Dual)

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

$$\text{s.t. } \alpha_i \geq 0, \quad \forall i = 1, \ldots, N$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

We could replace the dot product of the two feature vectors in the transformed space with a function k(x,z)
where $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi\left(\mathbf{x}^{(i)}\right) \cdot \phi\left(\mathbf{x}^{(j)}\right)$

Slide credit: CMU MLD Matt Gormley

Kernel Methods

- **Key idea:**
  1. Rewrite the algorithm so that we only work with **dot products** $x^T z$ of feature vectors
  2. Replace the **dot products** $x^T z$ with a **kernel function** $k(x, z)$

- The kernel $k(x,z)$ can be **any** legal definition of a dot product:

  $$k(x, z) = \varphi(x)^T \varphi(z) \text{ for any function } \varphi: \mathcal{X} \rightarrow \mathbf{R}^D$$

  So we only compute the φ dot product **implicitly**

- This **"kernel trick"** can be applied to many algorithms:
  - classification: perceptron, SVM, …
  - regression: ridge regression, …
  - clustering: k-means, …

Kernel Methods

Q: These are just non-linear features, right?

A: Yes, but…

Q: Can't we just compute the feature transformation φ explicitly?

A: That depends…

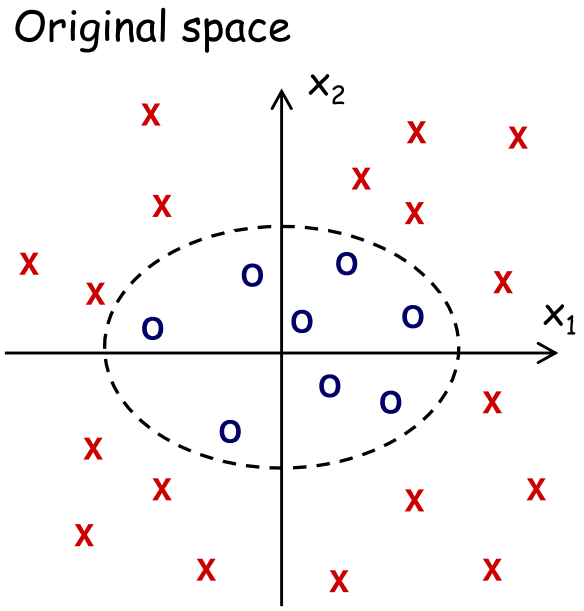
Q: So, why all the hype about the kernel trick?

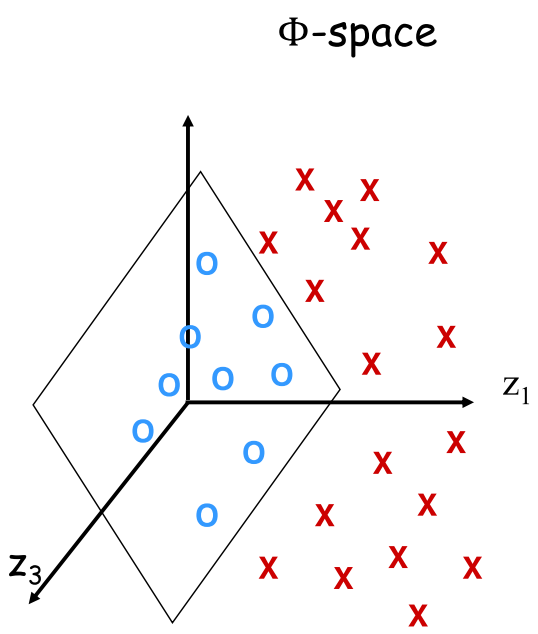
A: Because the **explicit features** might either be **prohibitively expensive** to compute or **infinite length** vectors

Slide credit: CMU MLD Matt Gormley

Example: Polynomial Kernel

https://www.youtube.com/watch?v=3liCbRZPrZA





Slide credit: CMU MLD Nina Balcan

Example: Polynomial Kernel

For n=2, d=2, the kernel $K(x, z) = (x \cdot z)^d$ corresponds to

$$\phi: R^2 \to R^3, (x_1, x_2) \to \Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(x) \cdot \phi(z) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2)$$

$$= (x_1z_1 + x_2z_2)^2 = (x \cdot z)^2 = K(x, z)$$

Slide credit: CMU MLD Nina Balcan

Kernel Examples

Side Note: The feature space might not be unique!

Explicit representation #1:

$$\phi: \mathrm{R}^2 \to \mathrm{R}^3, (\mathrm{x}_1, \mathrm{x}_2) \to \Phi(\mathrm{x}) = (\mathrm{x}_1^2, \mathrm{x}_2^2, \sqrt{2}\mathrm{x}_1\mathrm{x}_2)$$

$$\phi(\mathrm{x}) \cdot \phi(z) = \left(\mathrm{x}_1^2, \mathrm{x}_2^2, \sqrt{2}\mathrm{x}_1\mathrm{x}_2\right) \cdot (z_1^2, z_2^2, \sqrt{2}z_1z_2)$$

$$= (\mathrm{x}_1z_1 + \mathrm{x}_2z_2)^2 = (\mathrm{x} \cdot z)^2 = \mathrm{K}(\mathrm{x}, \mathrm{z})$$

Explicit representation #2:

$$\phi: \mathrm{R}^2 \to \mathrm{R}^4, (\mathrm{x}_1, \mathrm{x}_2) \to \Phi(\mathrm{x}) = (\mathrm{x}_1^2, \mathrm{x}_2^2, \mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2\mathrm{x}_1)$$

$$\phi(\mathrm{x}) \cdot \phi(z) = (\mathrm{x}_1^2, \mathrm{x}_2^2, \mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2\mathrm{x}_1) \cdot (\mathrm{z}_1^2, \mathrm{z}_2^2, \mathrm{z}_1\mathrm{z}_2, \mathrm{z}_2\mathrm{z}_1)$$

$$= (\mathrm{x} \cdot z)^2 = \mathrm{K}(\mathrm{x}, \mathrm{z})$$

These two different feature representations correspond to the same kernel function!

Slide credit: CMU MLD Nina Balcan

Kernel Examples

Name	Kernel Function (implicit dot product)	Feature Space (explicit dot product)
Linear	$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$	Same as original input space
Polynomial (v1)	$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z})^d$	All polynomials **of** degree d
Polynomial (v2)	$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z} + 1)^d$	All polynomials **up to** degree d
Gaussian (RBF)	$K(\mathbf{x}, \mathbf{z}) = \exp(-\frac{\|\|\mathbf{x} - \mathbf{z}\|\|_2^2}{2\sigma^2})$	Infinite dimensional space
Hyperbolic Tangent (Sigmoid) Kernel	$K(\mathbf{x}, \mathbf{z}) = \tanh(\alpha\mathbf{x}^T\mathbf{z} + c)$	(With SVM, this is equivalent to a 2-layer neural network)

Slide credit: CMU MLD Matt Gormley

Kernels: Mercer's Theorem

What functions are valid kernels that correspond to feature vectors $\varphi(\mathbf{x})$?

Answer: Mercer kernels K

- K is continuous
- K is symmetric
- K is positive semi-definite, i.e. $\mathbf{x}^T K \mathbf{x} \geq 0$ for all $\mathbf{x}$

Slide credit: CMU MLD Aarti Singh

SVMs with Kernels

- Choose a set of features and kernel function
- Solve dual problem to obtain support vectors α_i
- At classification time, compute:

