Indian Institute of Information Technology, Allahabad

CVBL IIIT Allahabad

Image Classifiers

By

Dr. Satish Kumar Singh & Dr. Shiv Ram Dubey
Computer Vision and Biometrics Lab
Department of Information Technology
Indian Institute of Information Technology, Allahabad

TEAM

Computer Vision and Biometrics Lab (CVBL)

Department of Information Technology

Indian Institute of Information Technology Allahabad

Course Instructors

Dr. Satish Kumar Singh, Associate Professor, IIIT Allahabad (Email: sk.singh@iiita.ac.in)

Dr. Shiv Ram Dubey, Assistant Professor, IIIT Allahabad (Email: srdubey@iiita.ac.in)

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.

PREVIOUS CLASS

Image features and categorization

Choosing right features

Object, Scene, Action, etc.

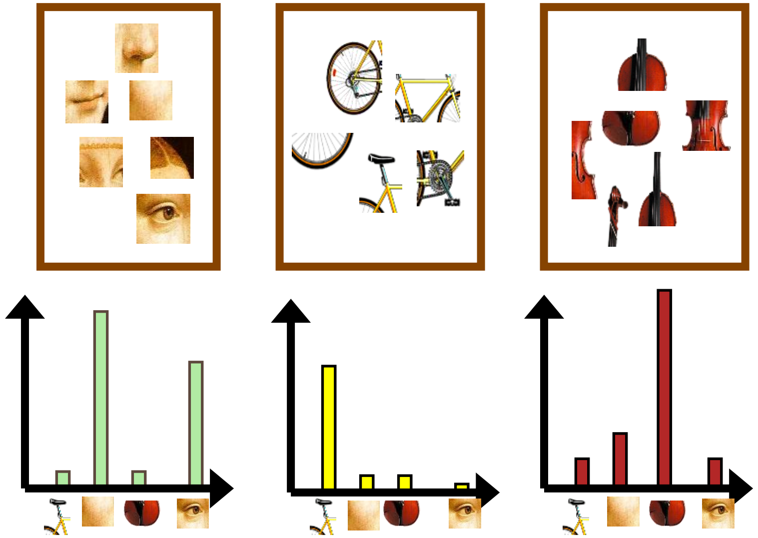
Bag-of-visual-words

Extract local features

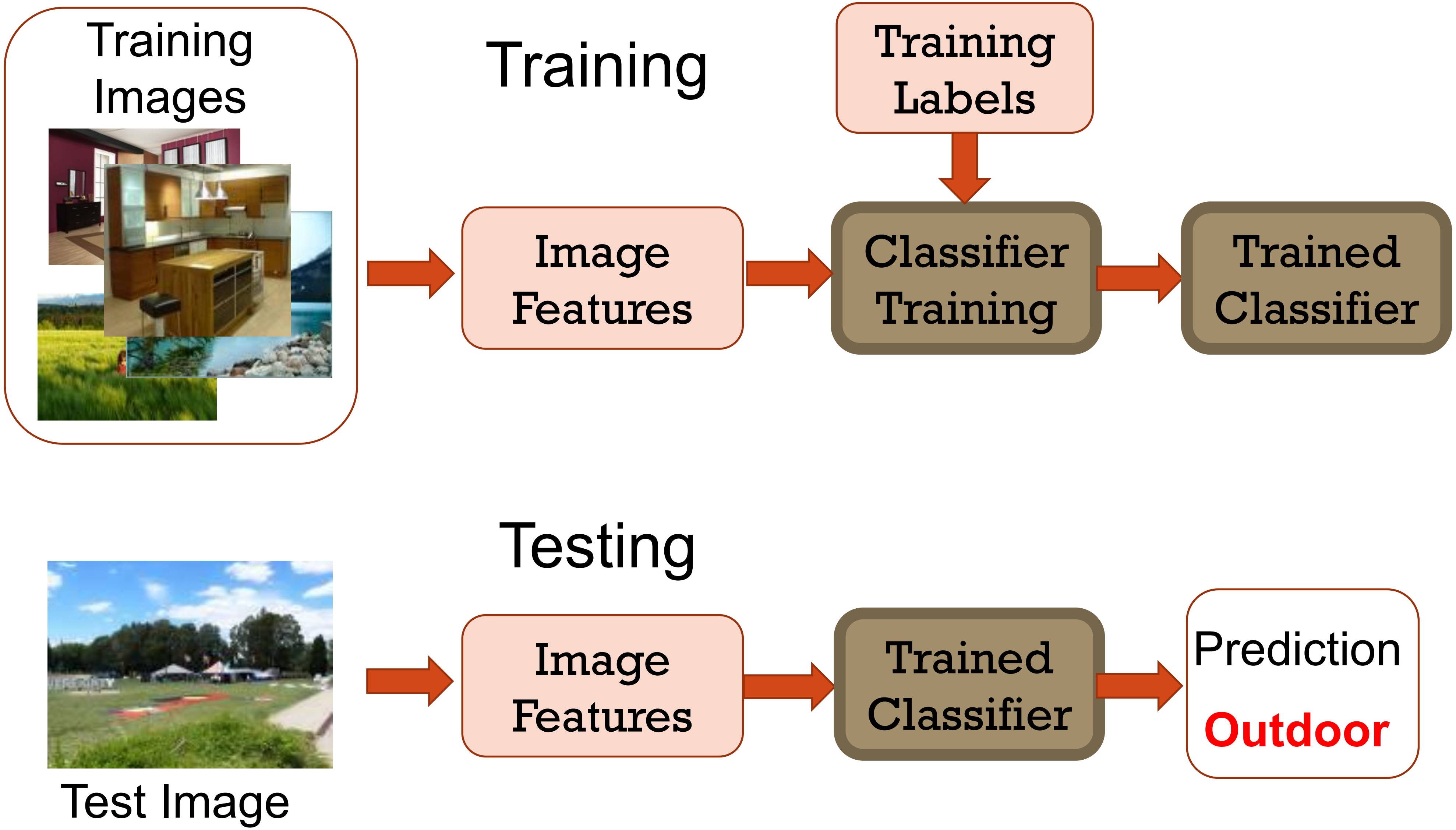
Learn "visual vocabulary"

