Indian Institute of Information Technology, Allahabad

CVBL IIIT Allahabad

Introduction to Machine Learning

Support Vector Machine

By

Dr. Shiv Ram Dubey
Associate Professor
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad
Web: https://profile.iiita.ac.in/srdubey/

DISCLAIMER

The content (text, image, and graphics) used in this slide are adopted from many sources for academic purposes. Broadly, the sources have been given due credit appropriately. However, there is a chance of missing out some original primary sources. The authors of this material do not claim any copyright of such material.

Support Vector Machines

- Support vector machine (SVM) is a supervised method for binary classification (two class).

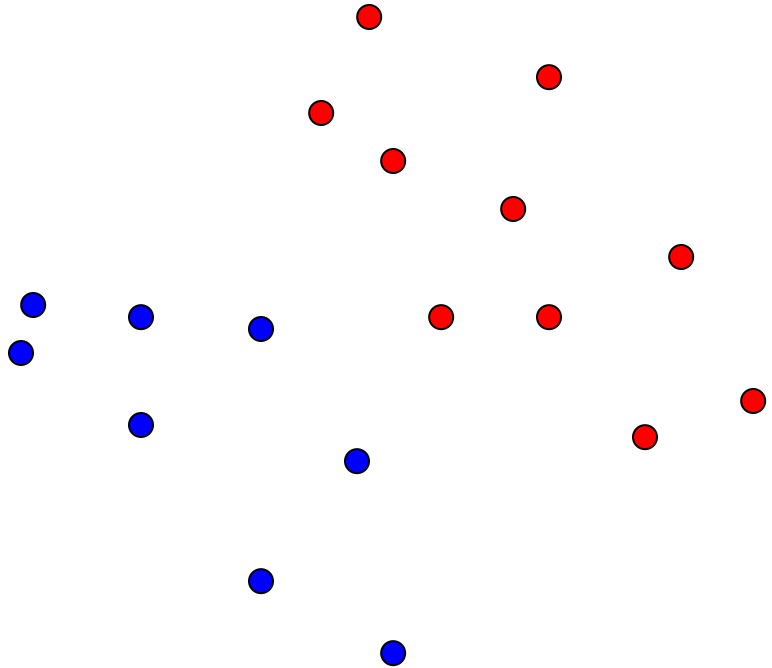
1. **Maximal margin classifier:** only applicable to linearly separable data.

2. **Support vector classifier:** can be applied to data that is not linearly separable. Decision boundary still linear.

3. **Support vector machine:** non-linear decision boundary.

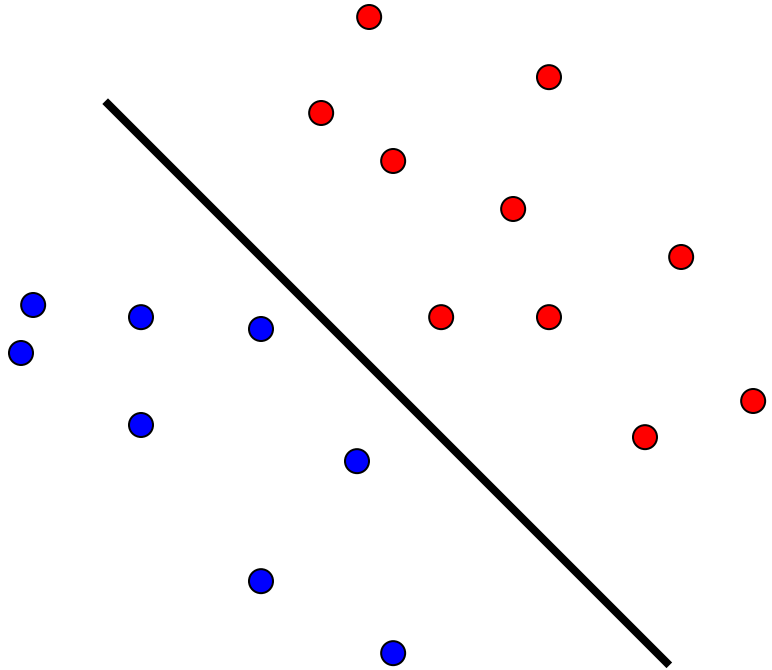
Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



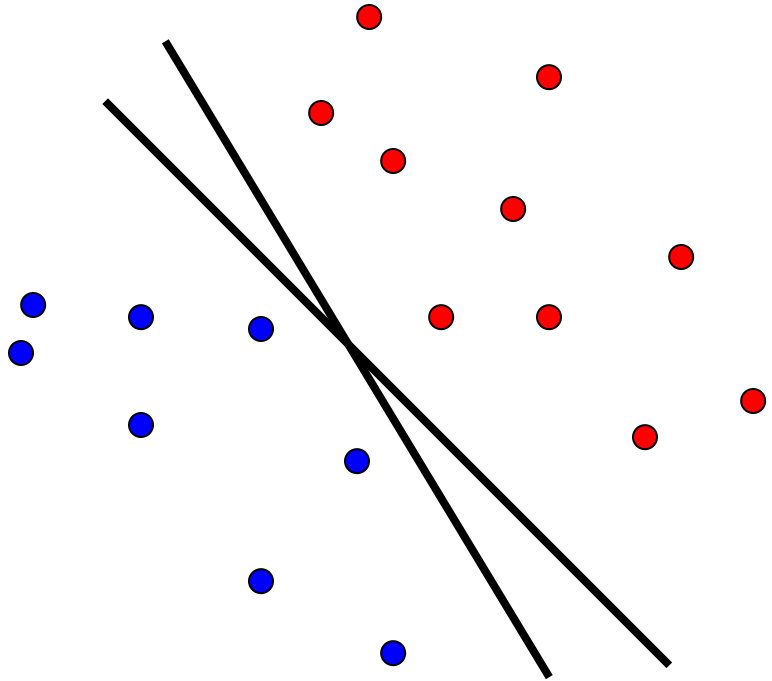
Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



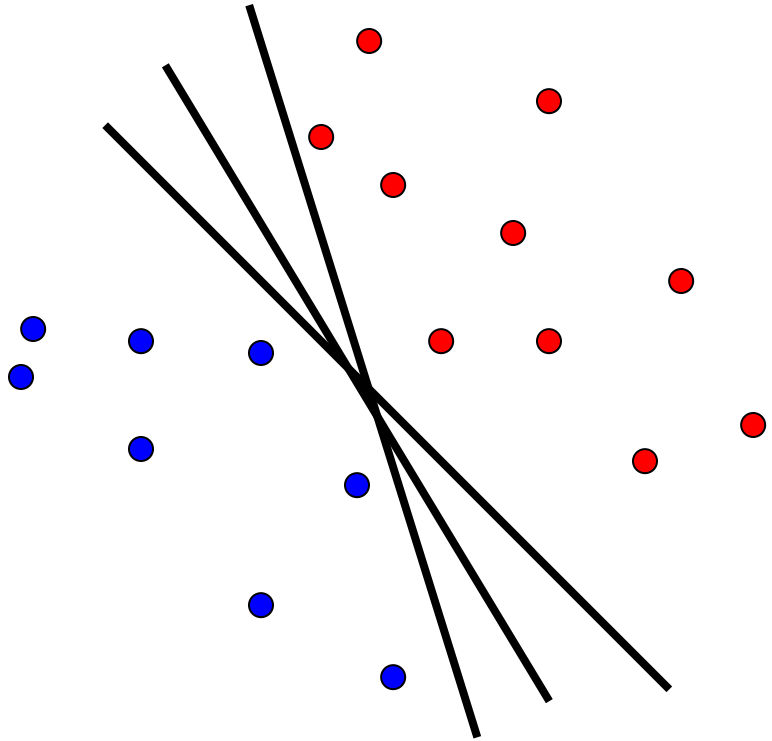
Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



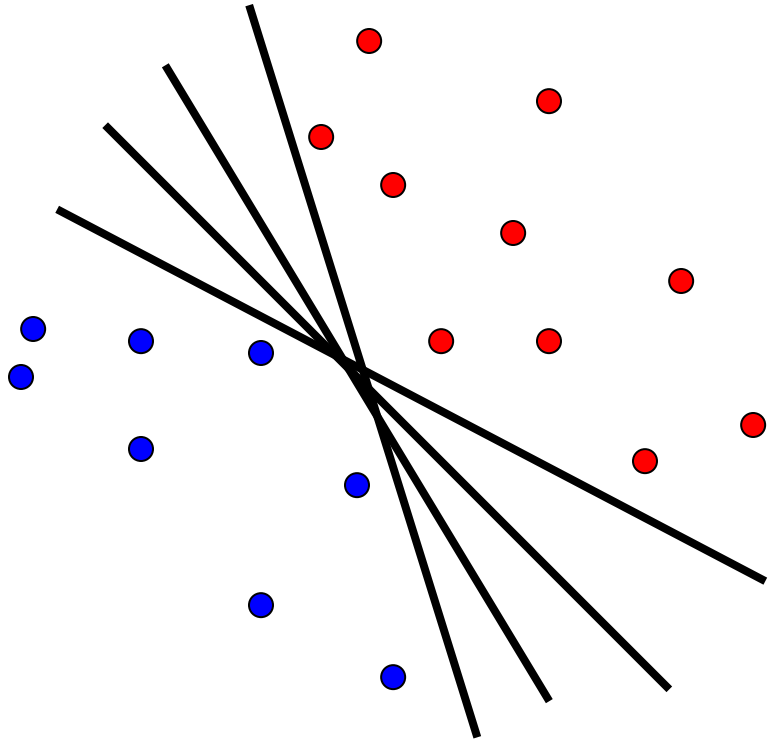
Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