$$\mathbf{w} \cdot \Phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i)$$

$$b = y_k - \sum_i \alpha_i y_i K(\mathbf{x}_k, \mathbf{x}_i)$$

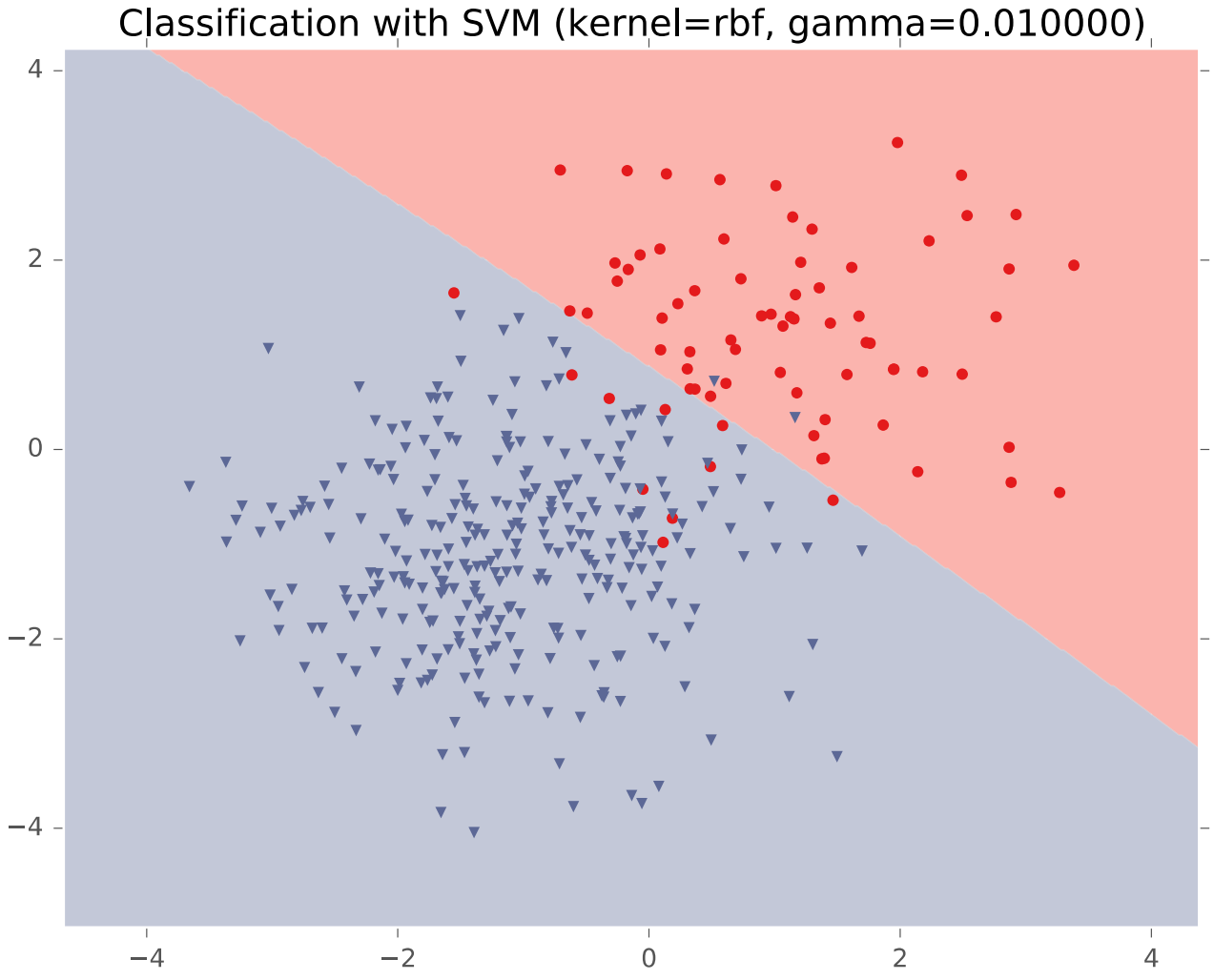
for any k where $C > \alpha_k > 0$

Classify as

$$sign\left(\mathbf{w} \cdot \Phi(\mathbf{x}) + b\right)$$

Slide credit: CMU MLD Aarti Singh

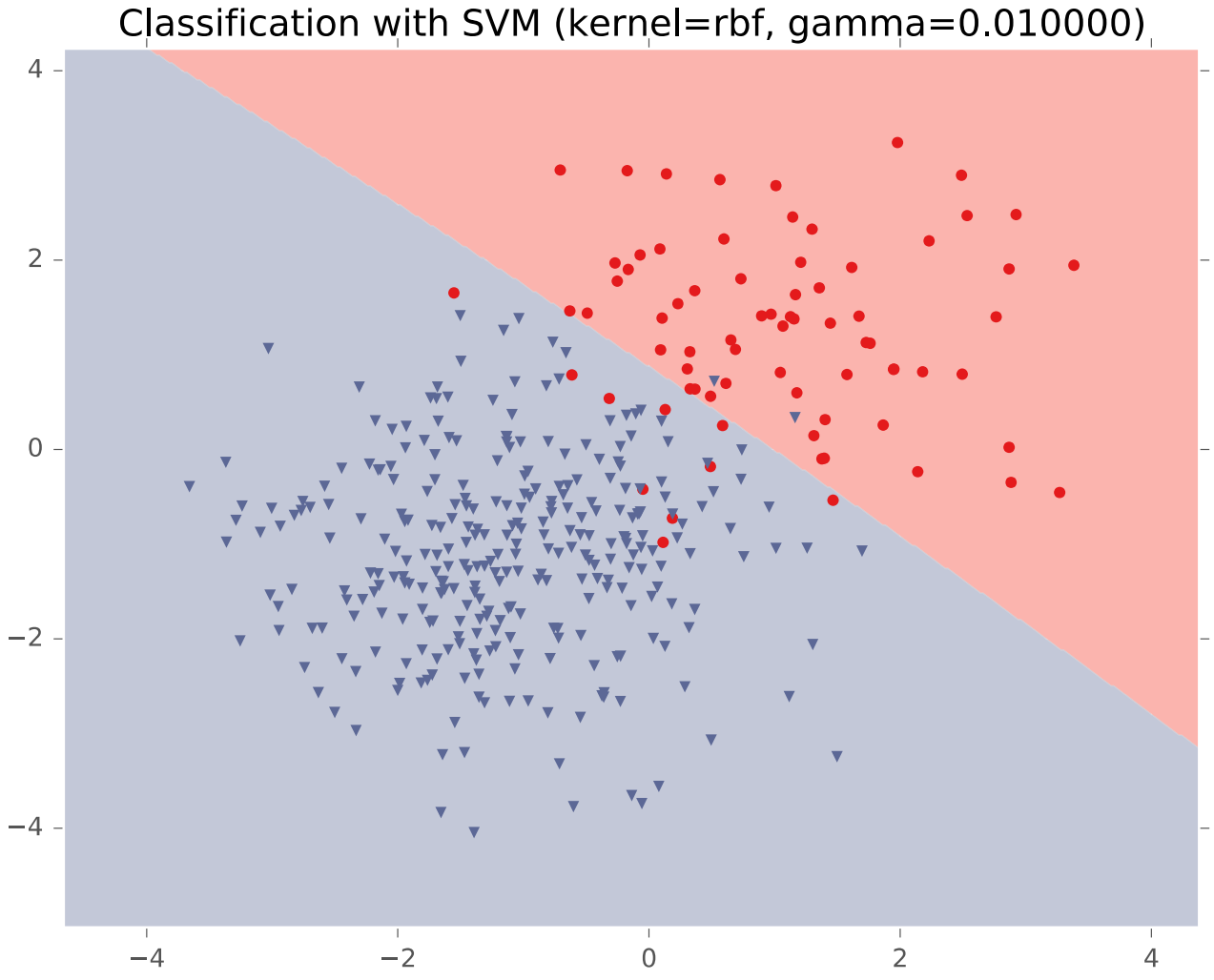
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

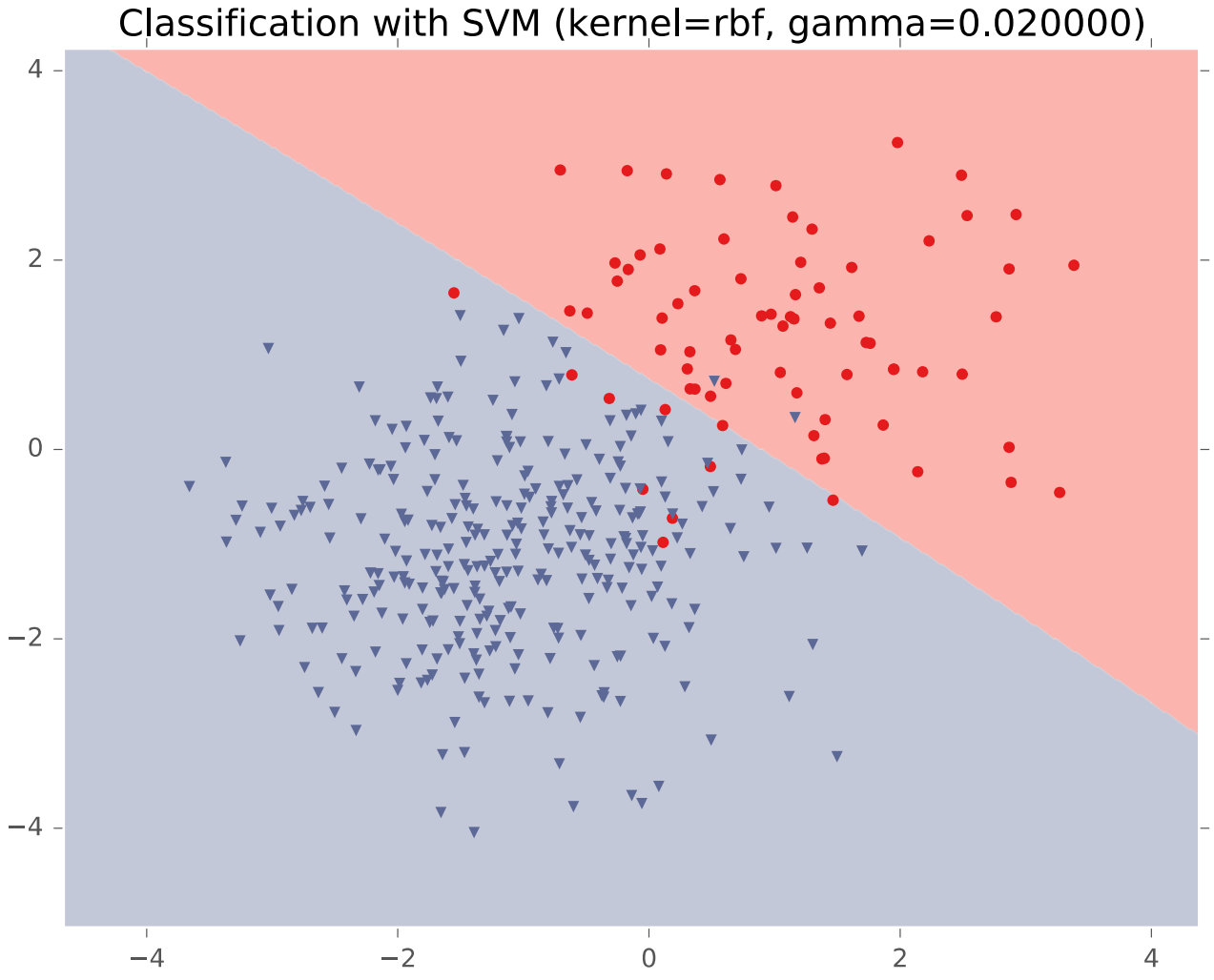
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

