Universität
Konstanz

Lecture 4 Support Vector Machine (SVM) – Part 1

Dr. Hanhe Lin
Dept. of Computer and Information Science
University of Konstanz

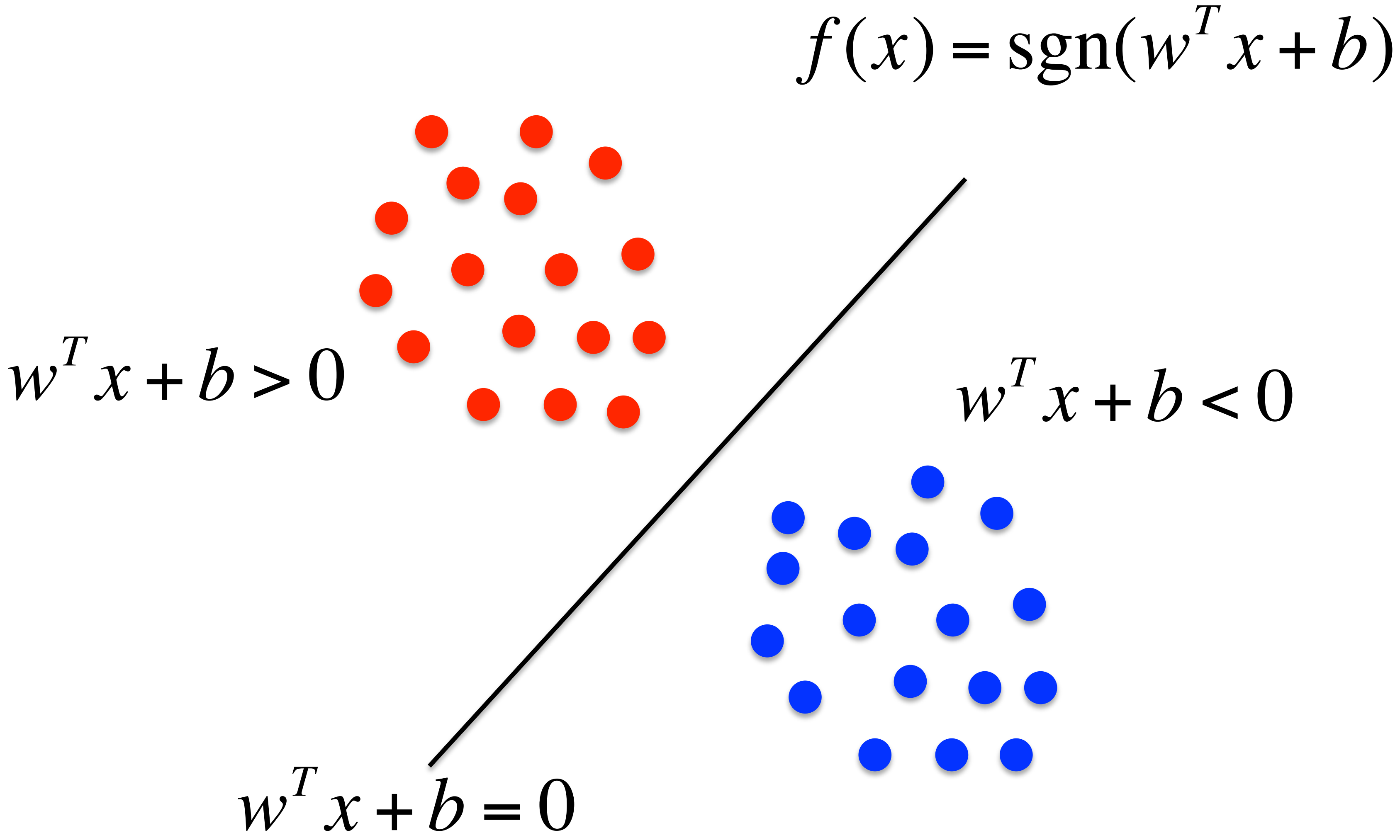
SVM notation

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \Rightarrow \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \Rightarrow \begin{bmatrix} b \\ w \end{bmatrix}$$