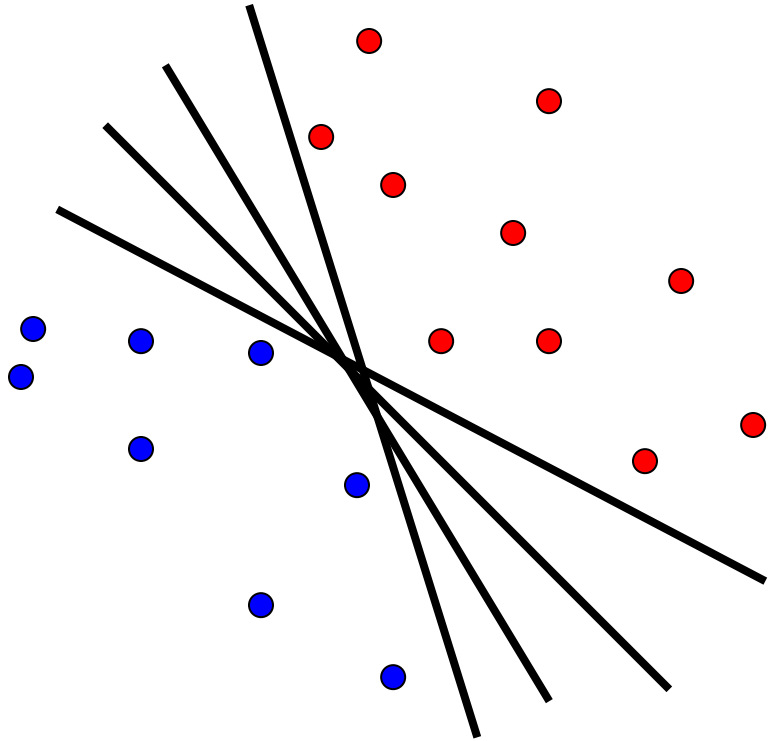
Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



Support Vector Machines

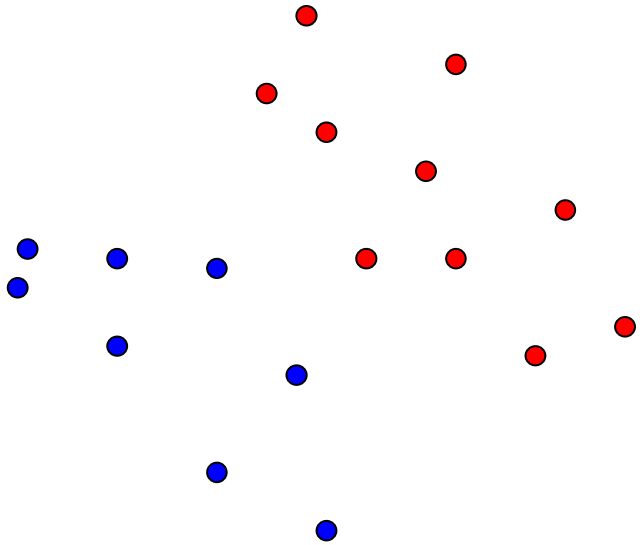
- When the data is linearly separable, there may be more than one separator (hyperplane)



Which separator is best?

Support Vector Machines

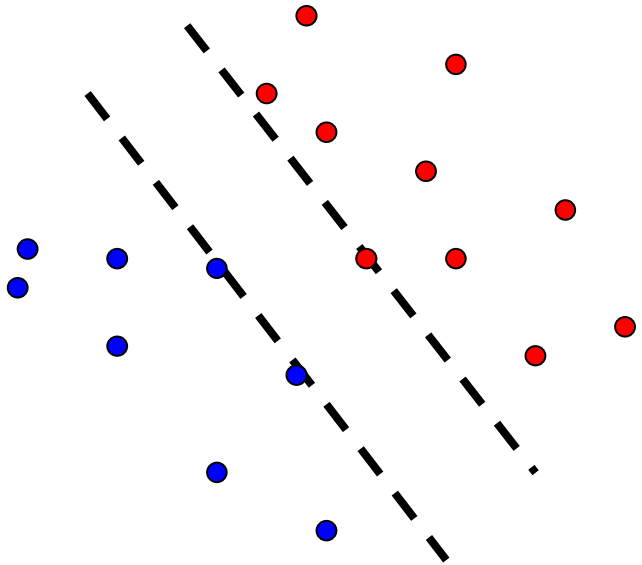
- When the data is linearly separable, there may be more than one separator (hyperplane)



C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

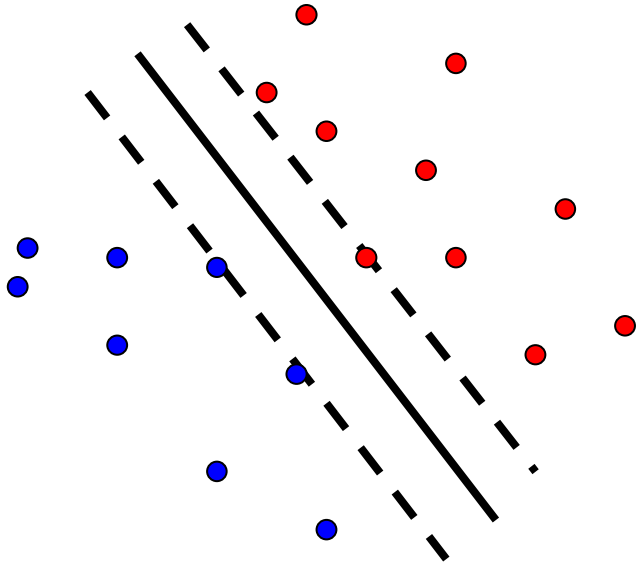
- When the data is linearly separable, there may be more than one separator (hyperplane)



C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

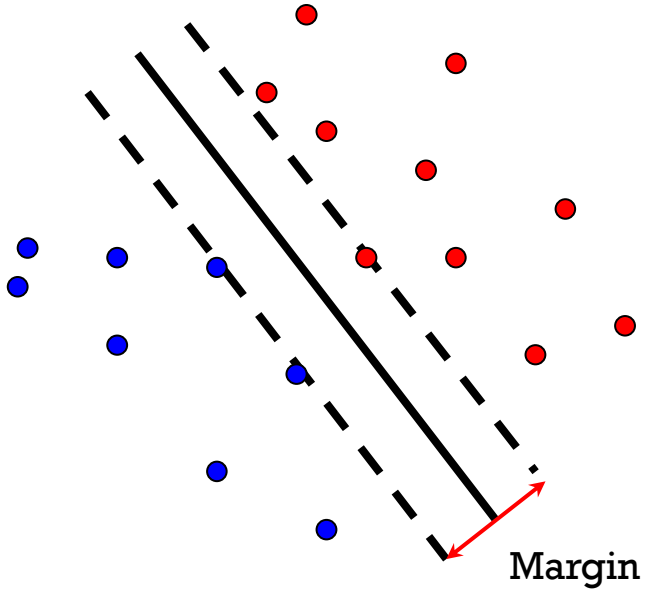
- When the data is linearly separable, there may be more than one separator (hyperplane)



C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

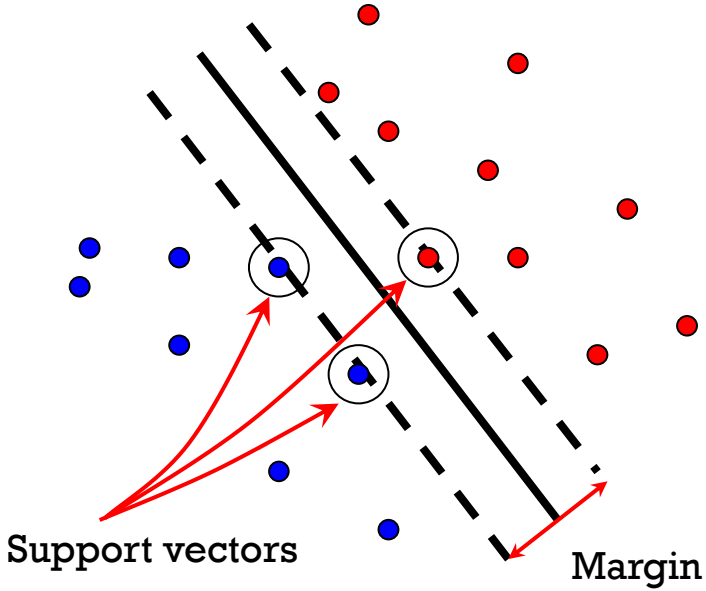
- When the data is linearly separable, there may be more than one separator (hyperplane)