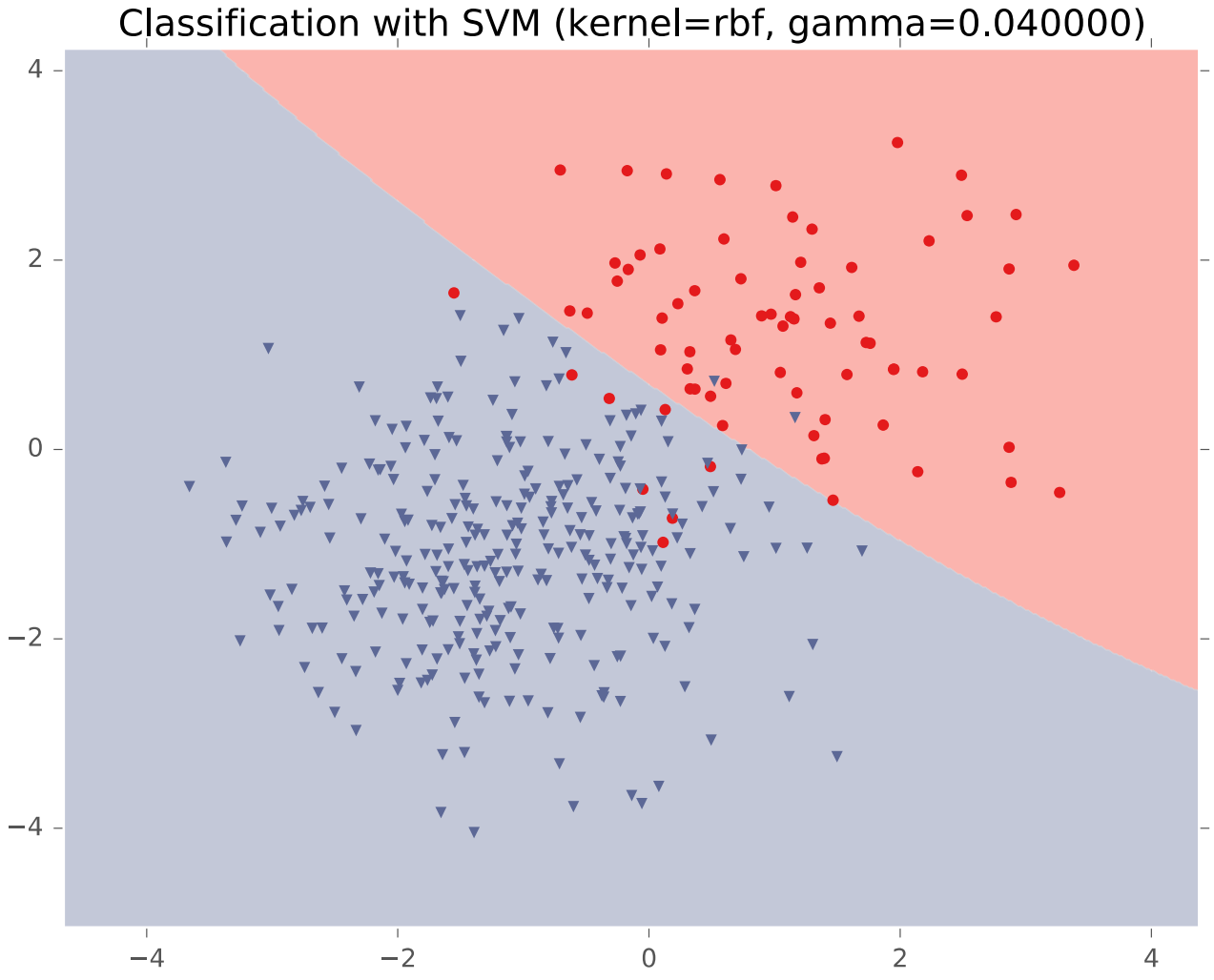
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

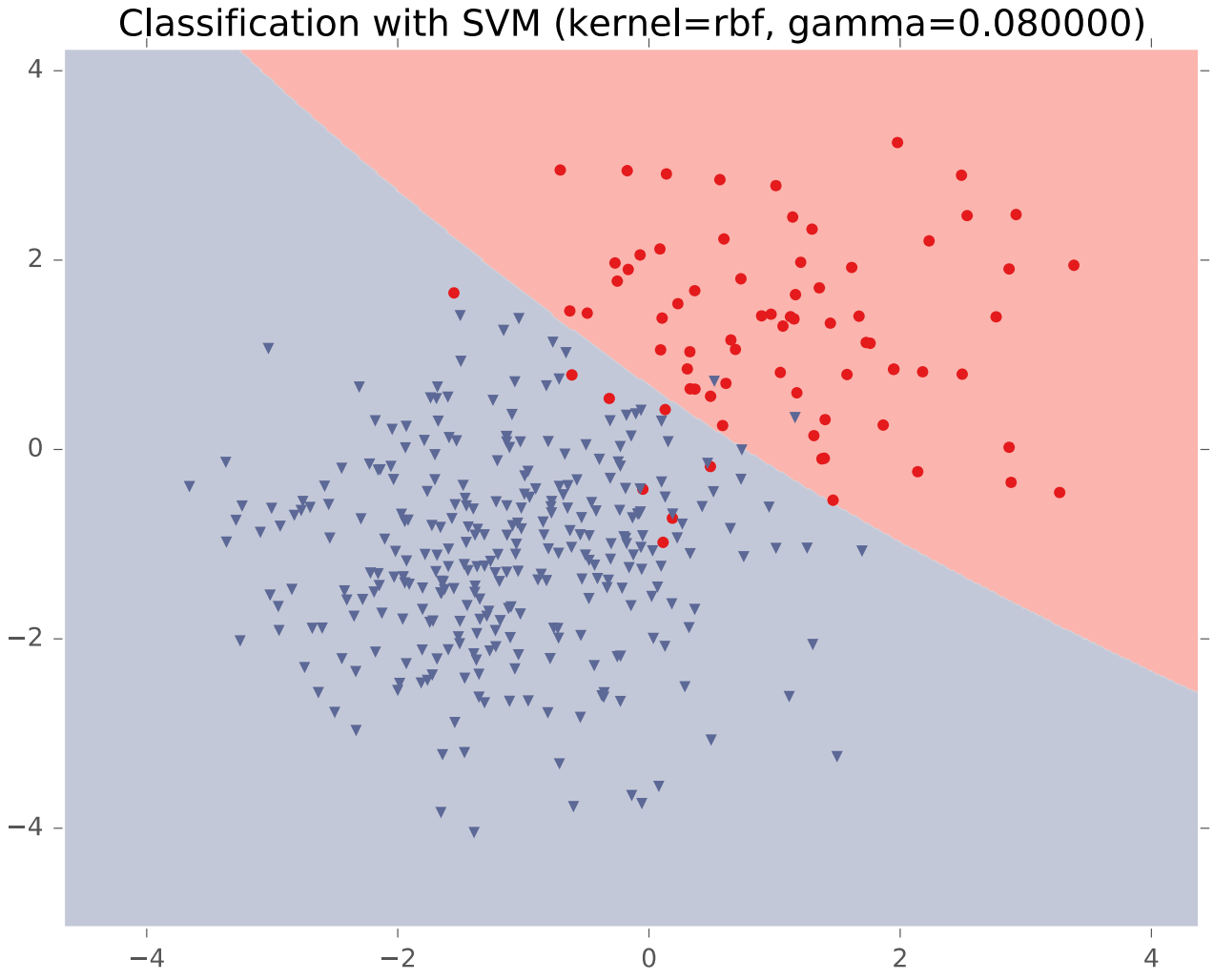
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