Quantize features using visual vocabulary

Represent by frequencies of "visual words"



TODAY'S CLASS



TODAY'S CLASS

K - Nearest Neighbor Classifier

Linear Classifier

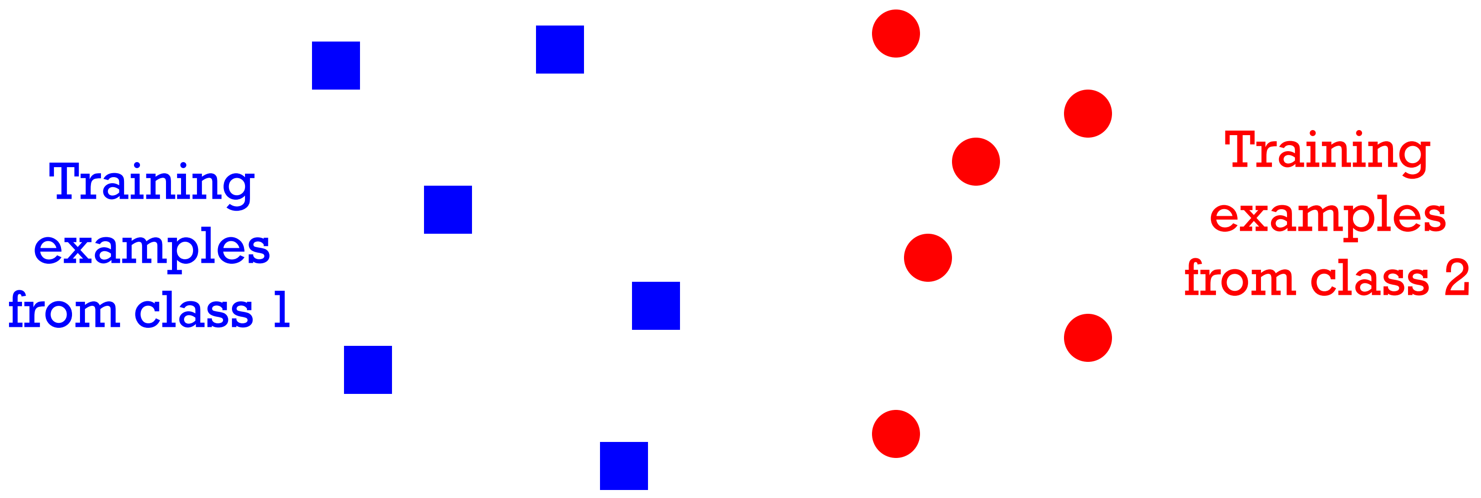
Support Vector Machine

Non-linear SVM

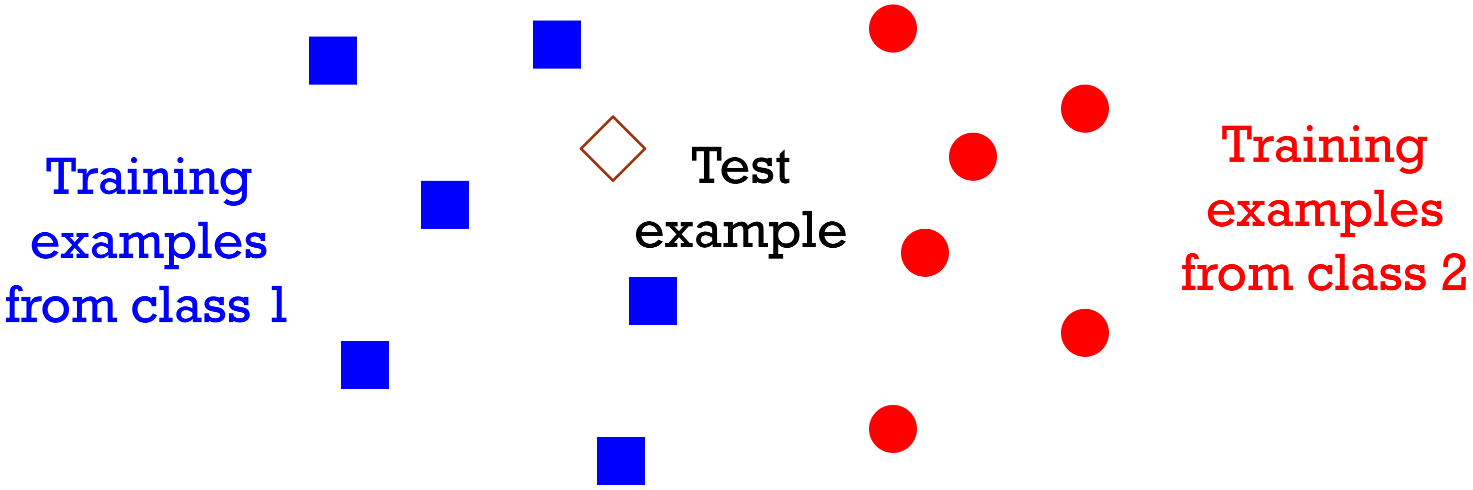
Multi-class SVM

Softmax Classifier

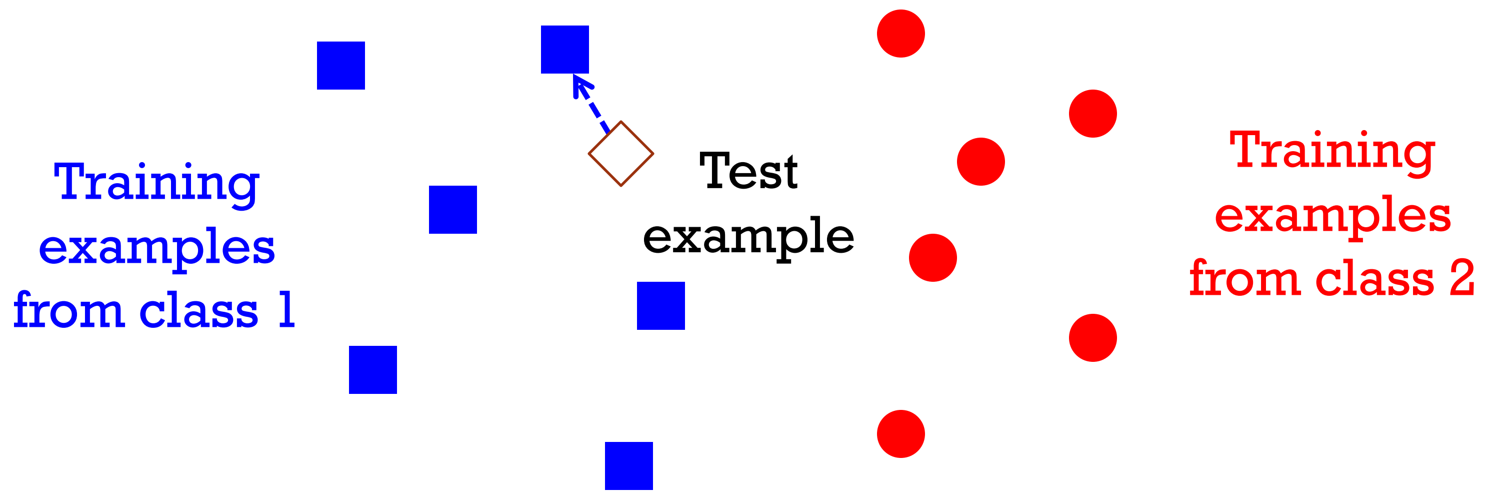