C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

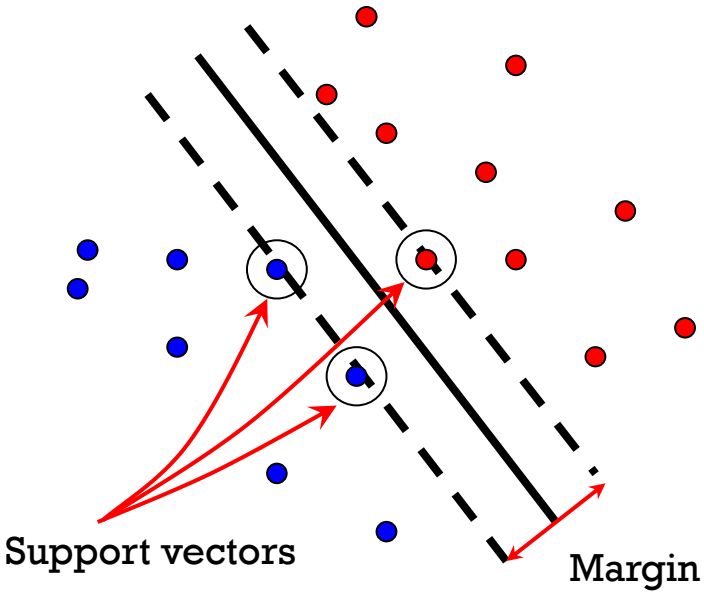
- When the data is linearly separable, there may be more than one separator (hyperplane)



C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



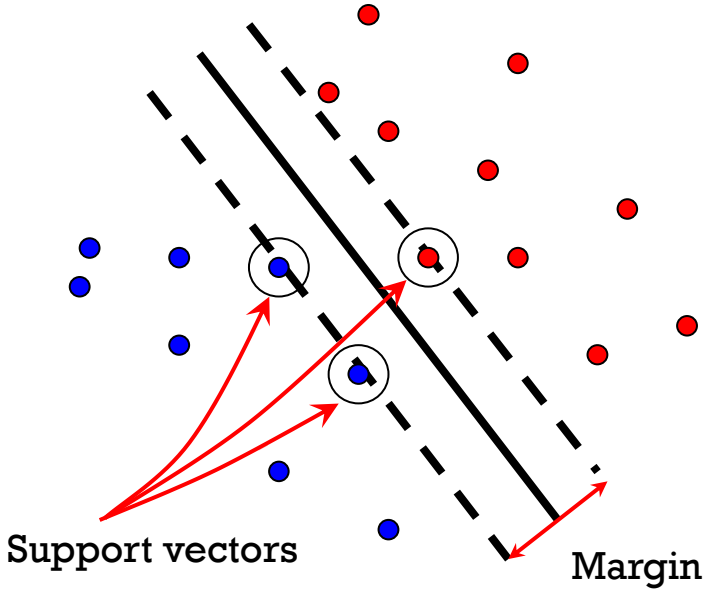
Why is it called a support vector?

- **"Support":** maximal margin hyperplane only depends on these observations.
- **"Vector":** points are vectors in p-dimensional space.
- If support vectors are perturbed, then MM hyperplane will change.
- If other training observations perturbed (provided not perturbed within margin distance of hyperplane), then MM hyperplane not affected.

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



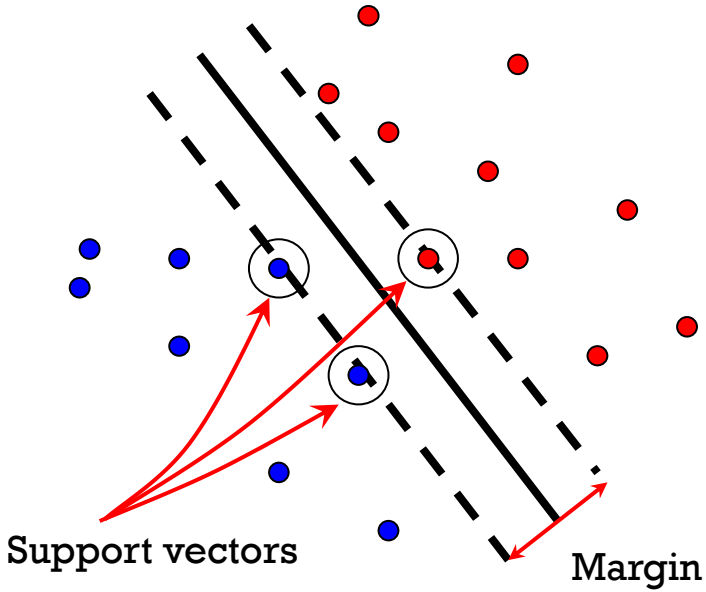
$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

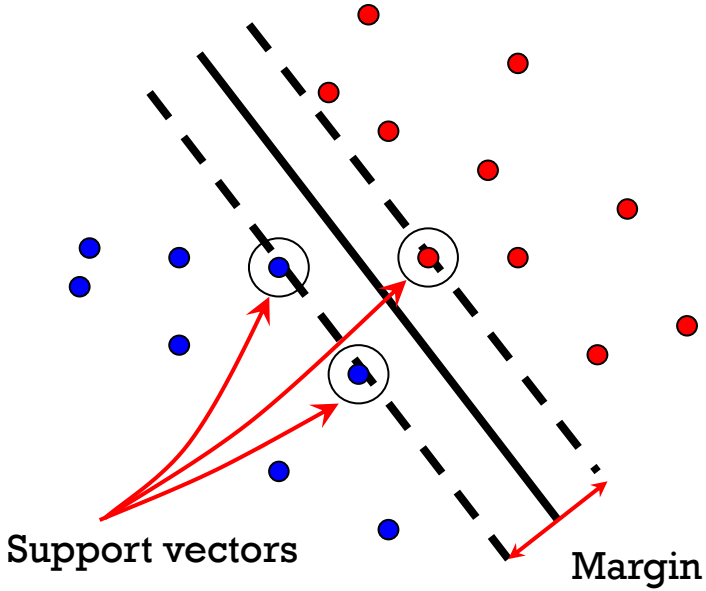
$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

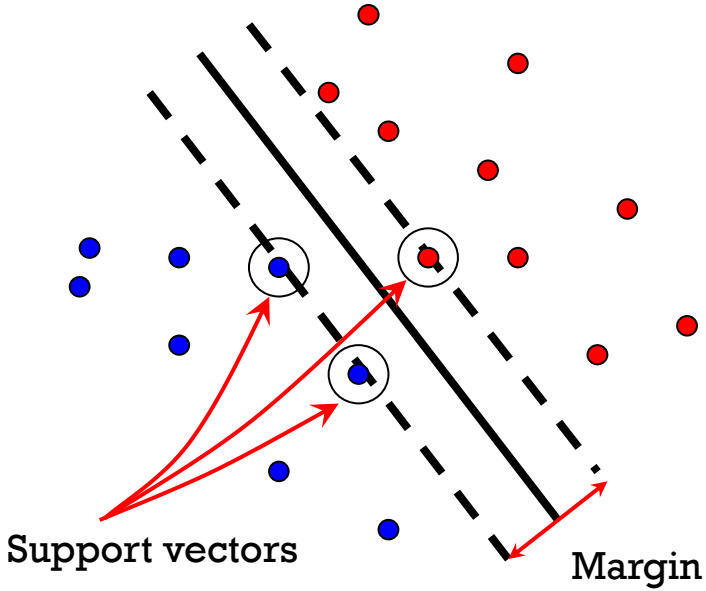
For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\frac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)



$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the Maximum Margin Hyperplane

1. **Maximize margin** $2 / \|\mathbf{w}\|$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the Maximum Margin Hyperplane

1. Maximize margin $2 / \|\mathbf{w}\|$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the Maximum Margin Hyperplane

1. Maximize margin $2 / \|\mathbf{w}\|$
2. Correctly classify all training data:

$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

- This is equivalent to *Quadratic optimization problem*:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Solving Optimal Margin Classifier

- This is equivalent to

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} w^T w \\ \text{s.t} \quad & 1 - y_i(w^T x_i + b) \le 0, \quad \forall i \end{aligned} \qquad (*)$$

- Write the Lagrangian:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i \left[y_i(w^T x_i + b) - 1 \right]$$

Recall that (*) can be reformulated as $\min_{w,b} \max_{\alpha_i \ge 0} \mathcal{L}(w,b,\alpha)$

Now we solve its **dual problem**: $\max_{\alpha_i \ge 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$

The Dual Problem

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i \left[y_i (w^T x_i + b) - 1 \right]$$

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$$

- We minimize *L* with respect to *w* and *b* first:

$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0, \qquad (*)$$

$$\nabla_b \mathcal{L}(w,b,\alpha) = \sum_{i=1}^{n} \alpha_i y_i = 0, \qquad (**)$$

Note that (*) implies:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \qquad (***)$$

- Plus (***) back to *L*, and using (**), we have:

$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

The Dual Problem

- Now we have the following dual opt problem:

$$\max_{\alpha} \mathcal{J}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

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

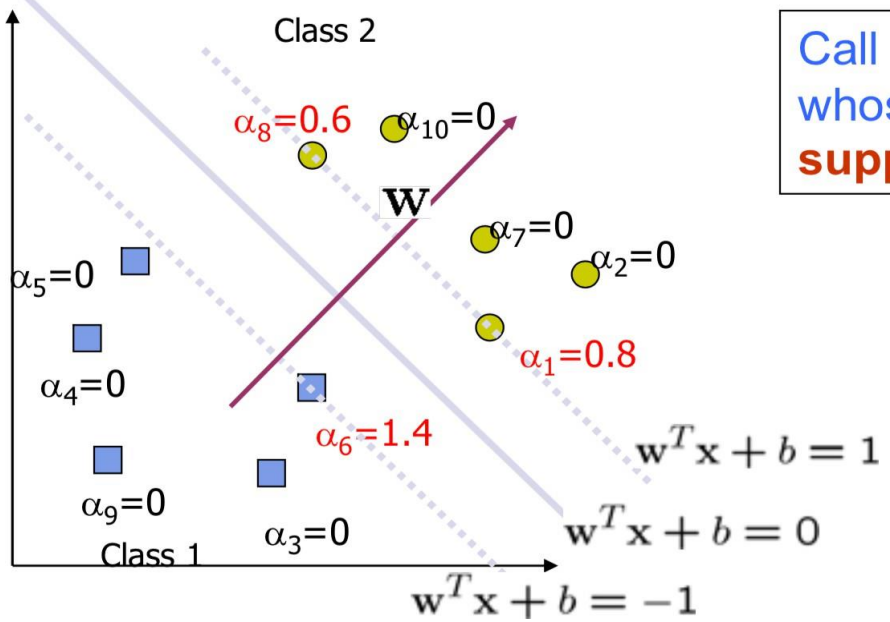
$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

- This is,(again,) a **quadratic programming** problem.
  - A global maximum of α_i can always be found.
  - Note two things:
    1. w can be recovered by $w = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$
    2. The "kernel" $\mathbf{x}_i^T \mathbf{x}_j$

Support Vectors

- Note the Karush-Kuhn-Tucker (KKT) condition

  --- only a few Lagrange multipliers α_i's can be nonzero!

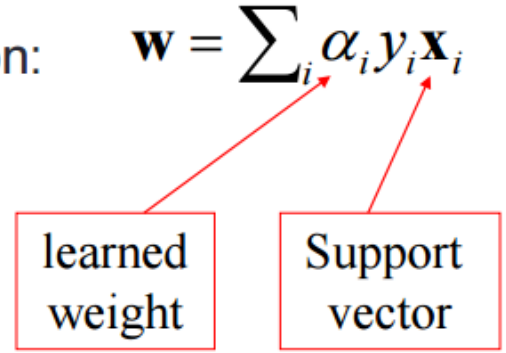


Call the training data points whose α_i's are nonzero the **support vectors** (SV)

Finding the Maximum Margin Hyperplane

Once we have the Lagrange multipliers $\{\alpha_i\}$, we can reconstruct the parameter vector w as a weighted combination of the training examples:

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$



- The weights α_i are non-zero only at support vectors.

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the Maximum Margin Hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Finding the Maximum Margin Hyperplane

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Classification function:

$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + \mathrm{b})$$

If f(x) < 0, classify as negative,

$$= \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

if f(x) > 0, classify as positive

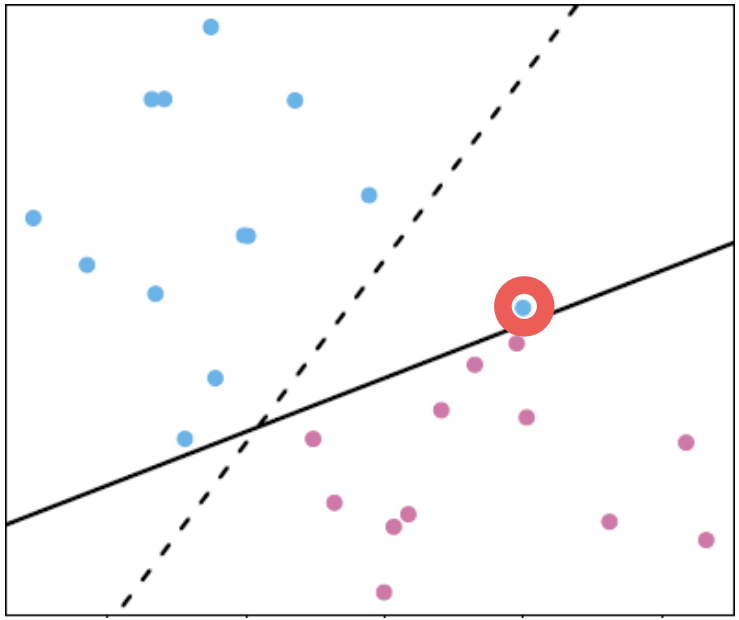
Dot product only!

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Maximal Margin Classifier

Furthermore, notice a disadvantage of the maximal margin classifier:

- Can be sensitive to individual observations
- May overfit training data



Support Vector Classifier

Like the maximal margin classifier, it looks for a hyperplane to perform classification.

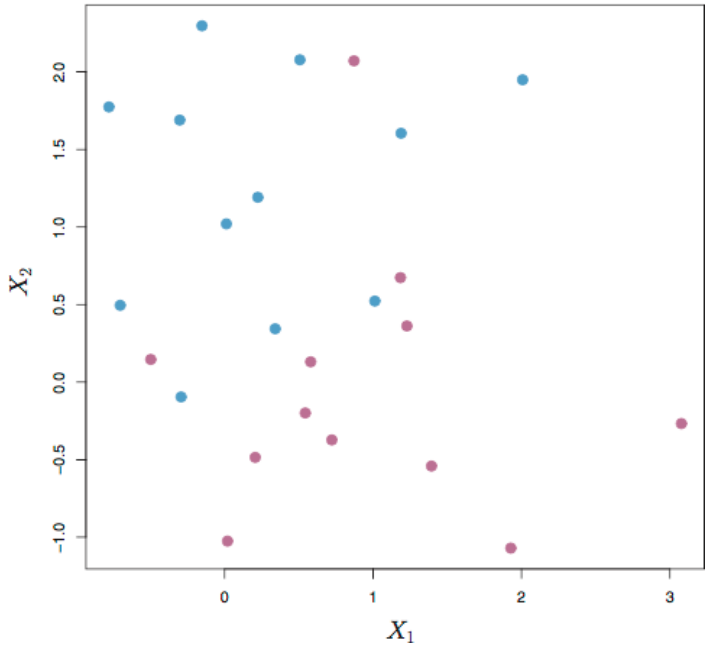


FIGURE 9.4, ISL (8th printing 2017)

Support Vector Classifier

Like the maximal margin classifier, it looks for a hyperplane to perform classification.

However, training samples are allowed to be on the “wrong side” of the margin or hyperplane.

This hyperplane *almost* separates the classes using a “soft margin”.

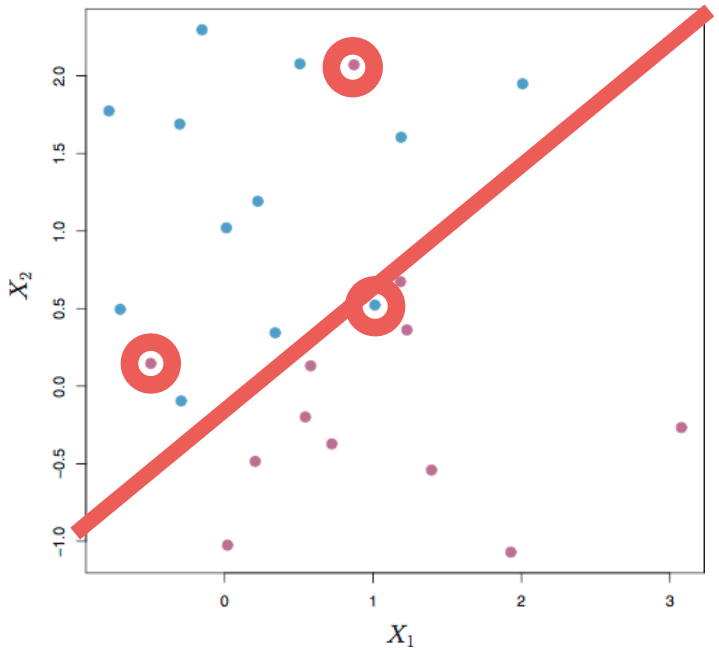


FIGURE 9.4, ISL (8th printing 2017)

Support Vector Classifier

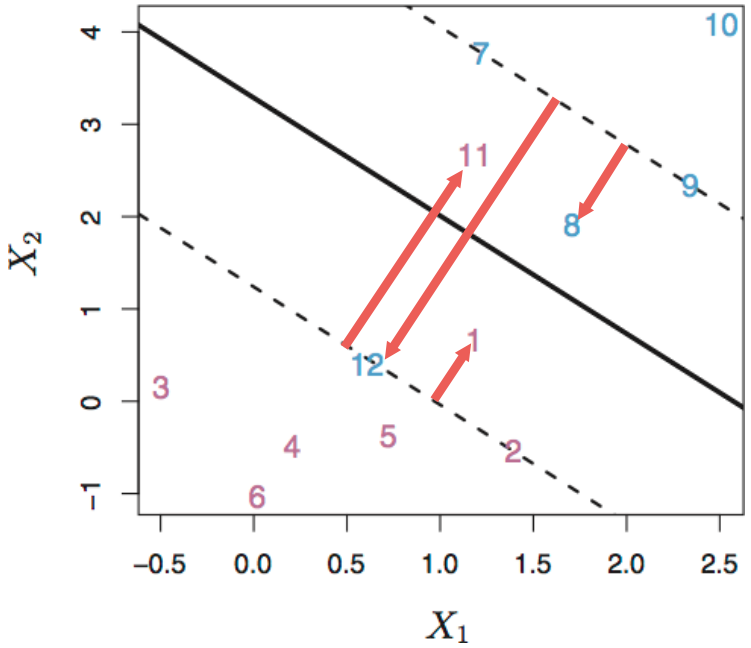


FIGURE 9.6, ISL (8th printing 2017)

Some points are allowed to violate the margin.