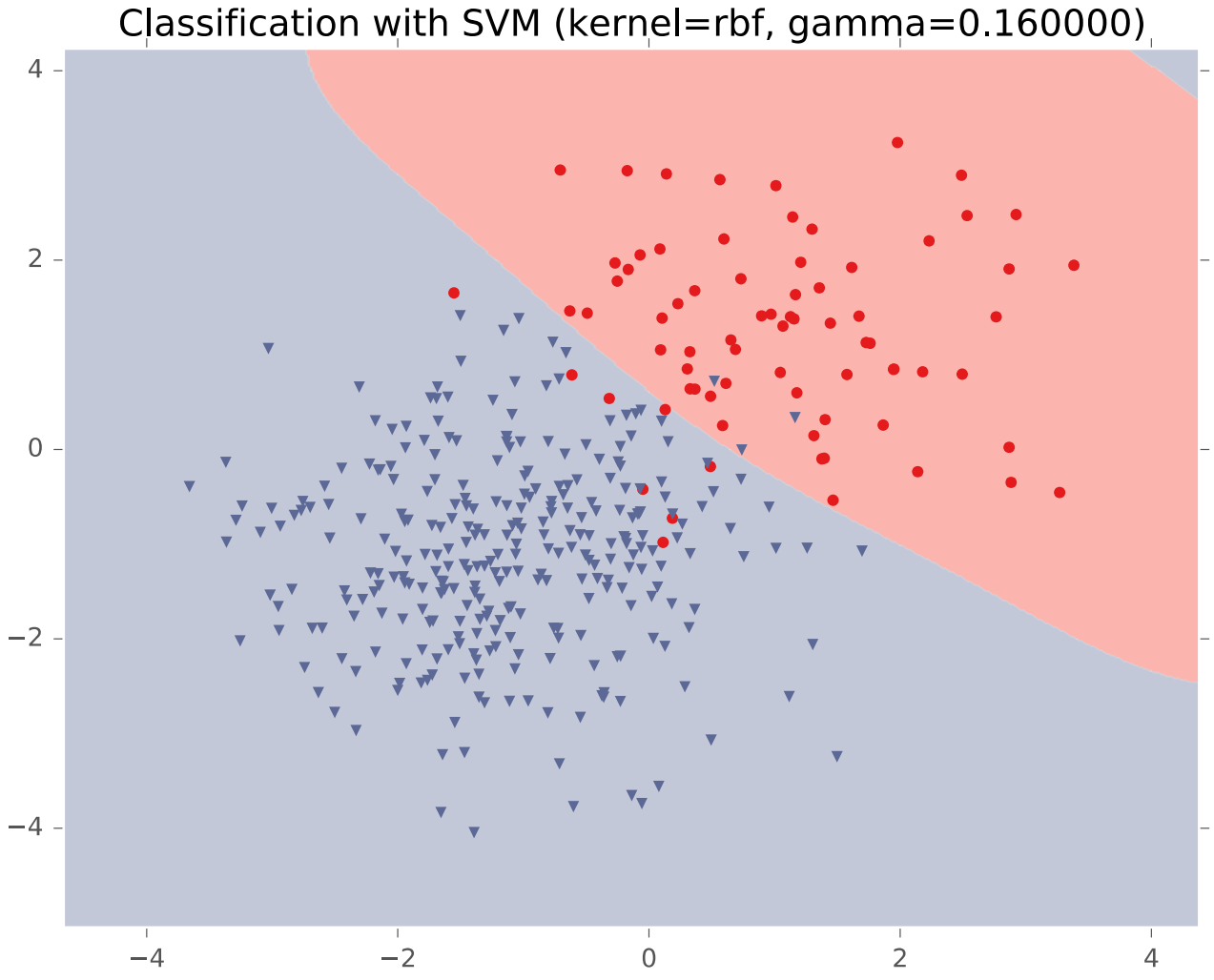
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

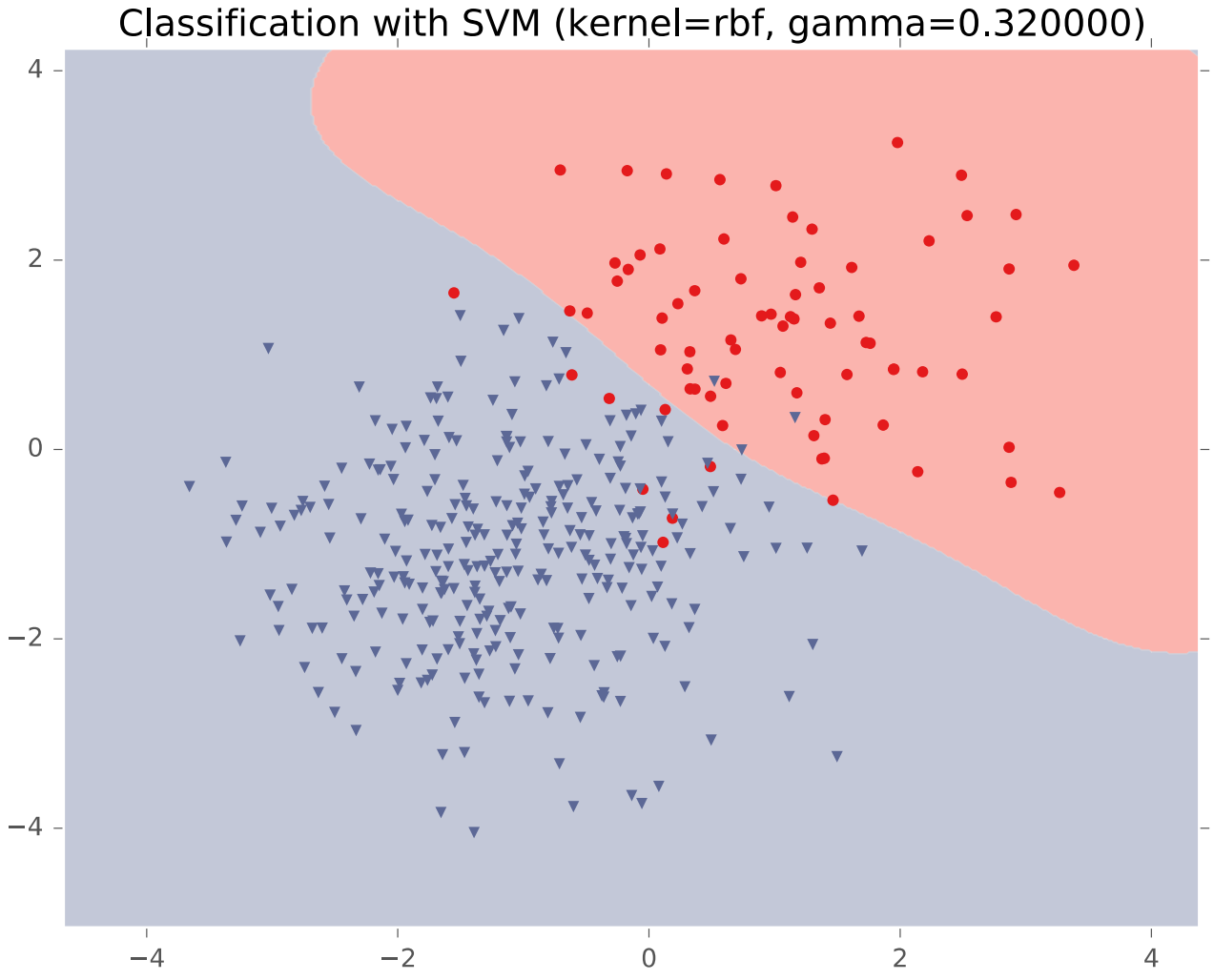
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

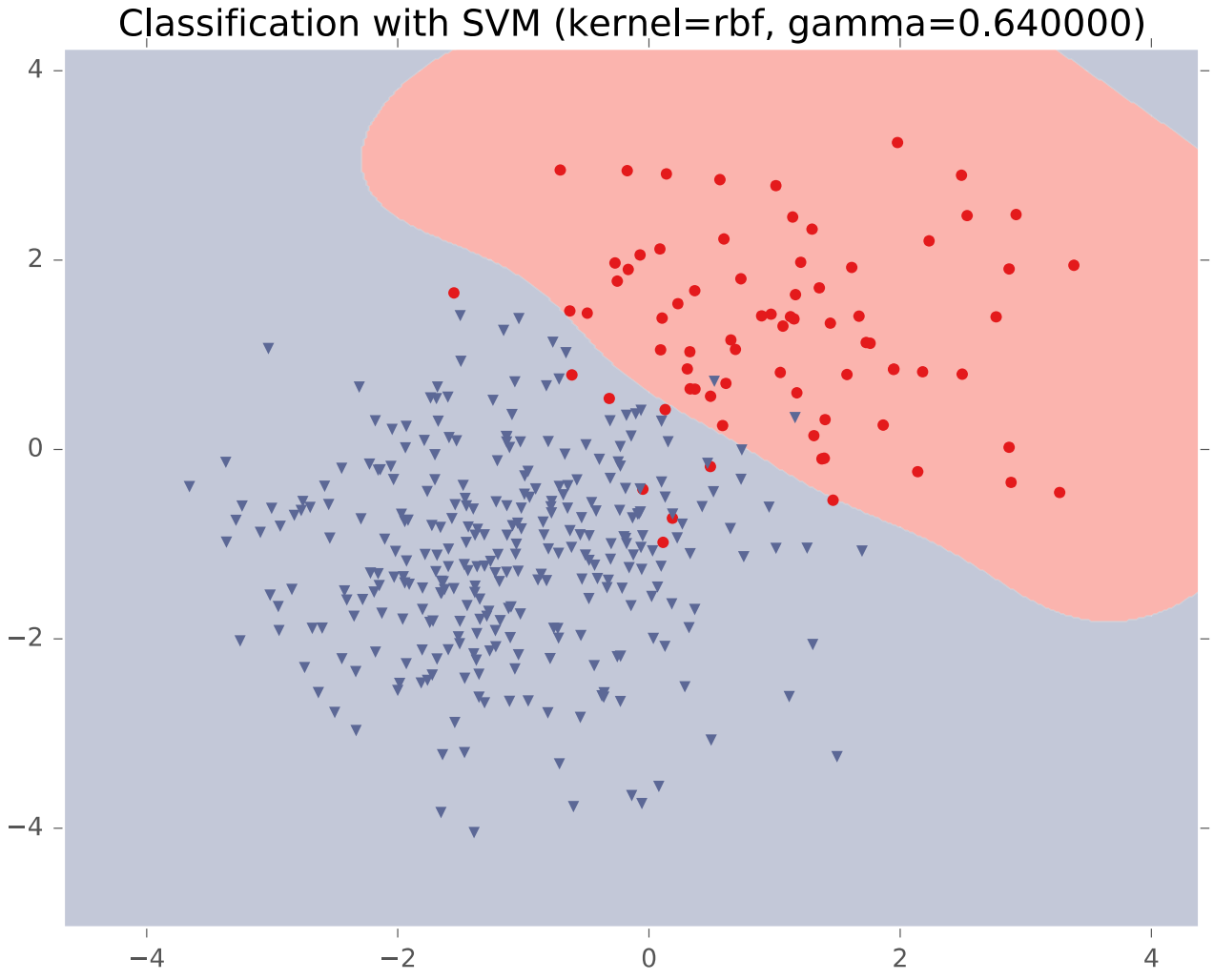
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