CLASSIFIERS: NEAREST NEIGHBOR



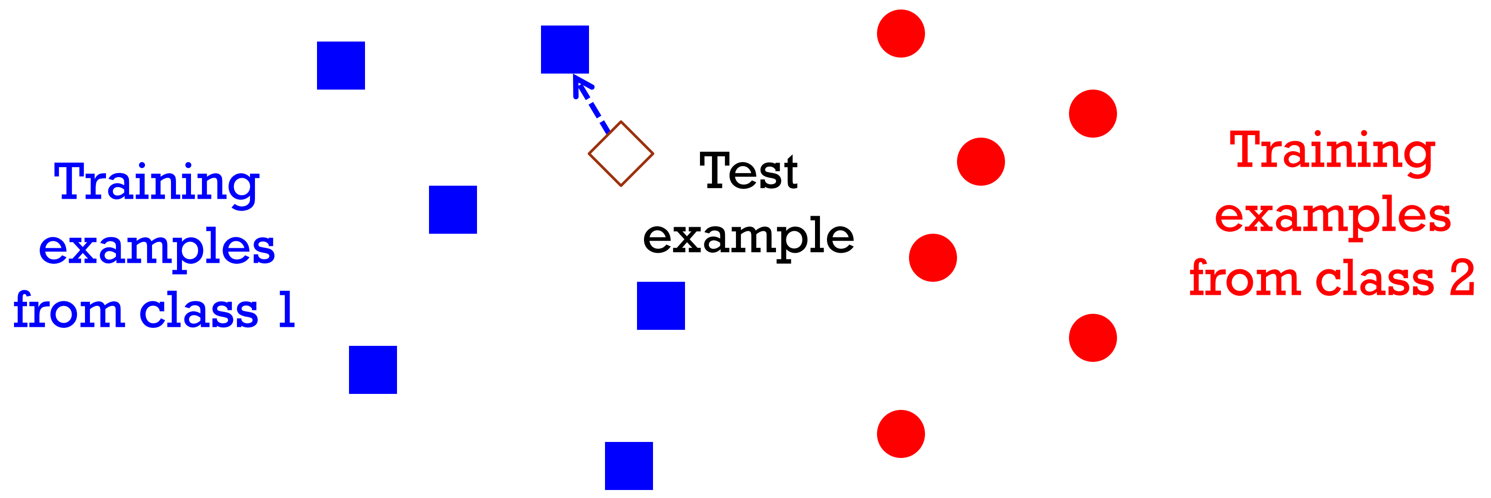
CLASSIFIERS: NEAREST NEIGHBOR



CLASSIFIERS: NEAREST NEIGHBOR



CLASSIFIERS: NEAREST NEIGHBOR

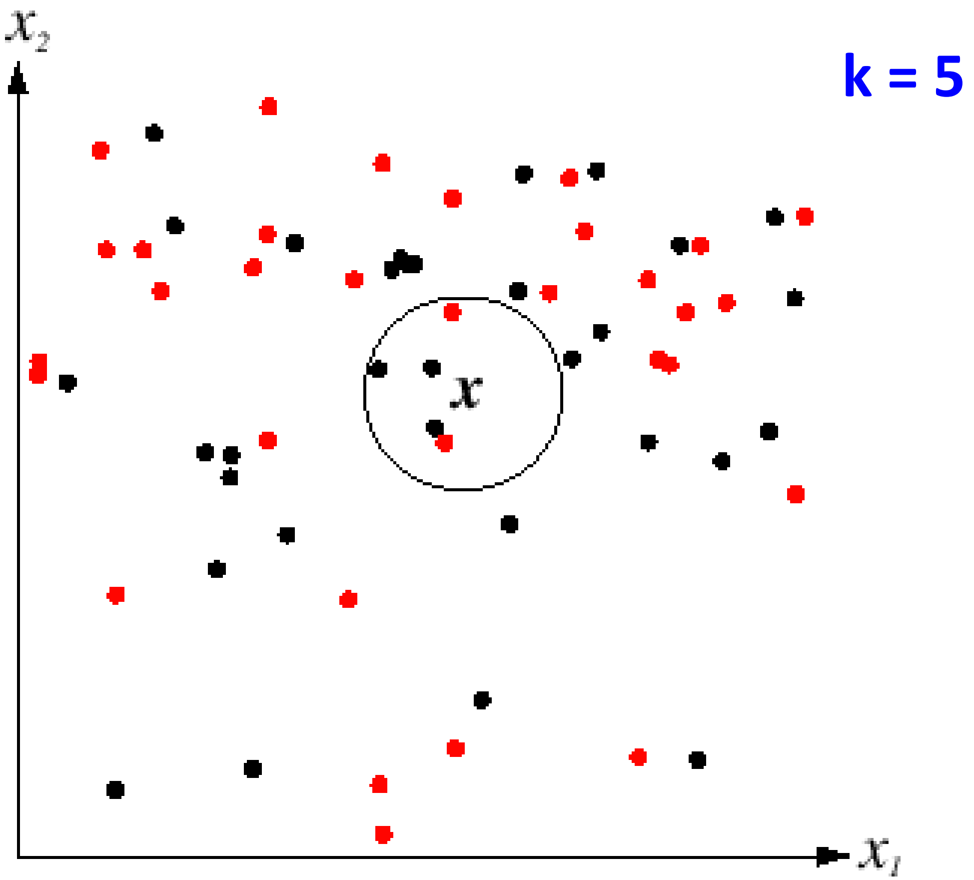