Support Vector Classifier

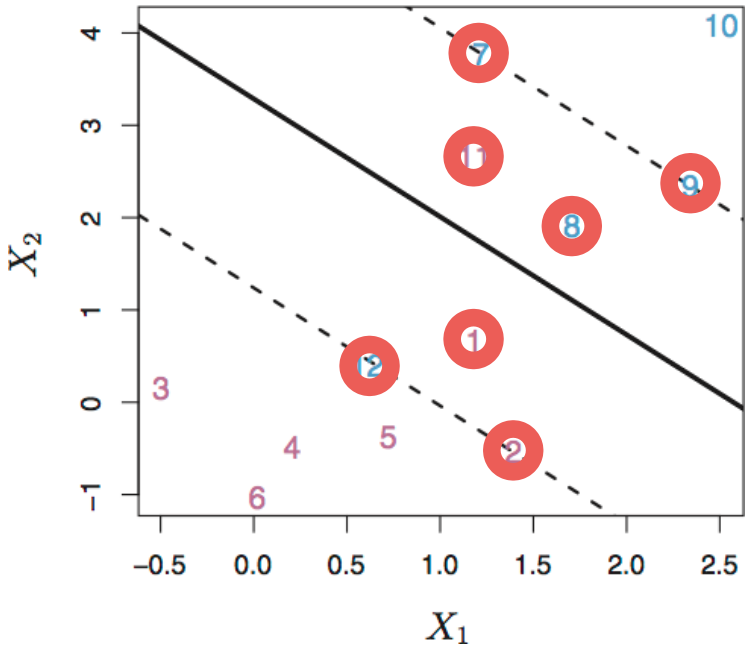


FIGURE 9.6, ISL (8th printing 2017)

Support vector classifiers also have support vectors.

They are points lying directly on the margin, or on the wrong side of the margin for their class.

These observations affect the hyperplane.

SVM Parameter Learning

- **Separable data:** $\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$ subject to $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$

Maximize margin

Classify training data correctly

SVM Parameter Learning

- **Separable data:** $\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$ subject to $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$

  Maximize margin

  Classify training data correctly

- **Non-separable data:**

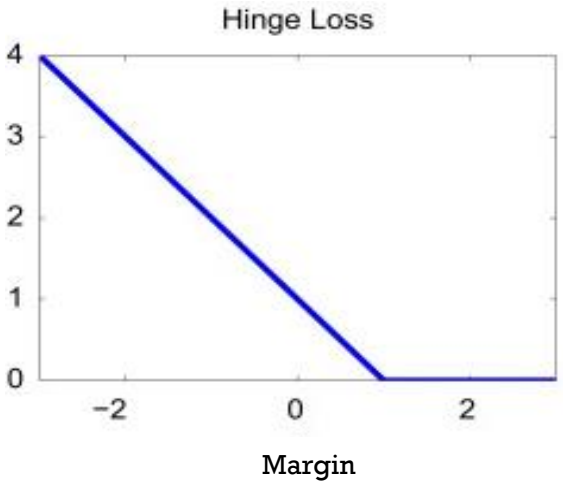
$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b))$$

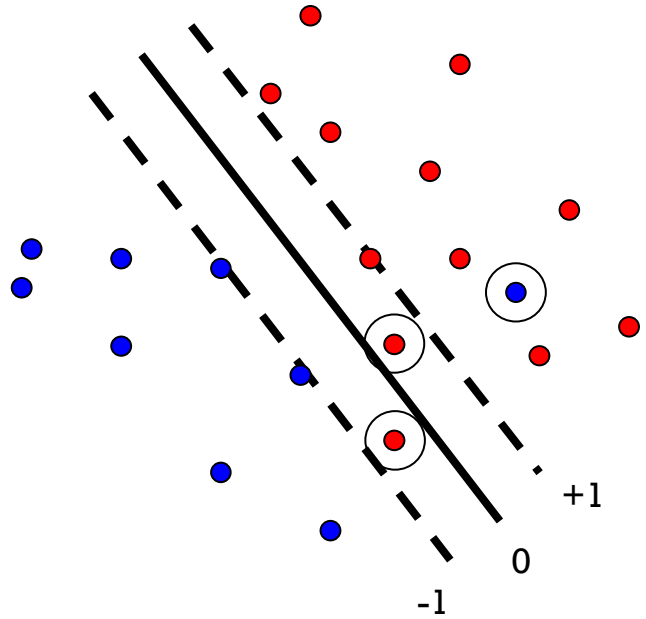
  Maximize margin

  Minimize classification mistakes

SVM Parameter Learning

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b))$$





- Demo: http://cs.stanford.edu/people/karpathy/svmjs/demo

Bias, Variance and C

Penalty parameter C is the total "budget" for violations

- Large C
  - Large violation budget
  - Large margin
  - Many support vectors
- Small C
  - Small violation budget
  - Small margin
  - Few support vectors

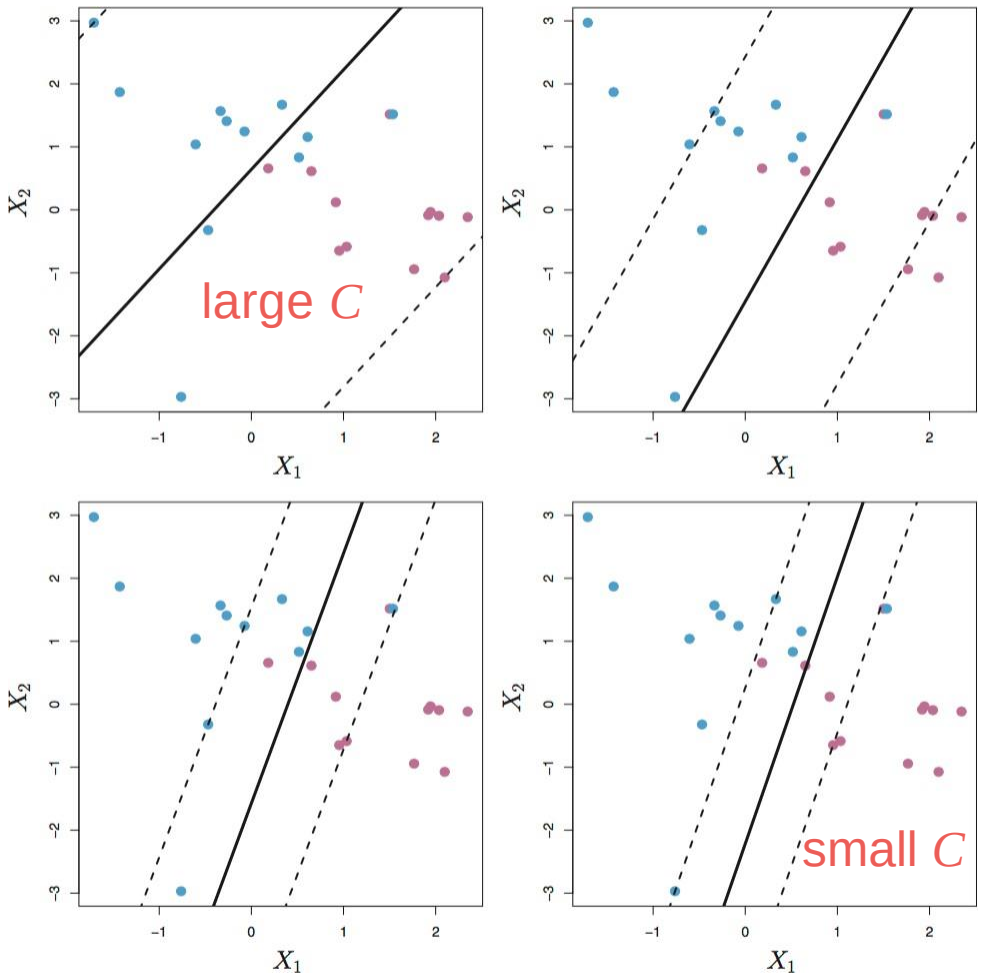


FIGURE 9.7, ISL (8th printing 2017)