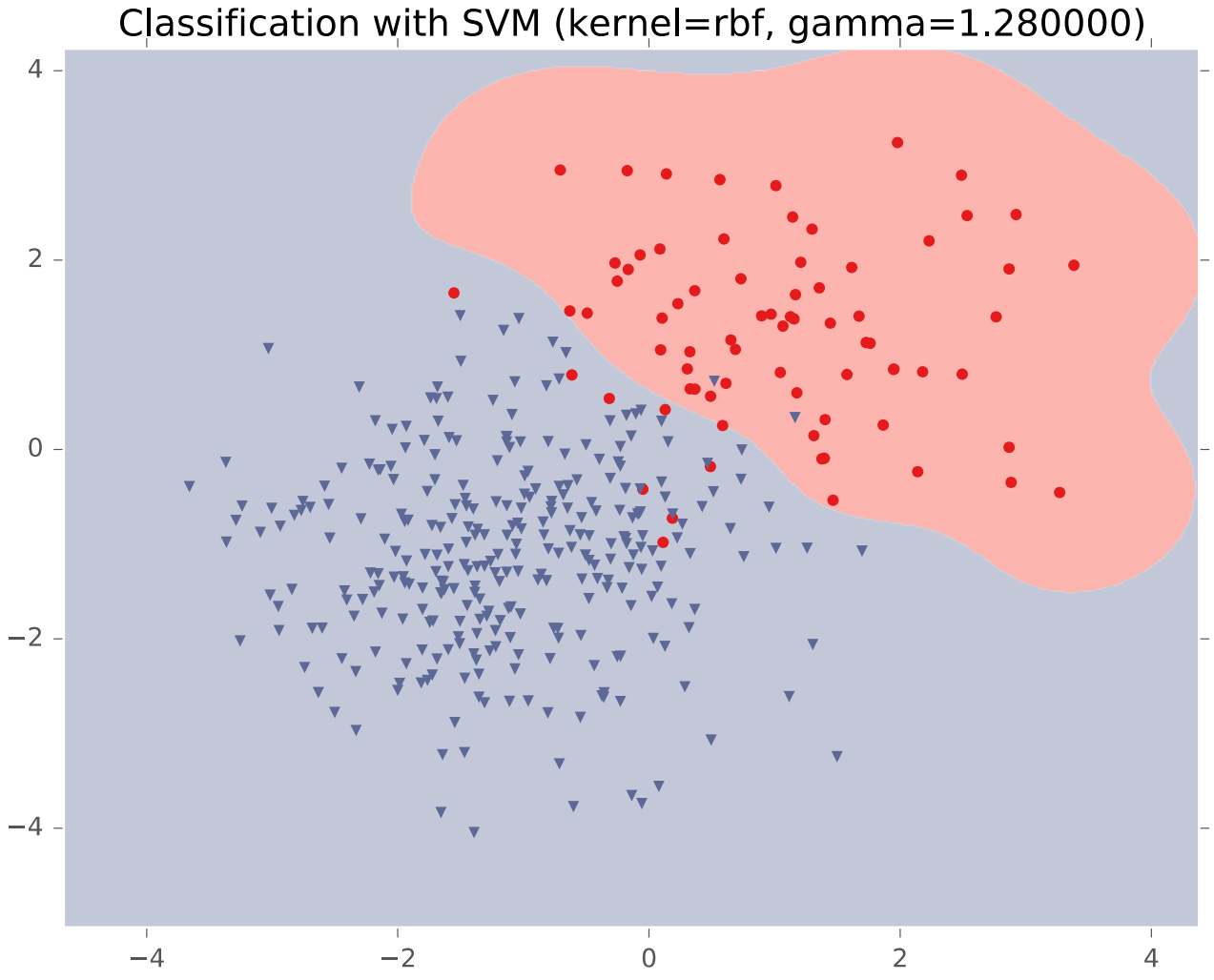
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

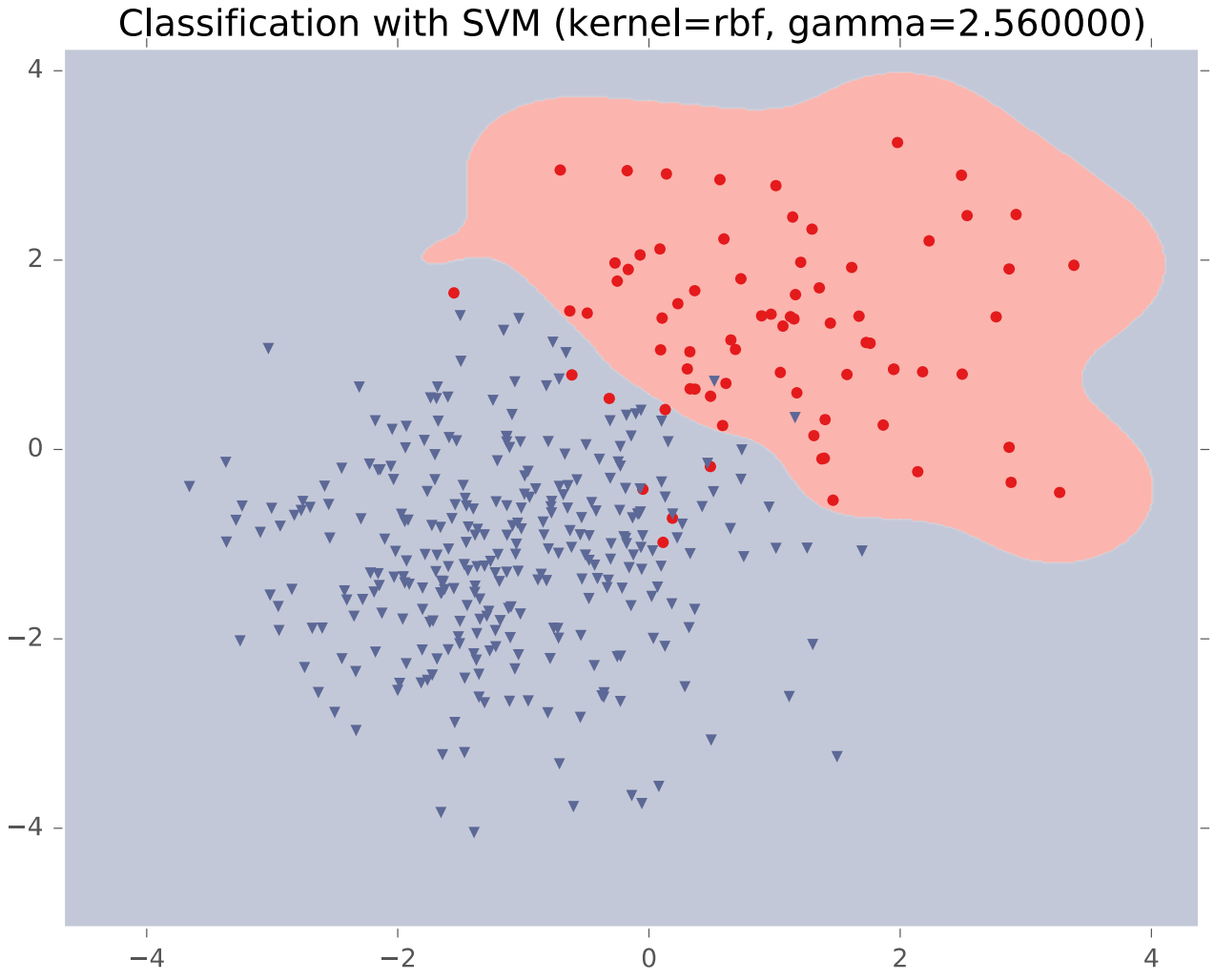
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