f(**x**) = label of the training example nearest to **x**

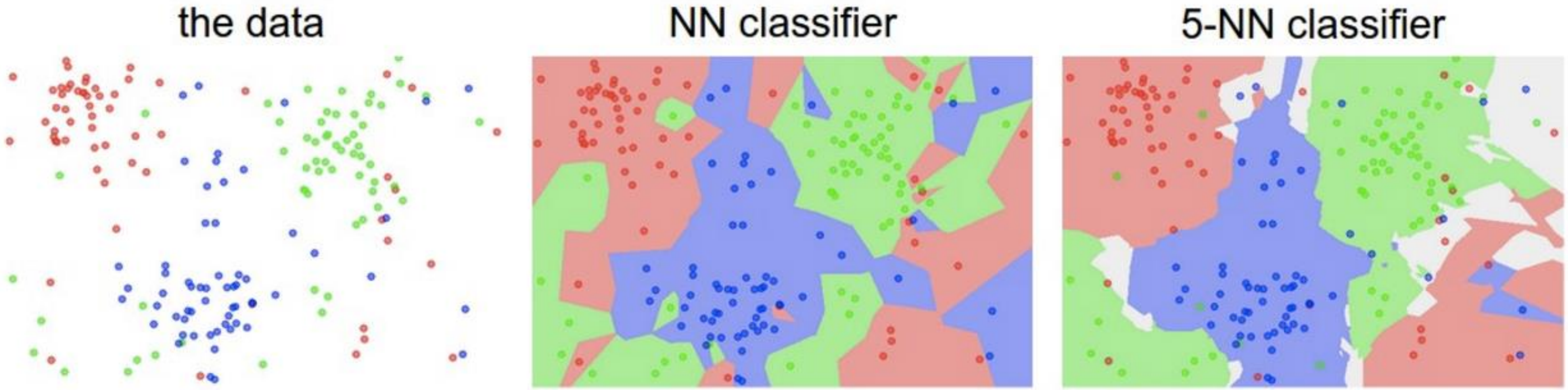
- All we need is a distance function for our inputs
- No training required!