Bias, Variance and C

Penalty parameter C is the total "budget" for violations

- Large C
  - High bias
  - Low variance
- Small C
  - Low bias
  - High variance

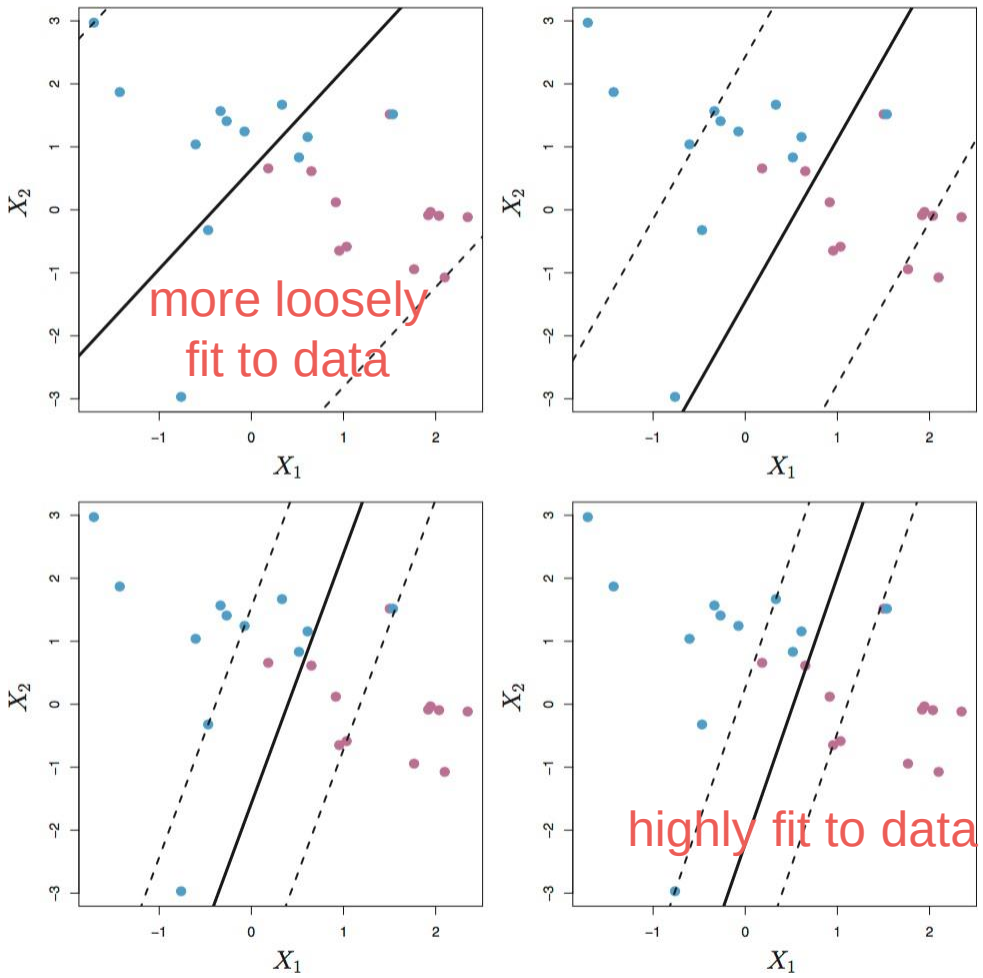


FIGURE 9.7, ISL (8th printing 2017)

Nonlinear SVMs

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

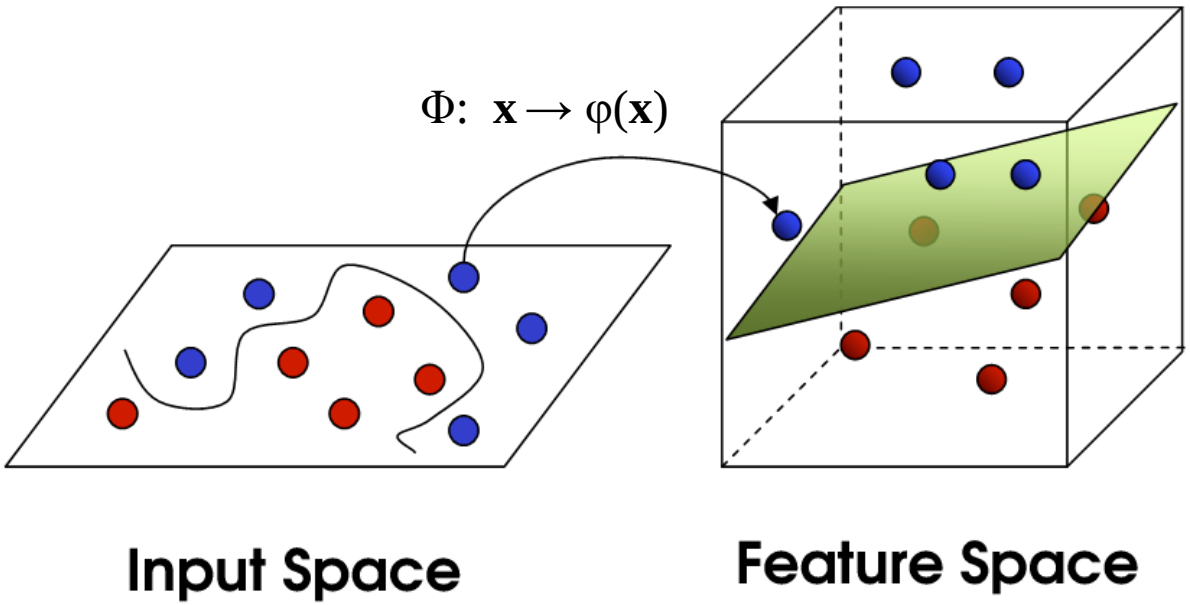
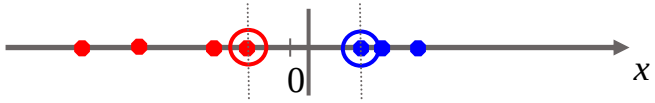


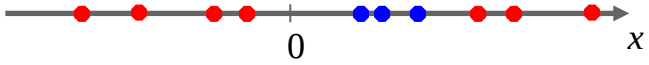
Image source

Nonlinear SVMs

- Linearly separable dataset in 1D:



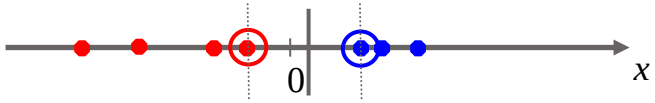
- Non-linearly separable dataset in 1D:



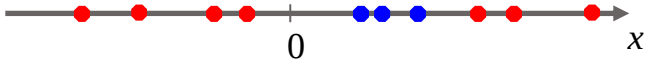
Slide credit: Andrew Moore

Nonlinear SVMs

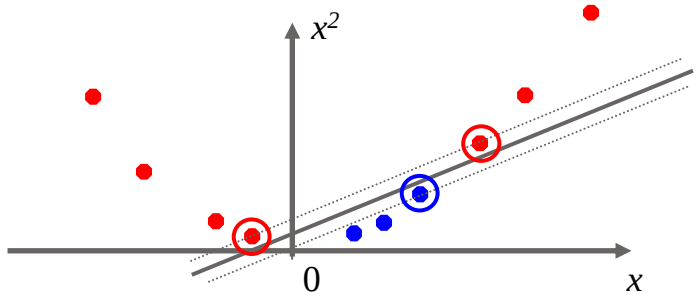
- Linearly separable dataset in 1D:



- Non-linearly separable dataset in 1D:



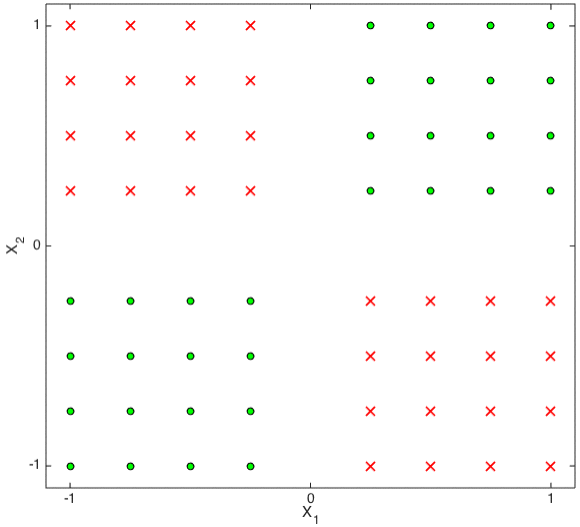
- We can map the data to a *higher-dimensional space*:



Slide credit: Andrew Moore

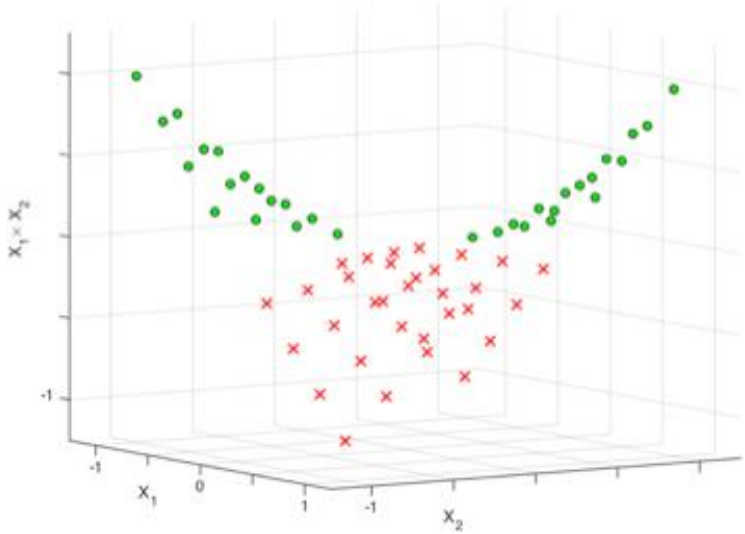
Expanding Feature Space

Original feature space



variables x_1, x_2

New feature space



variables x_1, x_2, x_1x_2

The Kernel Trick

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

- **The kernel trick:** instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}, \mathbf{y}) = \boldsymbol{\varphi}(\mathbf{x}) \cdot \boldsymbol{\varphi}(\mathbf{y})$$

The Kernel Trick

- Linear SVM decision function:

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

learned weight

Support vector

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

The Kernel Trick

- Linear SVM decision function:

$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Kernel SVM decision function:

$$\sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- This gives a nonlinear decision boundary in the original feature space

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Why use kernels?

Why use kernels instead of explicitly constructing a larger feature space?

Computational advantage:

$$\phi : \mathbb{R}^p \to \mathbb{R}^P, \quad p \ll P$$

$$K(x, x') = \langle \phi(x), \phi(x') \rangle \quad \text{in } O(p)$$

The Kernel Trick: Example

2-dimensional vectors $x=[x_1 \ x_2]$;

let $K(x_i,x_j)=(1 + x_i^T x_j)^2$

from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html

The Kernel Trick: Example

2-dimensional vectors $x=[x_1 \; x_2]$;

let $K(x_i,x_j)=(1 + x_i^T x_j)^2$

Need to show that $K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$:

The Kernel Trick: Example

2-dimensional vectors $x=[x_1 \; x_2]$;

let $K(x_i,x_j)=(1 + x_i^T x_j)^2$

Need to show that $K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$:

$K(x_i,x_j)=(1 + x_i^T x_j)^2$,

$= 1+ x_{i1}^2 x_{j1}^2 + 2\, x_{i1} x_{j1}\, x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2}$

from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html A. Borbick

The Kernel Trick: Example

2-dimensional vectors $x=[x_1 \; x_2]$;

let $K(x_i,x_j)=(1 + x_i^T x_j)^2$

Need to show that $K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$:

$K(x_i,x_j)=(1 + x_i^T x_j)^2$,

$= 1+ x_{i1}^2 x_{j1}^2 + 2\, x_{i1} x_{j1}\, x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2}$

$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1} x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2} x_{i1} \;\; \sqrt{2} x_{i2}]^T$

$[1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1} x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2} x_{j1} \;\; \sqrt{2} x_{j2}]$

from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html

The Kernel Trick: Example

2-dimensional vectors $x=[x_1 \; x_2]$;

let $K(x_i,x_j)=(1 + x_i^T x_j)^2$

Need to show that $K(x_i,x_j)= \varphi(x_i)^T \varphi(x_j)$:

$K(x_i,x_j)=(1 + x_i^T x_j)^2$,

$= 1+ x_{i1}^2 x_{j1}^2 + 2\, x_{i1} x_{j1}\, x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2}$

$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1} x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2} x_{i1} \;\; \sqrt{2} x_{i2}]^T$

$[1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1} x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2} x_{j1} \;\; \sqrt{2} x_{j2}]$

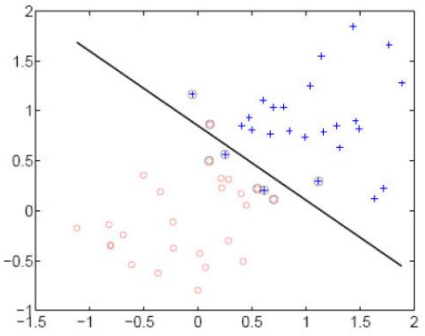
$= \varphi(x_i)^T \varphi(x_j)$,

where $\varphi(x) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1 x_2 \;\; x_2^2 \;\; \sqrt{2} x_1 \;\; \sqrt{2} x_2]$

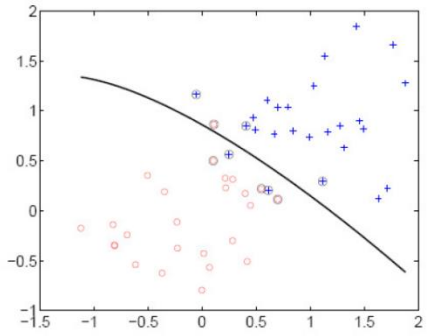
from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html

Polynomial Kernel

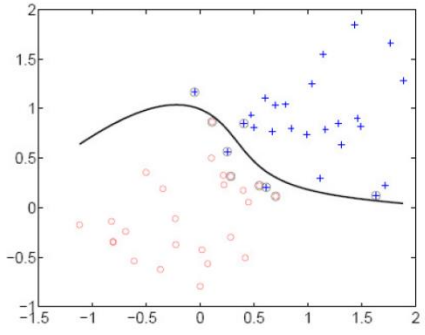
$$K(\mathbf{x}, \mathbf{y}) = (c + \mathbf{x} \cdot \mathbf{y})^d$$



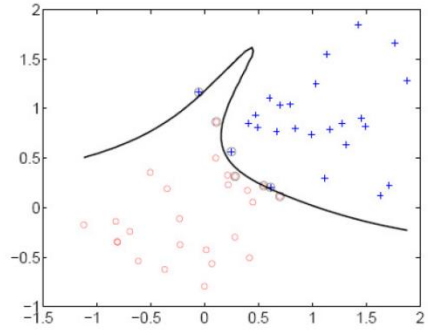
linear



2^{nd} order polynomial



4^{th} order polynomial

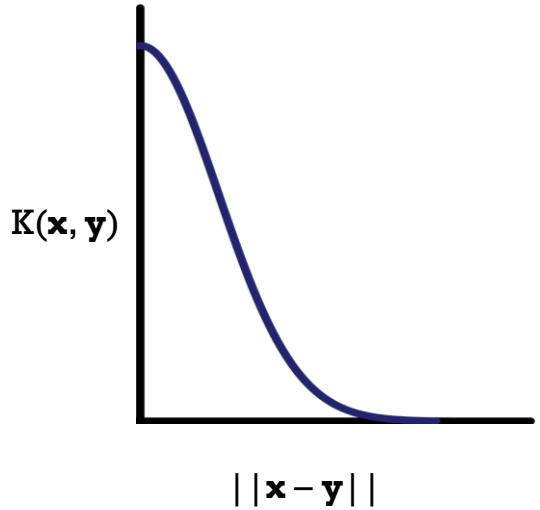


8^{th} order polynomial

Gaussian Kernel

- Also known as the radial basis function (RBF) kernel:

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{\sigma^2} \|\mathbf{x} - \mathbf{y}\|^2 \right)$$



Kernels for Histograms

- **Histogram intersection:**

$$K(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- **Square root (Bhattacharyya kernel):**

$$K(h_1, h_2) = \sum_{i=1}^{N} \sqrt{h_1(i) h_2(i)}$$

SVMs: Pros and Cons

- Pros
  - Kernel-based framework is **very powerful**, flexible
  - Training is convex optimization, **globally optimal solution** can be found
  - SVMs work very well in practice, even with **very small training** sample sizes

- Cons
  - No “direct” **multi-class SVM**, must combine two-class SVMs (e.g., with one-vs-others)
  - **Computation, memory** (esp. for nonlinear SVMs)

Multiclass Support Vector Machine Loss

- i^{th} example: image x_i and the label y_i
- Score for the j^{th} class: $s_j = f(x_i, W)_j$

Source: cs231n, http://cs231n.github.io/linear-classify/

Multiclass Support Vector Machine Loss

- i^{th} example: image x_i and the label y_i
- Score for the j^{th} class: $s_j = f(x_i, W)_j$

- The Multiclass SVM loss for the i^{th} example is then formalized as follows:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$

Hinge Loss

Margin

Source: cs231n, http://cs231n.github.io/linear-classify/

Multiclass Support Vector Machine Loss

- i^{th} example: image x_i and the label y_i
- Score for the j^{th} class: $s_j = f(x_i, W)_j$

- The Multiclass SVM loss for the i^{th} example is then formalized as follows:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$

Hinge Loss

Margin

Problem: W is not necessarily unique

Source: cs231n, http://cs231n.github.io/linear-classify/

Multiclass Support Vector Machine Loss

- **Regularization Penalty:**

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

Source: cs231n, http://cs231n.github.io/linear-classify/

Multiclass Support Vector Machine Loss

- **Regularization Penalty:**

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

- **Hinge Loss:**

$$L = \underbrace{\frac{1}{N} \sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