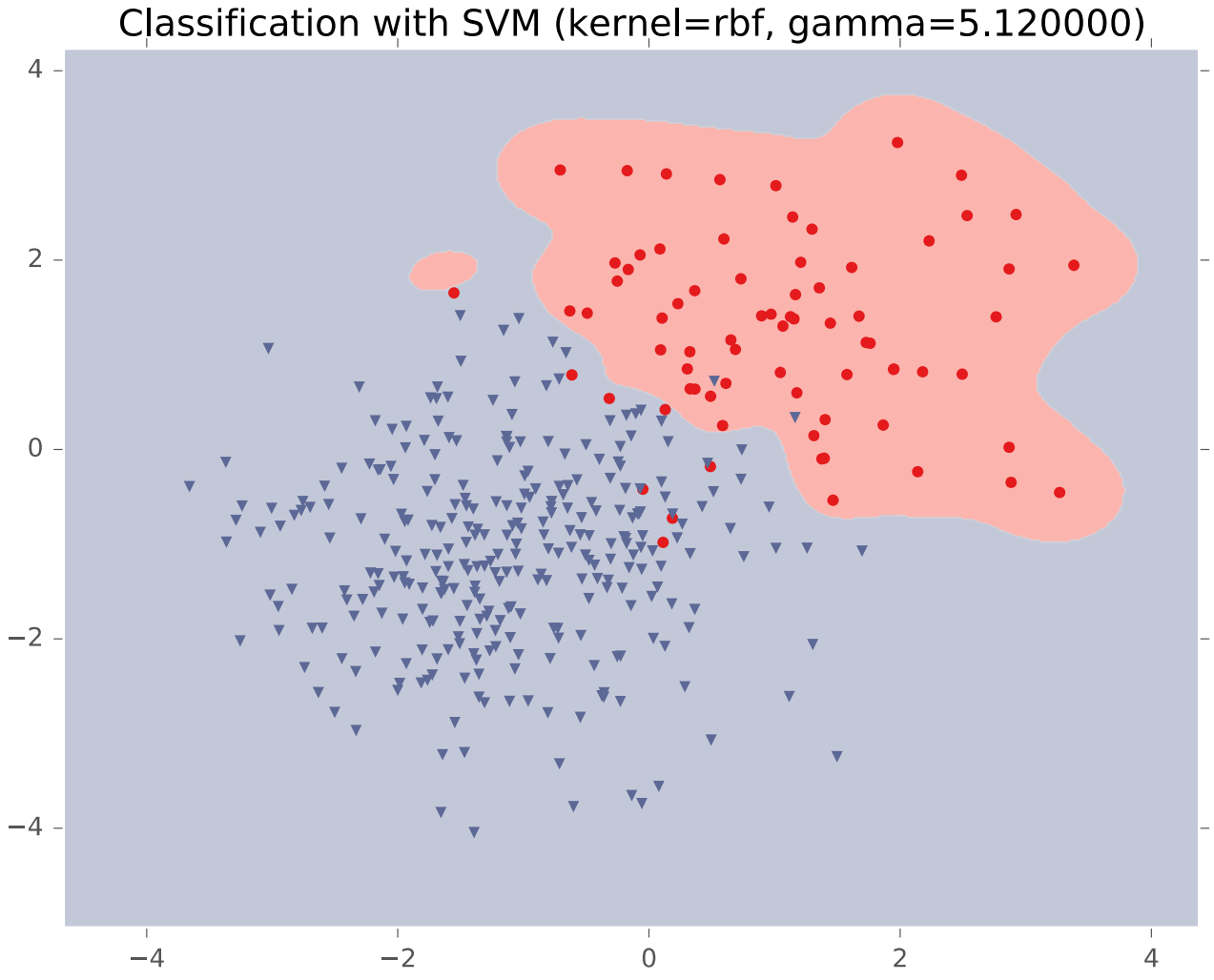
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