$$h_\theta(x) = \theta^T x \Rightarrow f(x) = w^T x + b$$

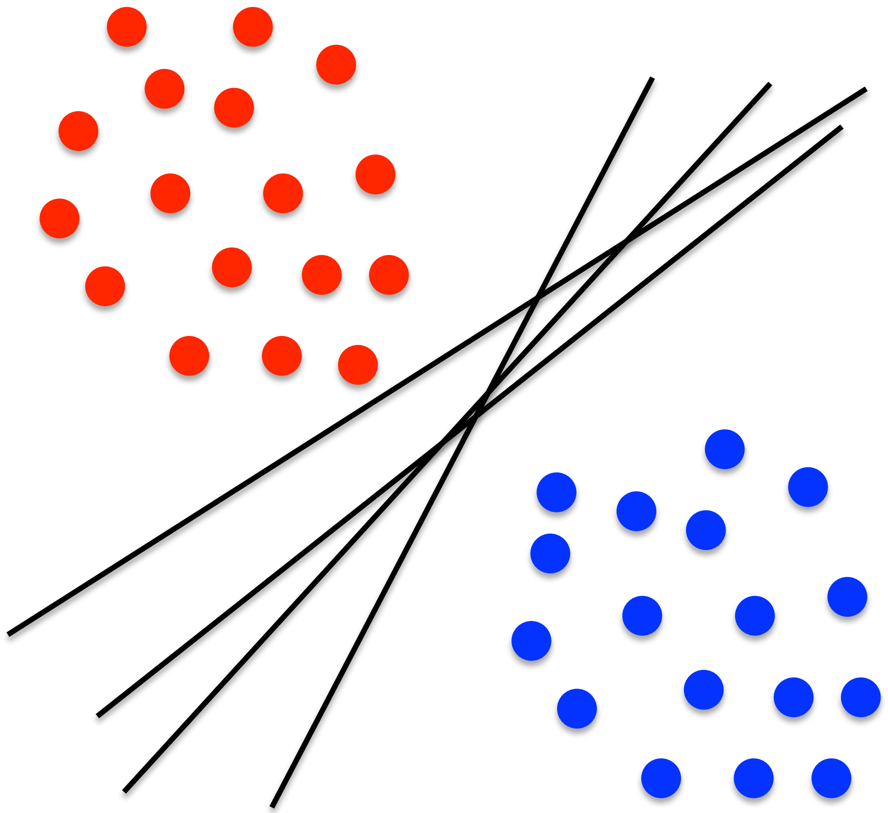
w is weight vector, b is the bias

Intuition – binary classification

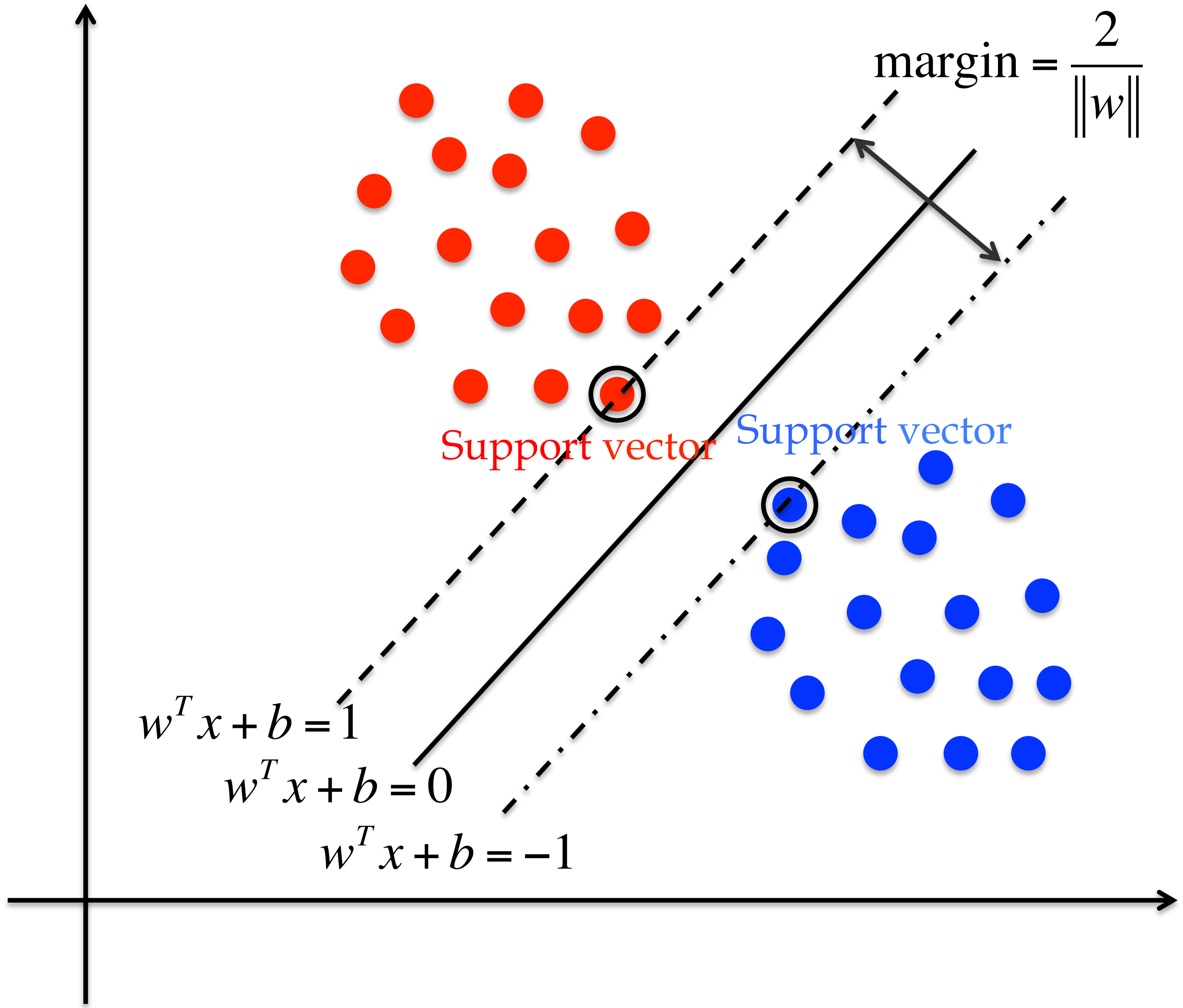


Intuition – binary classification

Which of the linear classifier is optimal?



Intuition – binary classification



Model representation

- Given training examples $(x^{(i)}, y^{(i)})$, SVM aims to find an optimal hyperplane so that:

$$f(x^{(i)}) = w^T x^{(i)} + b \begin{cases} \geq 1 \text{ if } y^{(i)} = 1 \\ \leq -1 \text{ if } y^{(i)} = -1 \end{cases}$$

- Which is equivalent to minimizing the following cost function

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y^{(i)} \underbrace{(w^T x^{(i)} + b)}_{f(x^{(i)})}) + \frac{\lambda}{2}\|w\|^2$$

- Hinge loss: $\max(0, 1 - y^{(i)} f(x^{(i)}))$

$y^{(i)} f(x^{(i)}) \geq 1$ No contribution to loss

$y^{(i)} f(x^{(i)}) < 1$ Contributes to loss

From logistic regression to SVM

Logistic regression

- Label:

$$(x^{(i)}, y^{(i)})\ y^{(i)} \in \{0,1\}$$

- Hypothesis:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

- Objective:

$$h_\theta(x) \begin{cases} \approx 1 \text{ if } y^{(i)} = 1 \\ \approx 0 \text{ if } y^{(i)} = 0 \end{cases}$$

SVM

- Label:

$$(x^{(i)}, y^{(i)})\ y^{(i)} \in \{-1,1\}$$

- Hypothesis:

$$f(x) = w^T x + b$$

- Objective:

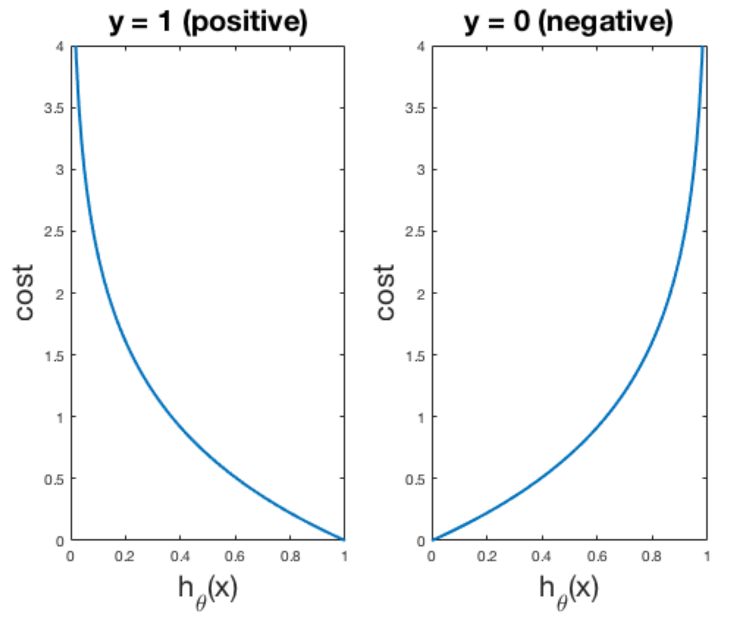
$$f(x^{(i)}) \begin{cases} \geq 1 \text{ if } y^{(i)} = 1 \\ \leq -1 \text{ if } y^{(i)} = -1 \end{cases}$$