K-NEAREST NEIGHBOR CLASSIFIER

- For a new point, find the k closest points from training data
- Vote for class label with labels of the k points

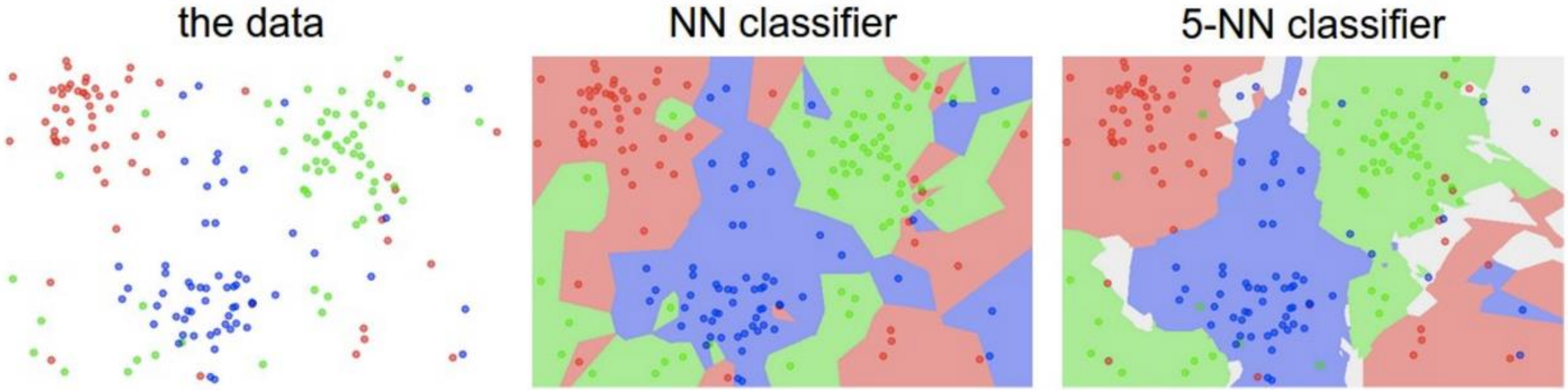


K-NEAREST NEIGHBOR CLASSIFIER



Credit: cs231n, http://cs231n.github.io/classification/

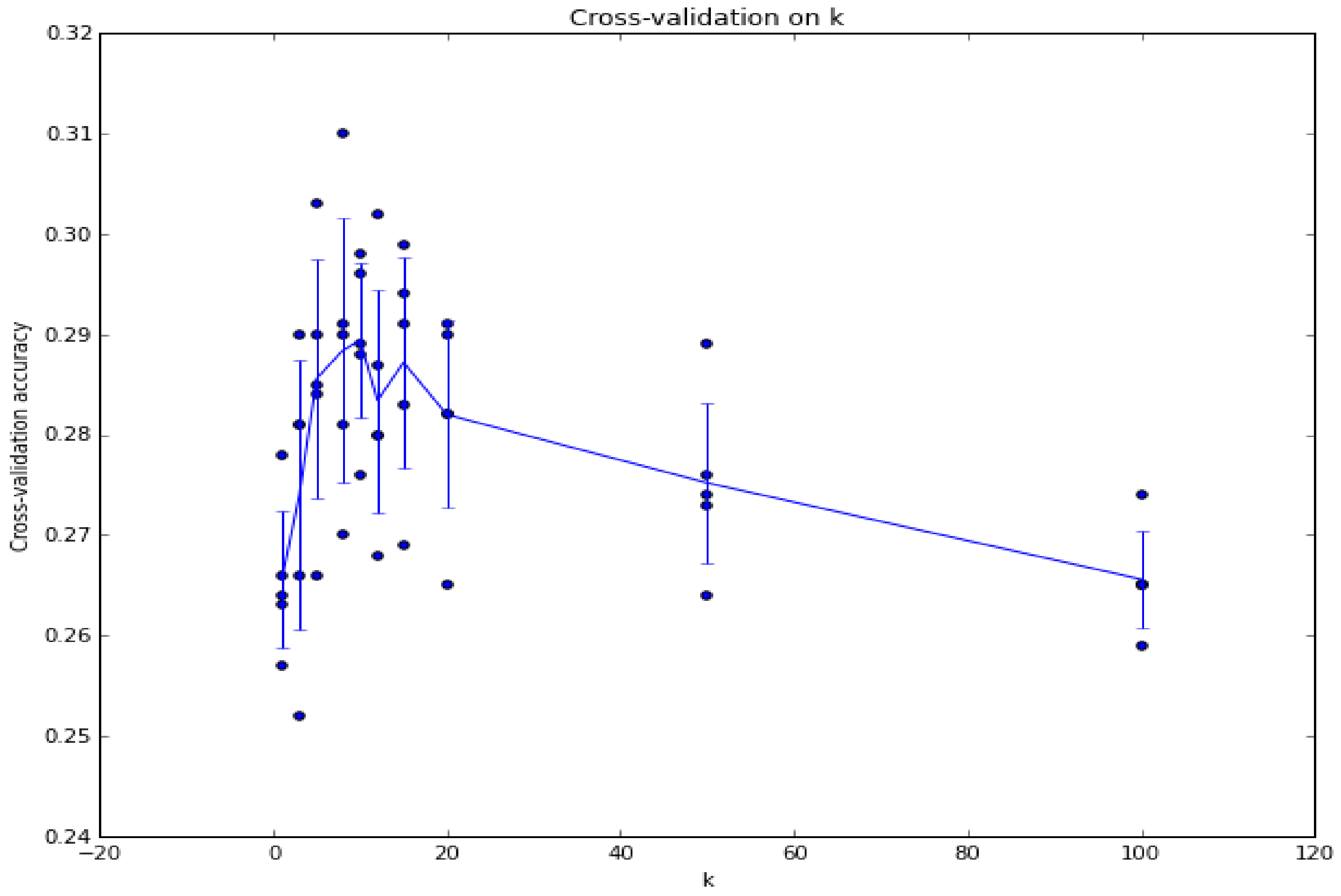
K-NEAREST NEIGHBOR CLASSIFIER



- Which classifier is more robust to *outliers*?

Credit: cs231n, http://cs231n.github.io/classification/

CHOICE OF K IN KNN CLASSIFIER



Credit: cs231n, http://cs231n.github.io/classification/

Example of a 5-fold cross-validation run for the parameter **k**. Note that in this particular case, the cross-validation suggests that a value of about **k** = 7 works best on this particular CIFAR10 dataset (corresponding to the peak in the plot).