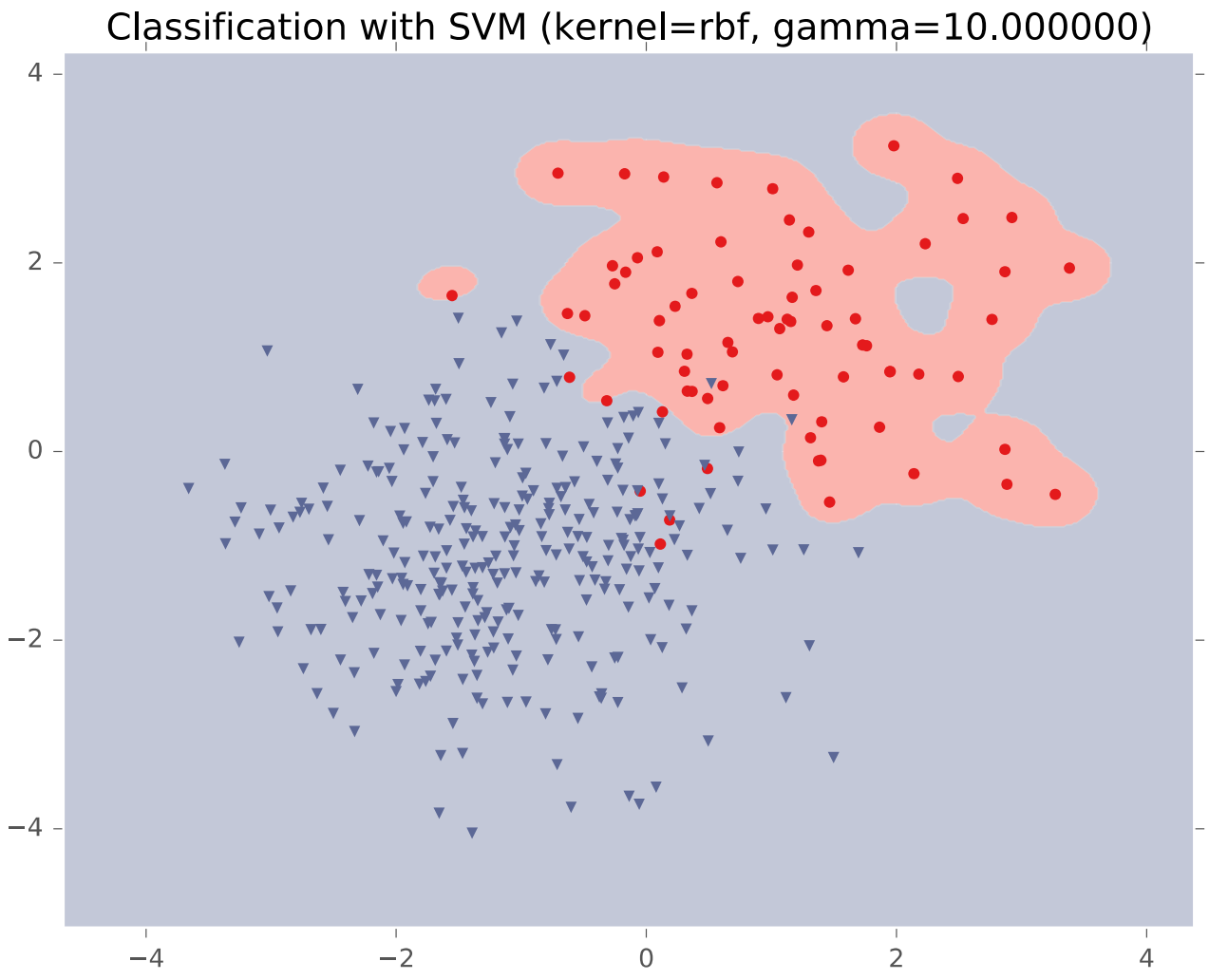
RBF Kernel Example



RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

RBF Kernel Example

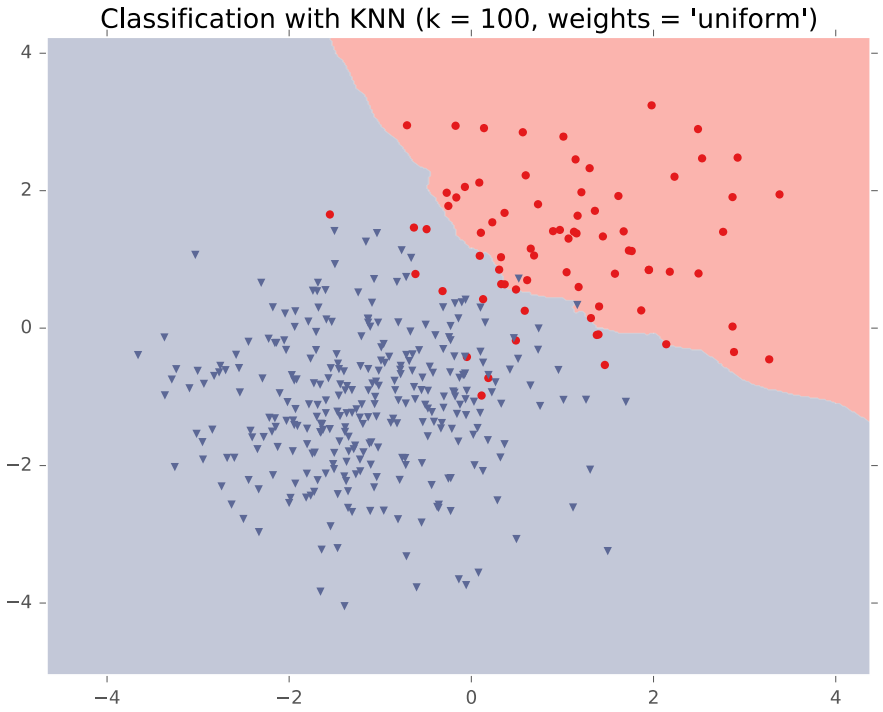


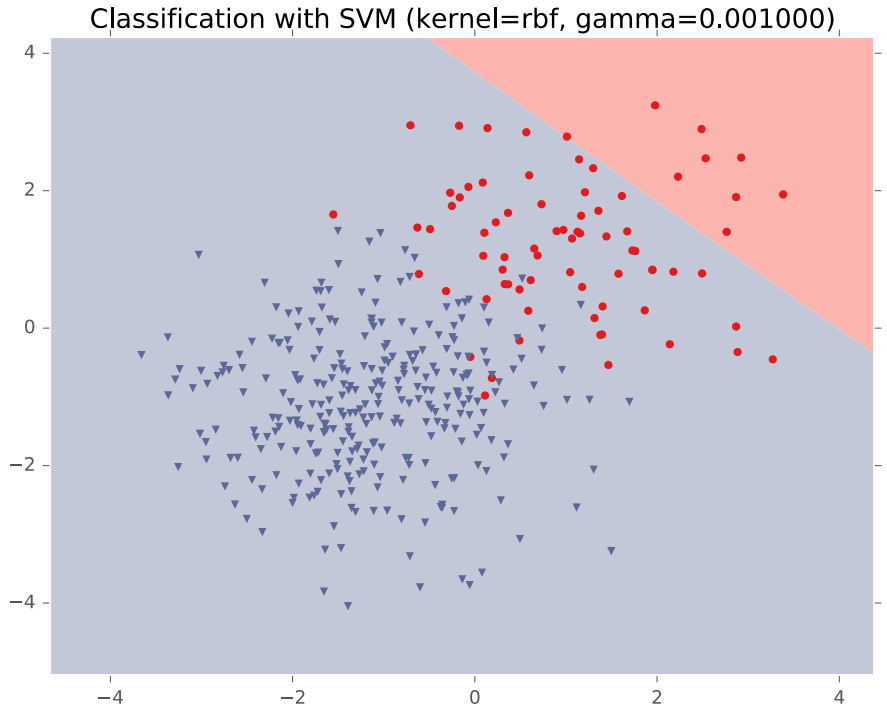
RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

RBF Kernel Example

KNN vs. SVM



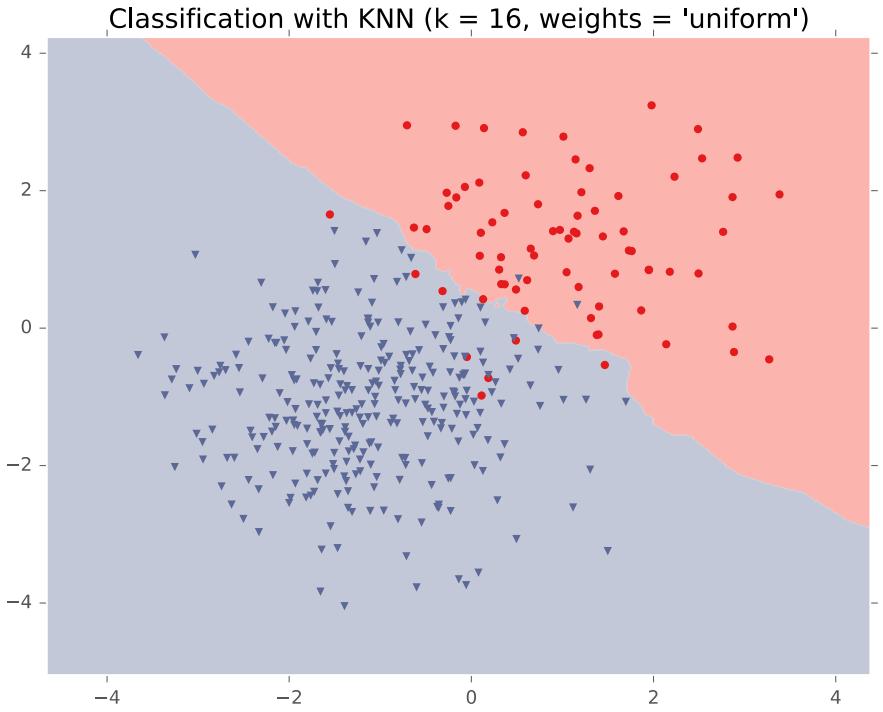


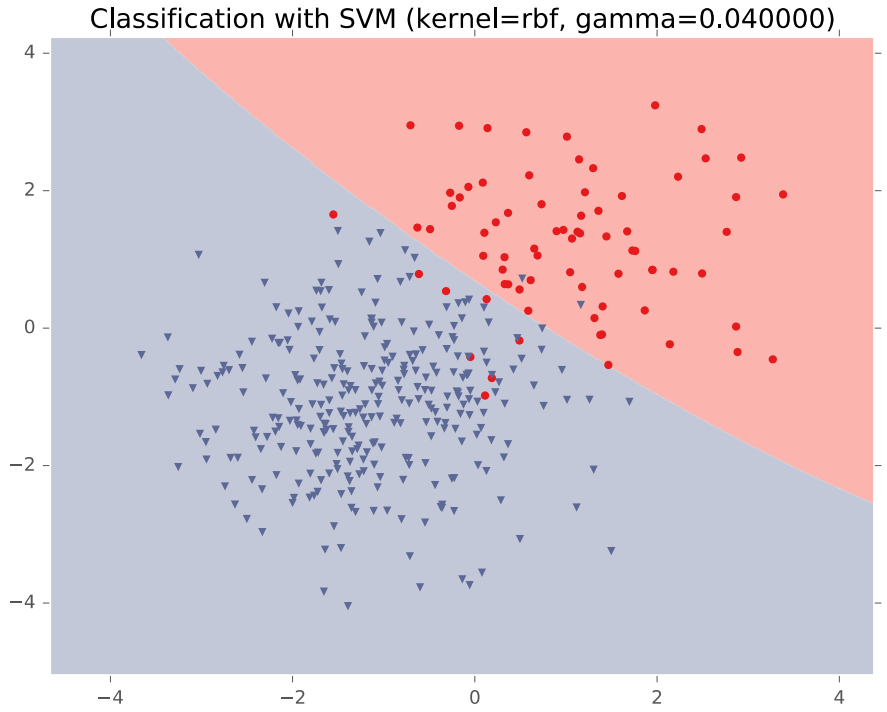
RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