From logistic regression to SVM

Logistic regression

- Cost function

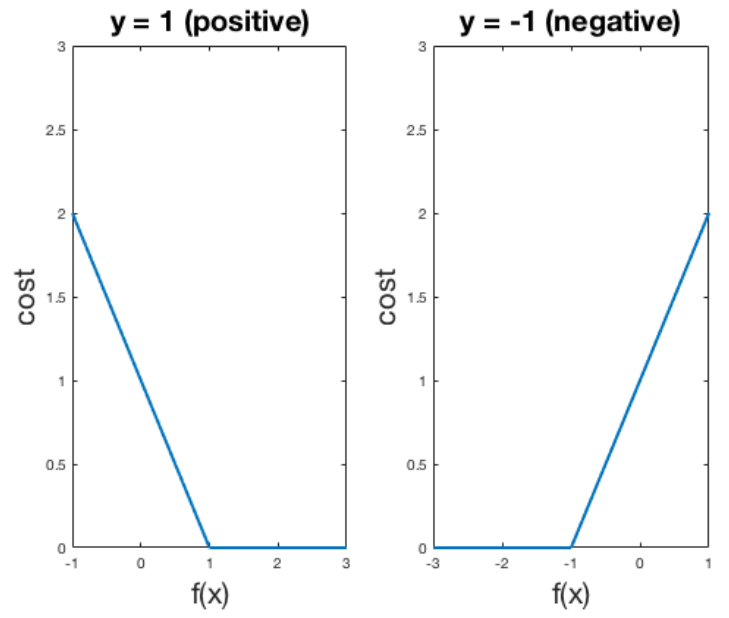
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) \text{ if } y = 1 \\ -\log(1 - h_\theta(x)) \text{ if } y = 0 \end{cases}$$



SVM

- Cost function

$$\mathrm{Cost}(f(x), y) = \begin{cases} \max(0, 1 - f(x)) \text{ if } y = 1 \\ \max(0, 1 + f(x)) \text{ if } y = -1 \end{cases}$$



From logistic regression to SVM

- Logistic regression:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right)$$

- SVM:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\max(0,1-y^{(i)}f(x^{(i)})) + \frac{\lambda}{2}\|w\|^2$$

Gradient computing

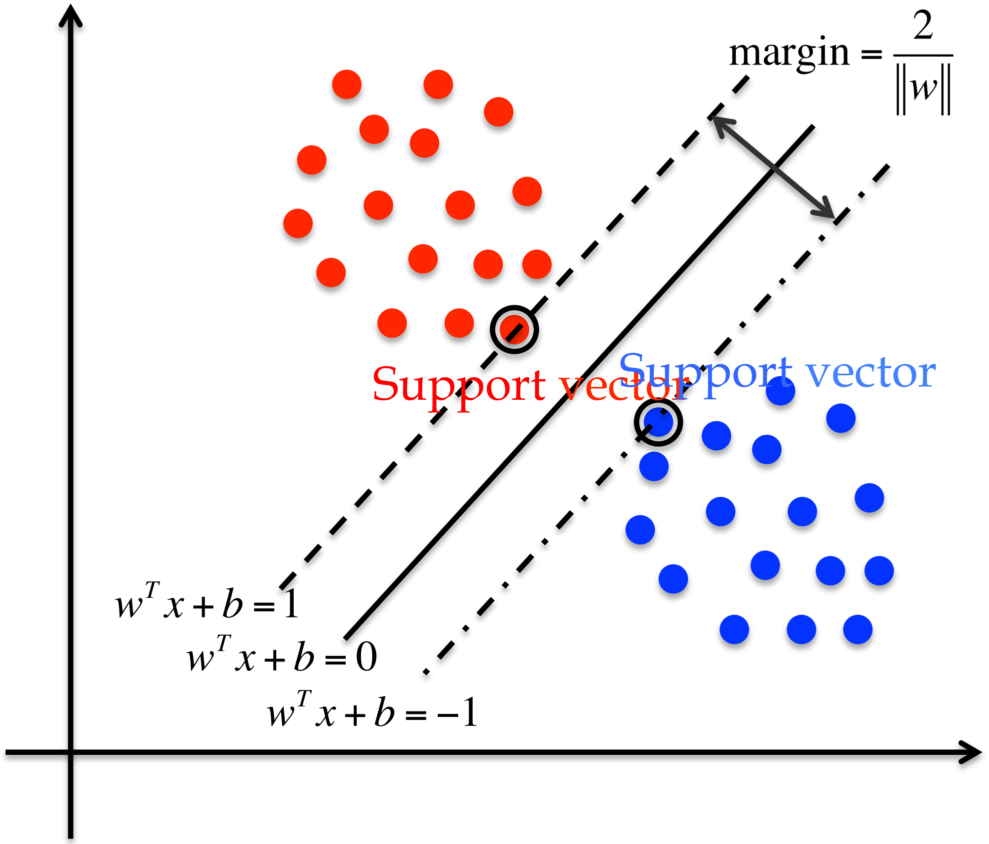
- The hinge loss is not differentiable, a sub-gradient is computed instead:

$$\frac{\partial}{\partial w} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} -y^{(i)} x^{(i)} \text{ if } 1\text{-}y^{(i)} f(x^{(i)}) > 0 \\ 0 \text{ else} \end{cases} + \lambda w$$

$$\frac{\partial}{\partial b} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \begin{cases} 1 \text{ if } 1\text{-}y^{(i)} f(x^{(i)}) > 0 \\ 0 \text{ else} \end{cases}$$

Maximum margin classifier

- Maximizing the margin is good according to tuition and PAC theory
- Implies that only support vectors are important; while other training examples are ignorable