Source: cs231n, http://cs231n.github.io/linear-classify/

Multiclass Support Vector Machine Loss

- **Regularization Penalty:**

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

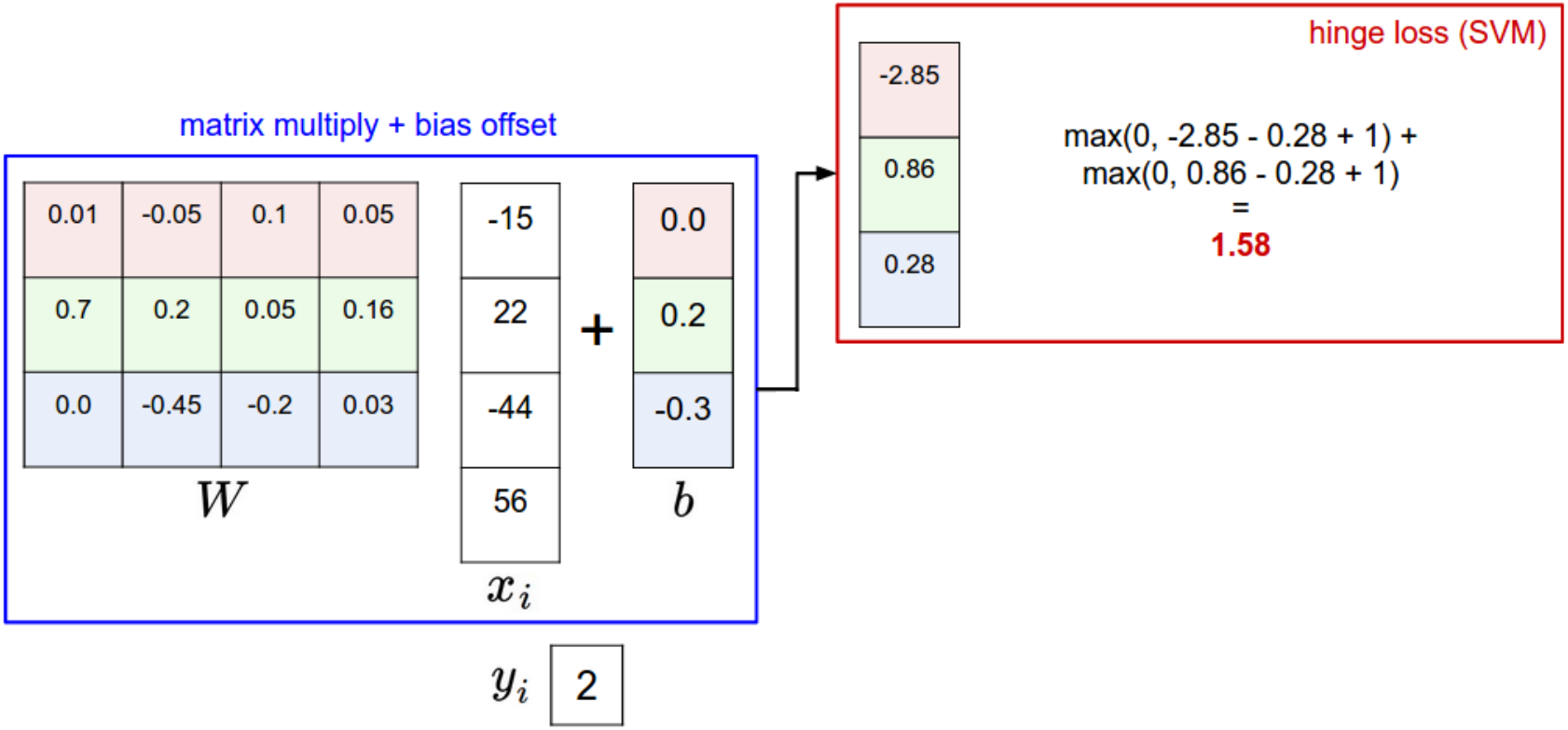
- **Hinge Loss:**

$$L = \underbrace{\frac{1}{N} \sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

$$L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda \sum_k \sum_l W_{k,l}^2$$

Source: cs231n, http://cs231n.github.io/linear-classify/

Hinge Loss



Source: cs231n, http://cs231n.github.io/linear-classify/

Softmax Classifier

- Interprets the class scores as the unnormalized log probabilities for each class and replace the *hinge loss* with a **cross-entropy loss** that has the form:

$$L_i = -log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) = -s_{y_i} + log\sum_j e^{s_j}$$

Source: cs231n, http://cs231n.github.io/linear-classify/

Softmax Classifier

- Interprets the class scores as the unnormalized log probabilities for each class and replace the *hinge loss* with a **cross-entropy loss** that has the form:

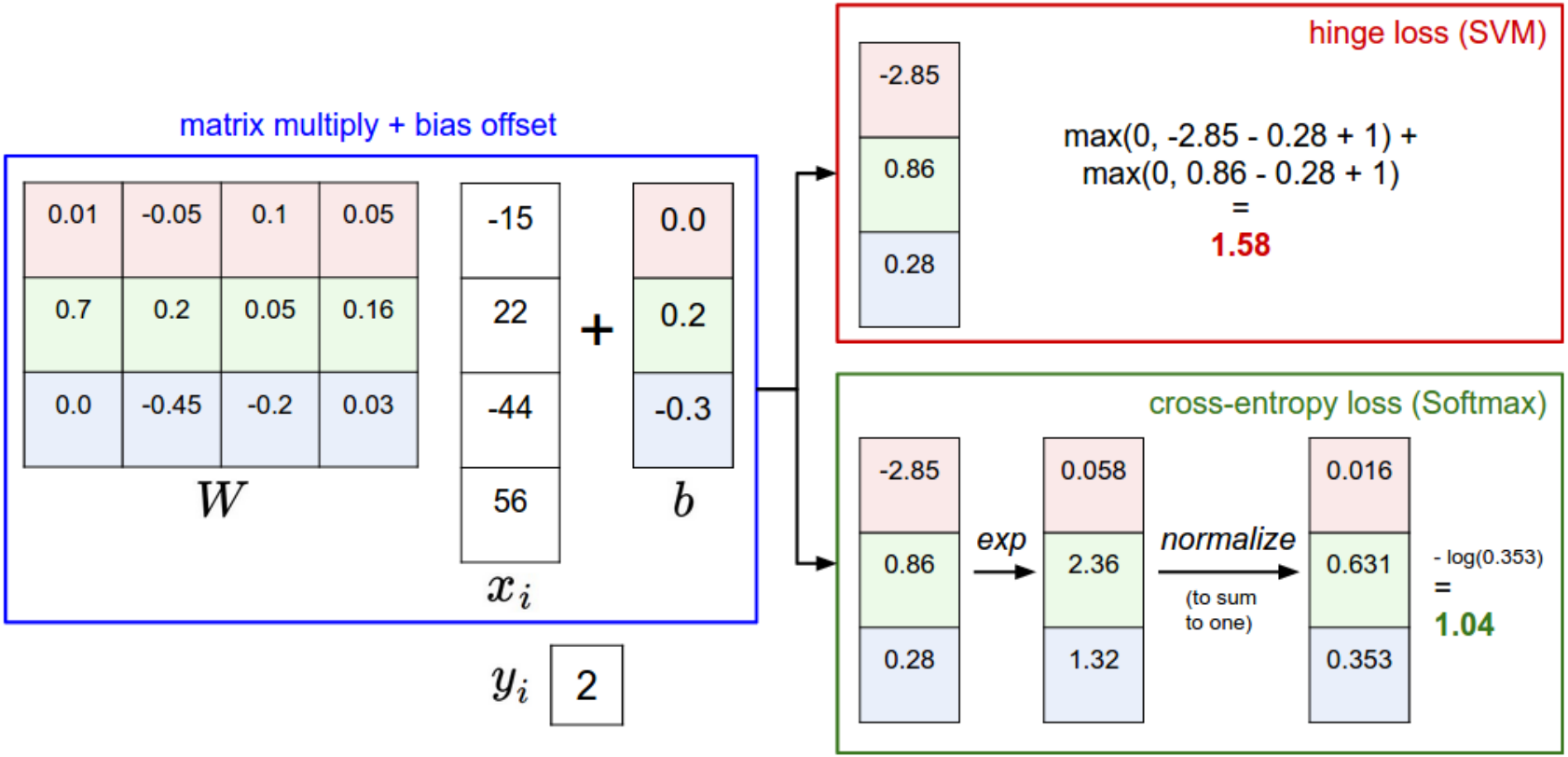
$$L_i = -log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) = -s_{y_i} + log\sum_j e^{s_j}$$

- Softmax Loss:

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

Source: cs231n, http://cs231n.github.io/linear-classify/

Hinge vs Cross-Entropy Loss



Source: cs231n, http://cs231n.github.io/linear-classify/

Acknowledgement

Most of the slides are based on the ML course of:

- CS231n, Stanford University
- Sherrie Wang, Stanford University
- Yifeng Tao, Carnegie Mellon University

THANK YOU