RBF Kernel Example

KNN vs. SVM



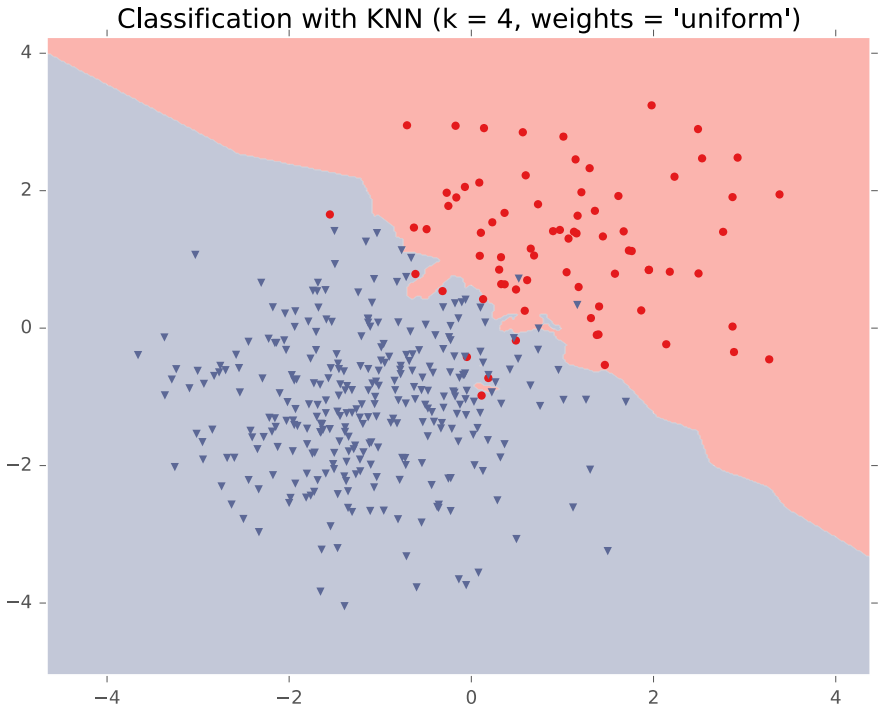


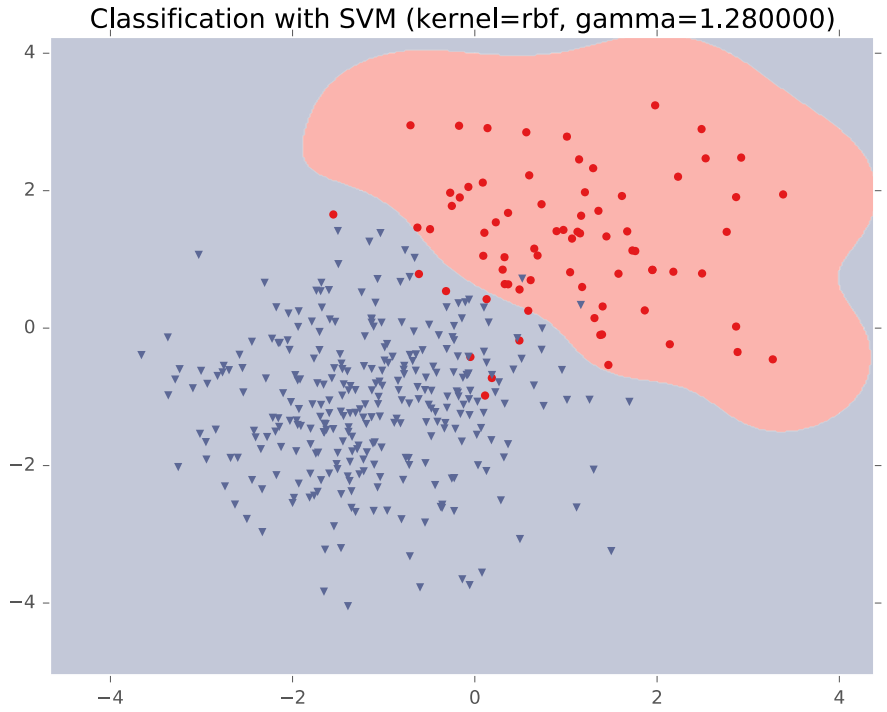
RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

RBF Kernel Example

KNN vs. SVM



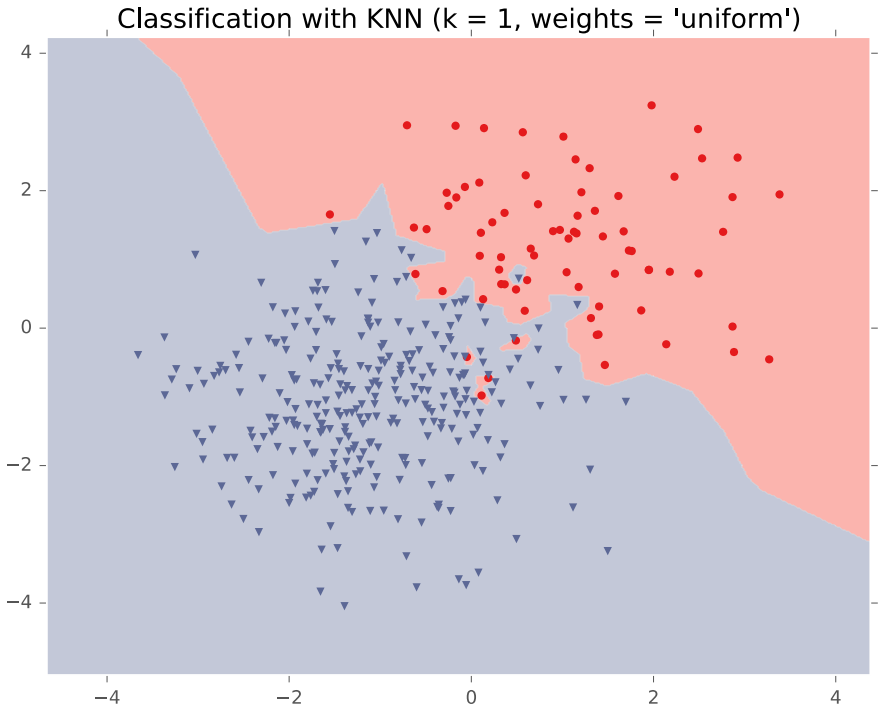


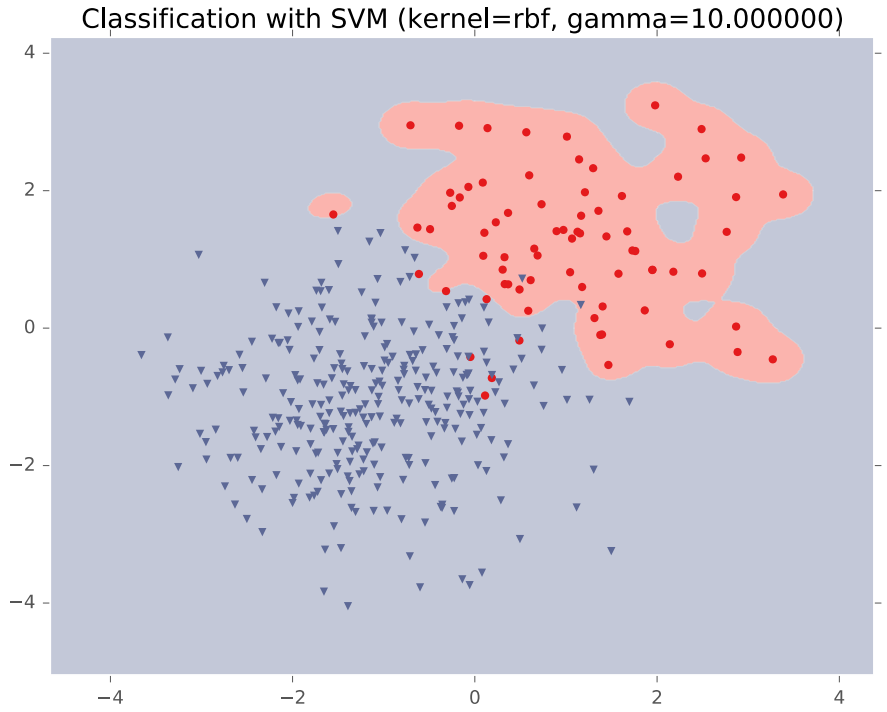
RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

RBF Kernel Example

KNN vs. SVM





RBF Kernel: $K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2^2)$

Slide credit: CMU MLD Matt Gormley

Kernel Methods

- **Key idea:**
  1. Rewrite the algorithm so that we only work with **dot products** $x^T z$ of feature vectors
  2. Replace the **dot products** $x^T z$ with a **kernel function** $k(x, z)$

- The kernel k(x,z) can be **any** legal definition of a dot product:

  $$k(x, z) = \varphi(x)^T \varphi(z) \text{ for any function } \varphi: \mathcal{X} \rightarrow \mathbf{R}^D$$

  So we only compute the φ dot product **implicitly**

- This **"kernel trick"** can be applied to many algorithms:
  - classification: perceptron, SVM, …
  - regression: ridge regression, …
  - clustering: k-means, …

Slide credit: CMU MLD Matt Gormley

SVM + Kernels: Takeaways

- Maximizing the margin of a linear separator is a **good training criteria**
- Support Vector Machines (SVMs) learn a **max-margin linear classifier**
- The SVM optimization problem can be solved with **black-box Quadratic Programming (QP) solvers**
- Learned decision boundary is defined by its **support vectors**
- Kernel methods allow us to work in a transformed feature space **without explicitly representing that space**
- The **kernel-trick** can be applied to **SVMs**, as well as many other algorithms

Slide credit: CMU MLD Matt Gormley

Support Vector Machines

Next steps

- Different optimization formulation
  - Primal → dual
  - “Support vectors”
- Support non-linear classification
  - Feature maps
  - Kernel trick
- Support non-separable data
  - Hard-margin SVM → soft-margin SVM

Support Vector Machines (SVMs)

Hard-margin SVM (Primal)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad \forall i = 1, \ldots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \quad \forall i = 1, \ldots, N$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

- Instead of minimizing the primal, we can maximize the dual problem
- For the SVM, these two problems give the same answer (i.e. the minimum of one is the maximum of the other)
- *Definition*: **support vectors** are those points $x^{(i)}$ for which $\alpha^{(i)} \neq 0$

Slide credit: CMU MLD Matt Gormley

Soft-Margin SVM

Hard-margin SVM (Primal)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad \forall i = 1, \ldots, N$$

Soft-margin SVM (Primal)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\left(\sum_{i=1}^{N} e_i\right)$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - e_i, \quad \forall i = 1, \ldots, N$$

$$e_i \geq 0, \quad \forall i = 1, \ldots, N$$

Slide credit: CMU MLD Matt Gormley

Soft-Margin SVM

Hard-margin SVM (Primal)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \quad \forall i = 1, \ldots, N$$

Hard-margin SVM (Lagrangian Dual)

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$$

$$\text{s.t. } \alpha_i \geq 0, \quad \forall i = 1, \ldots, N$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

Soft-margin SVM (Primal)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\left(\sum_{i=1}^{N} e_i\right)$$

$$\text{s.t. } y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - e_i, \quad \forall i = 1, \ldots, N$$

$$e_i \geq 0, \quad \forall i = 1, \ldots, N$$

Soft-margin SVM (Lagrangian Dual)

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \quad \forall i = 1, \ldots, N$$

$$\sum_{i=1}^{N} \alpha_i y^{(i)} = 0$$

We can also work with the dual of the soft-margin SVM

Slide credit: CMU MLD Matt Gormley