“Hard” margin

- The maximum margin can be optimized as:

$$\max_{w,b} \frac{2}{\|w\|}$$

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

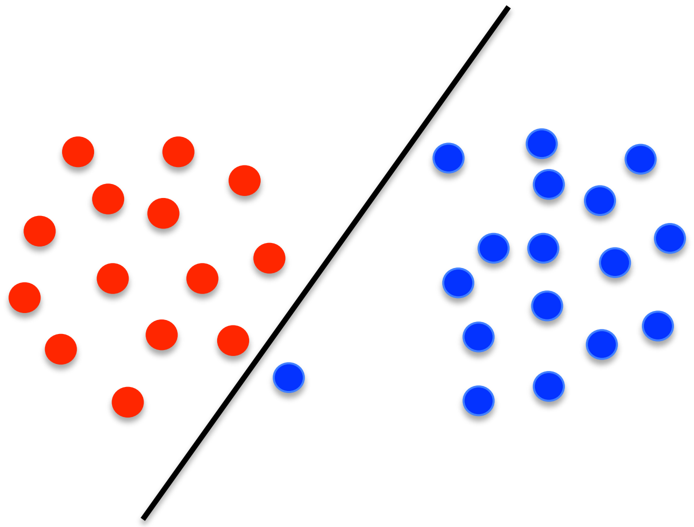
- Or equivalently:

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

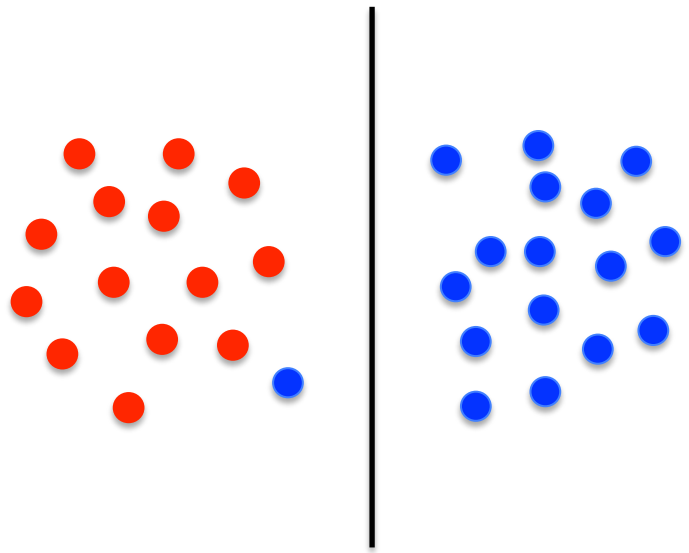
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

Which classifier is better?



The points can be linearly separated but there is a very narrow margin



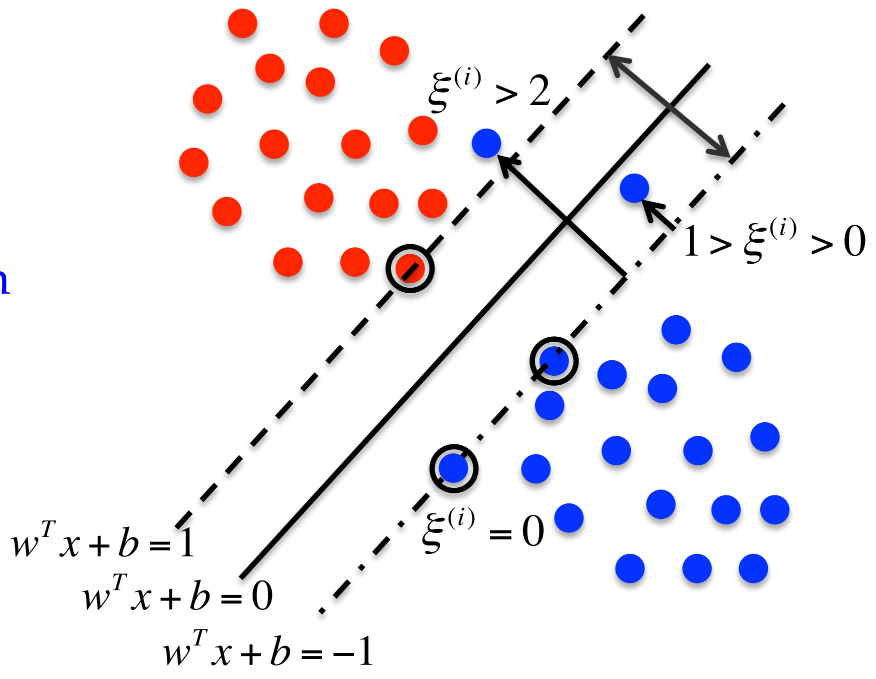
The large margin solution is better, even though one constraint is violated

Tradeoff between the margin and the number of mistakes in the training data

Introduce "slack" variables

$\xi^{(i)} \geq 0$

- For $1 > \xi^{(i)} > 0$ point is between margin and correct side of hyperplane. This is margin violation
- For $\xi^{(i)} > 1$ point is misclassified



"Soft" margin solution

The optimization problem becomes:

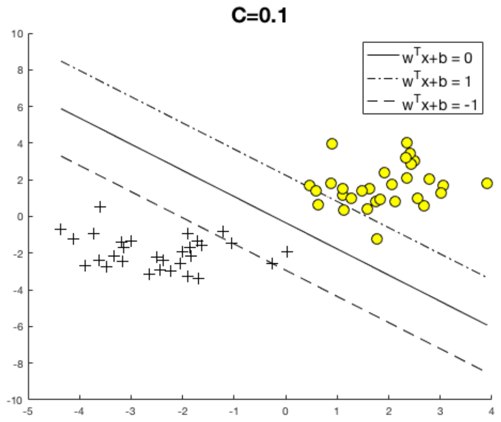
$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi^{(i)}$$

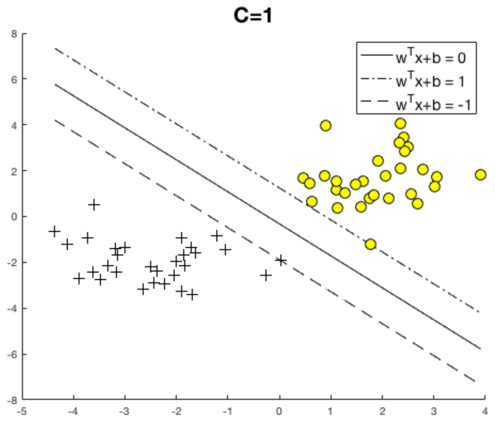
$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0 \quad \forall i = 1,\ldots,m$$

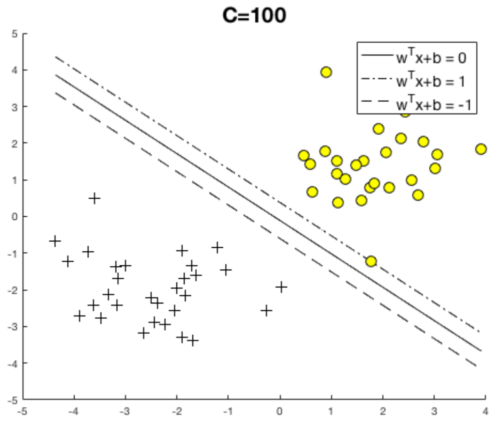
- Every constraint can be satisfied if $\xi^{(i)}$ is sufficiently large
- C is a regularization parameter:
  - Small C $\Rightarrow$ large margin
  - Large C $\Rightarrow$ narrow margin
  - C = ∞ $\Rightarrow$ hard margin
- It is called primal form of SVM

Still a quadratic optimization problem and there is a unique minimum

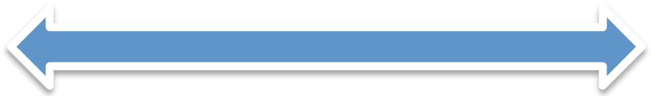
Example: different regularization







Soft margin



Hard margin

Primal form

The optimization problem becomes:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi^{(i)}$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0 \quad \forall i = 1,\ldots,m$$

Can solve it more efficiently by taking the Lagrangian dual

- Duality is a common idea in optimization
- It transforms a difficult optimization problem into a simpler one

Optimization: Lagrange multiplier

- By introducing Lagrange multipliers $\alpha^{(i)}, \beta^{(i)} \geq 0$, the corresponding Lagrangian is formulated as:

$$L = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi^{(i)} - \sum_{i=1}^{m}\alpha^{(i)}(y^{(i)}(w^T x^{(i)} + b) - 1 + \xi^{(i)}) - \sum_{i=1}^{m}\beta^{(i)}\xi^{(i)}$$

- To minimize it, its derivatives w.r.t to the variables are set to zeros:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m}\alpha^{(i)}y^{(i)}x^{(i)} = 0 \Rightarrow w = \sum_{i=1}^{m}\alpha^{(i)}y^{(i)}x^{(i)}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m}\alpha^{(i)}y^{(i)} = 0$$

$$\frac{\partial L}{\partial \xi^{(i)}} = C - \alpha^{(i)} - \beta^{(i)} = 0 \qquad C \geq \alpha^{(i)} \geq 0$$

Dual form

- Solve by a bunch of algebra and calculus, the dual form of SVM is:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} (x^{(i)})^T x^{(j)} - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\text{s.t. } 0 \le \alpha^{(i)} \le C, \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \;\; \forall i = 1, \ldots, m$$

- The decision can also be rewritten as:

$$w = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} x^{(i)}$$

$$f(x) = w^T x + b \Rightarrow f(x) = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} (x^{(i)})^T x + b$$

Understanding the dual

- Solve by a bunch of algebra and calculus, the dual form of SVM is:

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}(x^{(i)})^{T}x^{(j)} - \sum_{i=1}^{m}\alpha^{(i)}$$

$$\text{s.t.}\ \boxed{0 \le \alpha^{(i)} \le C},\ \boxed{\sum_{i=1}^{m}\alpha^{(i)}y^{(i)} = 0}\ \forall i = 1,\ldots,m$$

Constraint weights between 0 and C

Balances between the weight of constraints for different classes

Understanding the dual

- Solve by a bunch of algebra and calculus, the dual form of SVM is:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} \boxed{(x^{(i)})^T x^{(j)}} - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\text{s.t.} \boxed{0 \leq \alpha^{(i)} \leq C} \quad \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \quad \forall i = 1, \ldots, m$$

Points with different labels decrease the sum
Points with same labels increase the sum

Measures the similarity between points

Understanding the dual

- Solve by a bunch of algebra and calculus, the dual form of SVM is:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} (x^{(i)})^T x^{(j)} - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\text{s.t. } 0 \le \alpha^{(i)} \le C, \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \ \forall i = 1, \ldots, m$$

$C \ge \alpha^{(i)} > 0$ point is a support vector

$\alpha^{(i)} = 0$ point is not a support vector

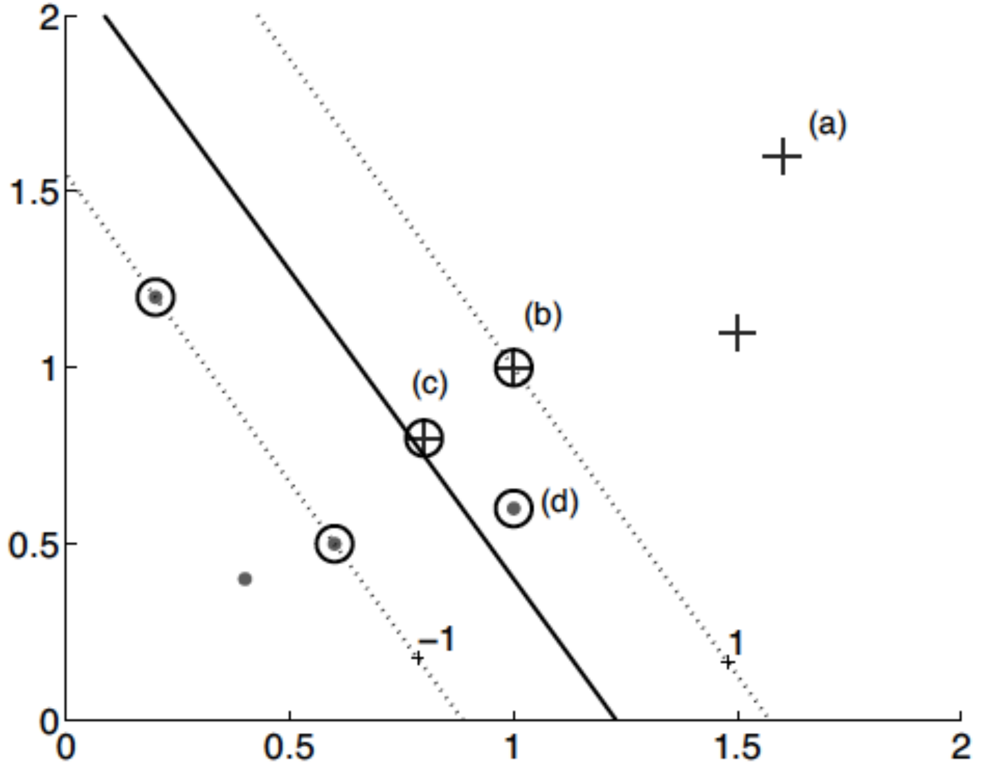
Q: support vectors

$\alpha^{(a)} = ?$

$\alpha^{(b)} = ?$

$\alpha^{(c)} = ?$

$\alpha^{(d)} = ?$



$\alpha^{(a)} = 0$

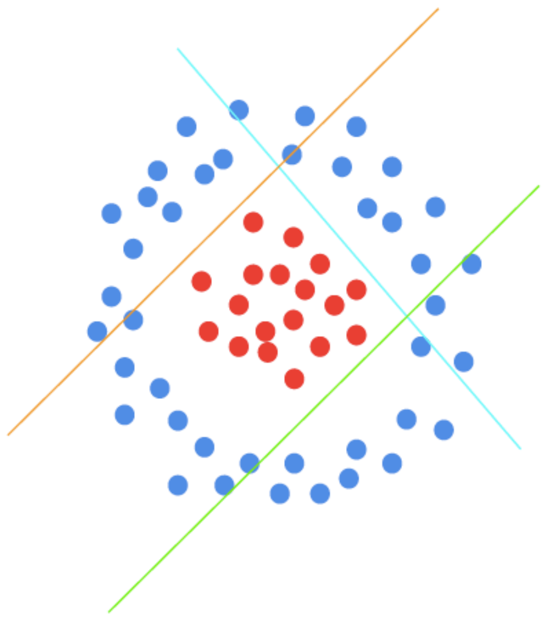
$C > \alpha^{(b)} > 0$

$\alpha^{(c)} = \alpha^{(d)} = C$

Prediction is very fast as most α are zeros

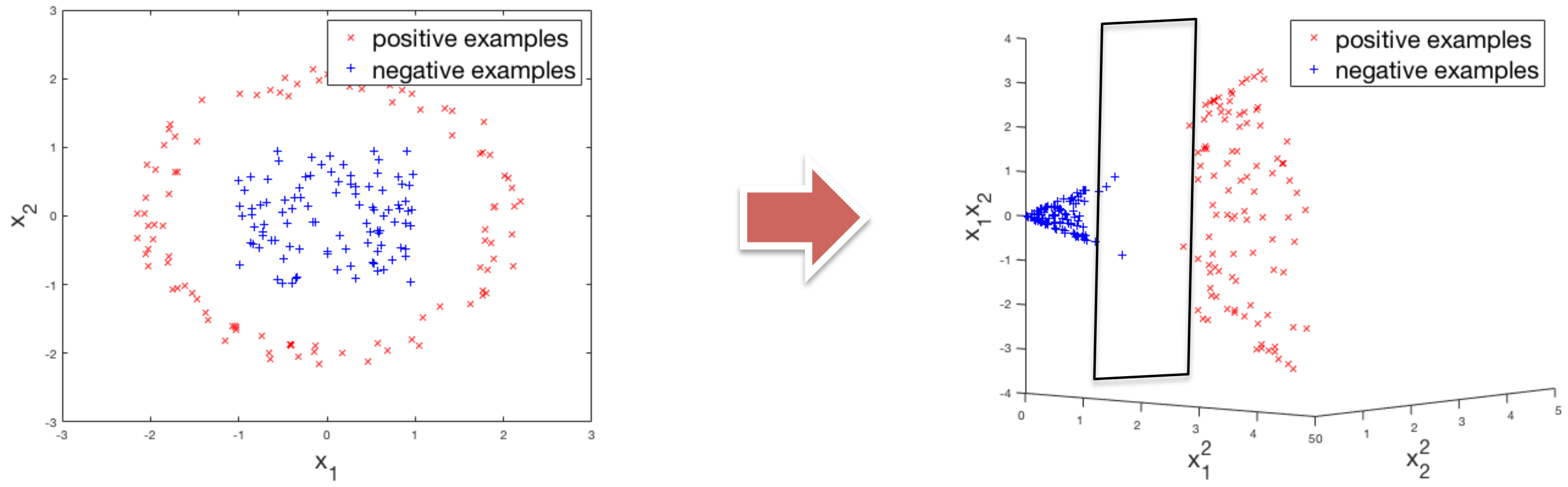
What if the data is not linearly separable?

- What's the advantage of SVM over logistic regression?
- In logistic regression, we add more parameters to make the decision boundary nonlinearly
- Expensive to compute and store when parameters are very large



Mapping data into a higher dimensional feature space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \end{pmatrix}$$



Feature map

- By mapping data from n-dimensional to N-dimensional space (n<N), we can still learn a linear classifier
- Feature map: $\Phi : x \rightarrow \Phi(x)$
- What if $\Phi(x)$ is really big?

  A: use kernels to compute it implicitly

Kernels: motivation

- Inefficient features:
  - Non-linearly separable data requires high dimensional representation
  - Might prohibitively expensive to compute or store

Kernel representation

- Kernel computing:

$$k(x^{(i)}, x^{(j)}) = \Phi(x^{(i)})^T \Phi(x^{(j)})$$

- Computing kernel should be efficient, much more than computing feature maps separately

Transformed feature in primal form

Classifier: $f(x) = w^T \Phi(x) + b$

Optimization:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi^{(i)}$$

$$\text{s.t. } y^{(i)}(w^T \Phi(x^{(i)}) + b) \geq 1 - \xi^{(i)}, \xi^{(i)} \geq 0 \quad \forall i = 1, \ldots, m$$

- Simply map x to $\Phi(x)$ where data is separable
- Solve for w in high dimensional feature space
- Still troublesome when $\Phi(x)$ is very large

Transformed feature in dual form

Classifier: $f(x) = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} \Phi(x^{(i)})^T \Phi(x) + b$

Optimization:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} \Phi(x^{(i)})^T \Phi(x^{(j)}) - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\text{s.t. } 0 \le \alpha^{(i)} \le C, \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \;\; \forall i = 1, \ldots, m$$

- In dual form, $\Phi(x)$ only occurs in pairs
- Only the m dimensional vector $\alpha^{(i)}$ needs to be learnt

Kernel SVM in dual form

Classifier:

$$f(x) = \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} k(x^{(i)}, x) + b$$

Optimization:

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} k(x^{(i)}, x^{(j)}) - \sum_{i=1}^{m} \alpha^{(i)}$$

$$\text{s.t. } 0 \le \alpha^{(i)} \le C, \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \ \forall i = 1, \ldots, m$$

Solve by quadratic programming!

The kernel trick

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

$$\Phi(x)^T \Phi(z) = \begin{pmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1x_2 \end{pmatrix} \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{pmatrix} = (x_1z_1 + x_2z_2)^2 = (x^T z)^2$$

- Classifier can be learnt and applied without explicitly computing
- All that is required to compute the kernel function $k(x^{(i)}, x^{(j)})$
- Complexity of learning depends on number of training examples *m* rather than the dimensions of feature space *N*.

The kernel trick

- "Given an algorithm which is formulated in terms of a positive definite kernel K_1, one can construct an alternative algorithm by replacing K_1 with another positive kernel K_2"

Common kernels

- Linear kernel: $k(x^{(i)}, x^{(j)}) = (x^{(i)})^T x^{(j)}$
- Polynomial kernel:

$$k(x^{(i)}, x^{(j)}) = (1 + (x^{(i)})^T x^{(j)})^d$$

Contains all polynomials terms up to degree *d*

- Gaussian kernel: $k(x^{(i)}, x^{(j)}) = \exp(-\frac{\left\| x^{(i)} - x^{(j)} \right\|^2}{2\sigma^2})$

Infinite dimensional feature space

Many more…
- Cosine similarity kernel
- Chi-squared kernel
- String/tree/graph kernel

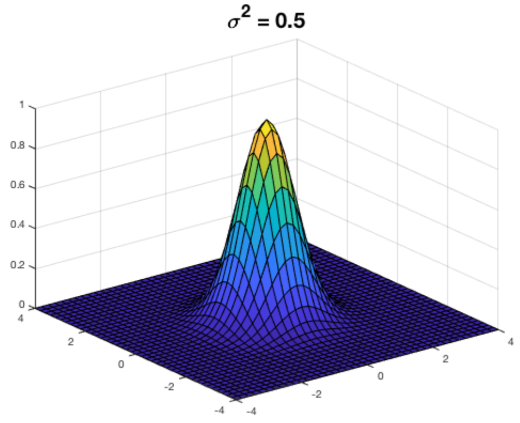
Q: how many hyper-parameters do you need to tune?

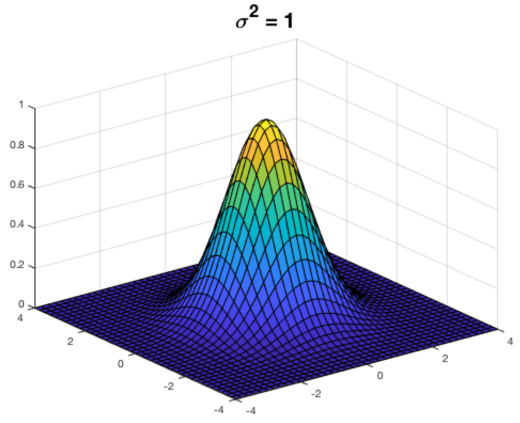
Gaussian kernel

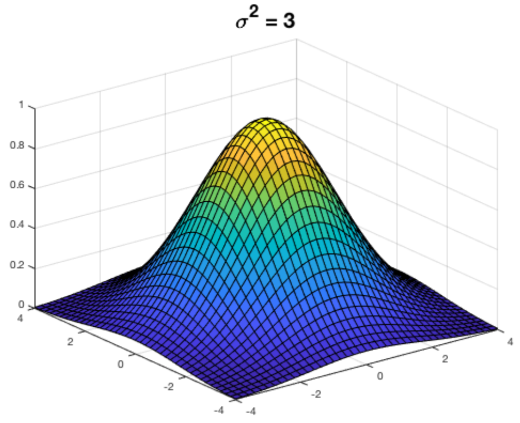
- Also called Radial Basis Function (RBF) kernel

$$k(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^2}{2\sigma^2}\right)$$

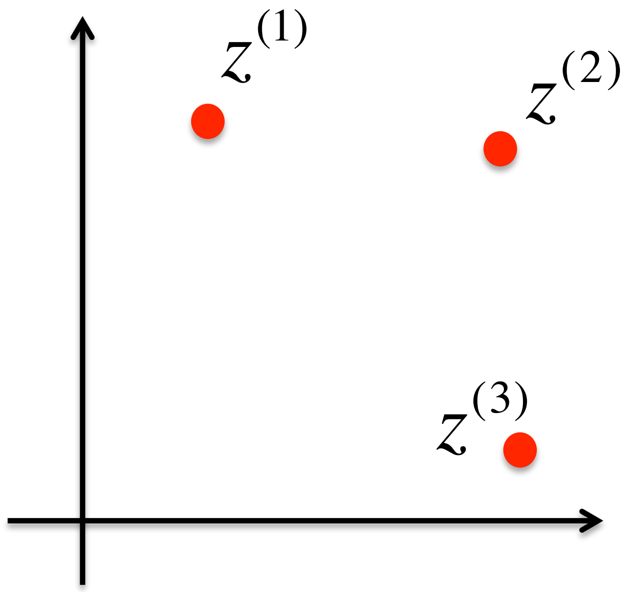
  - Has value 1 when $x^{(i)} = x^{(j)}$
  - Value falls off to 0 with increasing distance
  - Note: need to do feature normalization before using Gaussian kernel







Gaussian kernel example



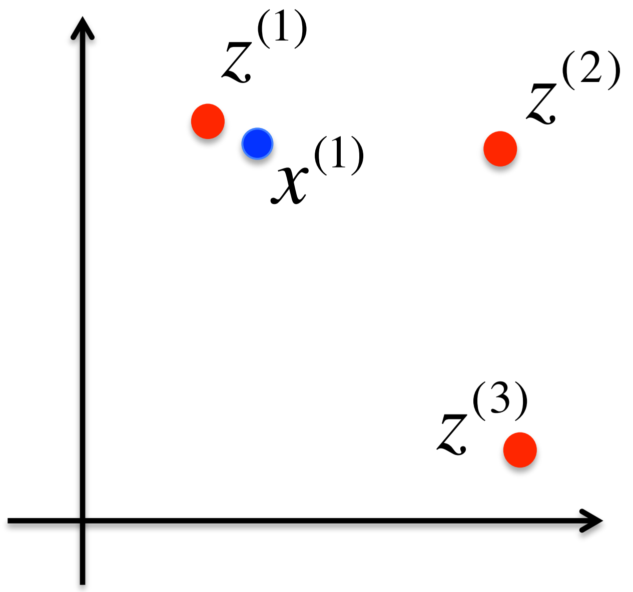
$$k(x^{(i)}, x^{(j)}) = \exp(-\frac{\left\| x^{(i)} - x^{(j)} \right\|^2}{2\sigma^2})$$

Image we've learned that:

$$\alpha = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad b = -0.5$$

Predict $+1$ if $\alpha_1 k(x, z^{(1)}) + \alpha_2 k(x, z^{(2)}) + \alpha_3 k(x, z^{(3)}) + b \geq 0$

Gaussian kernel example



$$k(x^{(i)},x^{(j)}) = \exp\left(-\frac{\left\|x^{(i)}-x^{(j)}\right\|^2}{2\sigma^2}\right)$$

Image we've learned that:

$$\alpha = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad b = -0.5$$

Predict $+1$ if $\alpha_1 k(x,z^{(1)}) + \alpha_2 k(x,z^{(2)}) + \alpha_3 k(x,z^{(3)}) + b \geq 0$

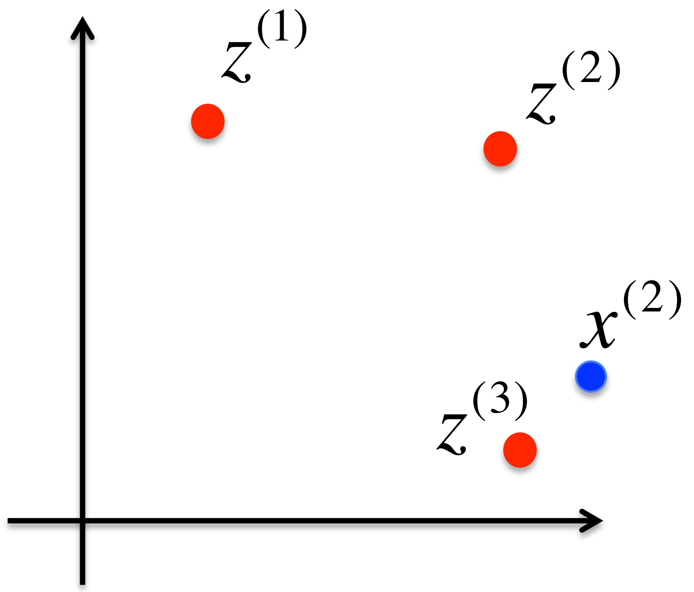
For $x^{(1)}$, we have $k(x^{(1)},z^{(1)}) \approx 1,\ k(x^{(1)},z^{(2)}) \approx 0,\ k(x^{(1)},z^{(3)}) \approx 0$

$\alpha_1(1) + \alpha_2(0) + \alpha_3(0) + b$

$= 1(1) + 1(0) + 0(0) - 0.5$

$= 0.5$, so predict $+1$

Gaussian kernel example



$$k(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\left\| x^{(i)} - x^{(j)} \right\|^2}{2\sigma^2}\right)$$

Image we've learned that:

$$\alpha = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad b = -0.5$$

Predict $+1$ if $\alpha_1 k(x, z^{(1)}) + \alpha_2 k(x, z^{(2)}) + \alpha_3 k(x, z^{(3)}) + b \geq 0$

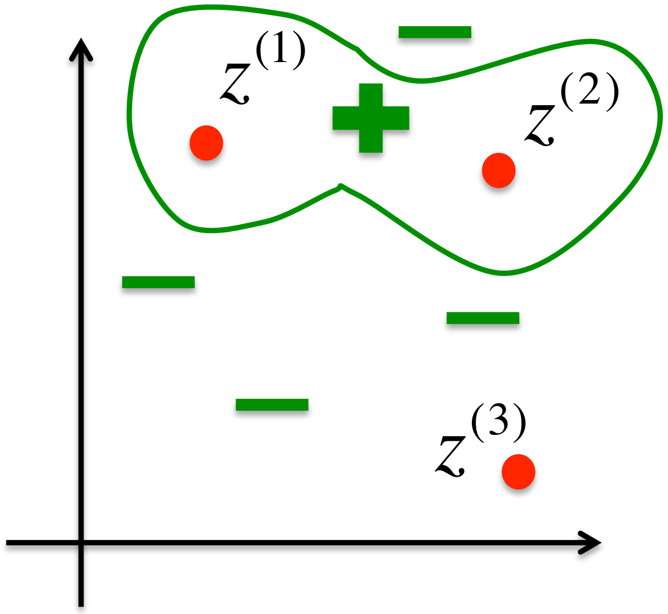
For $x^{(2)}$, we have $k(x^{(1)}, z^{(1)}) \approx 0,\ k(x^{(1)}, z^{(2)}) \approx 0,\ k(x^{(1)}, z^{(3)}) \approx 1$

$\alpha_1(0) + \alpha_2(0) + \alpha_3(1) + b$

$= 1(0) + 1(0) + 0(1) - 0.5$

$= -0.5$, so predict -1

Gaussian kernel example



$$k(x^{(i)}, x^{(j)}) = \exp\left(-\frac{\left\| x^{(i)} - x^{(j)} \right\|^2}{2\sigma^2}\right)$$

Image we've learned that:

$$\alpha = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad b = -0.5$$

Predict $+1$ if $\alpha_1 k(x, z^{(1)}) + \alpha_2 k(x, z^{(2)}) + \alpha_3 k(x, z^{(3)}) + b \geq 0$

Rough sketch of decision boundary

Summary: Kernel methods

- Key idea:
  - Rewrite the algorithm so that we only work with dot products $x^{(i)T}x^{(j)}$ of feature vectors
  - Replace the dot products $x^{(i)T}x^{(j)}$ with a kernel function $k(x^{(i)},x^{(j)})$
- Different kernel functions may be applied to different scenarios
- This "kernel trick" can be applied to many algorithms:
  - Kernel PCA
  - Kernel k-means
- Cons: the optimal parameters and kernel function have to be chosen empirically

SVM vs. logistic regression (Advice from Andrew Ng)

m = # training examples *n* = # features

- If *n* is large (relative to m) (e.g., *n* > *m* with *n* = 10,000, *m* = 10 to 1,000)
  - Use logistic regression or SVM with a linear kernel
- If *n* is small (up to 1,000), *m* is intermediate (up to 10,000)
  - Use SVM with Gaussian kernel
- If *n* is small (up to 1,000), *m* is large (50,000+)
  - Create/add more features, then use logistic regression or SVM without a kernel

Neural networks likely to work well for most of these settings, but may be slower to train.