CLASSIFIERS: K-NEAREST NEIGHBOR

"Non-parametric" classifier: the entire training set is essentially the model parameters.

CLASSIFIERS: K-NEAREST NEIGHBOR

"Non-parametric" classifier: the entire training set is essentially the model parameters.

Pros:

- **Very fast at training** time
- **Flexible**: all it requires is a way to compute similarity or distances between pairs of features. Applies to many different kinds of features.
- Works with **any number of classes**.
- Works well in practice for large datasets (but see cons)

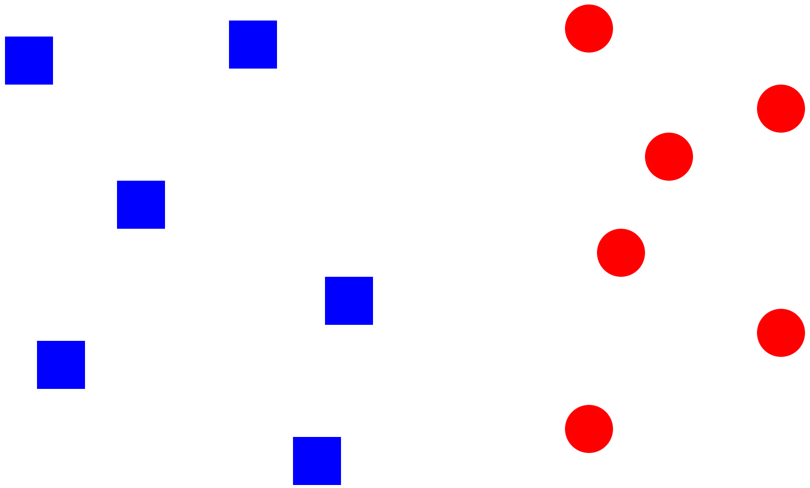
CLASSIFIERS: K-NEAREST NEIGHBOR

"Non-parametric" classifier: the entire training set is essentially the model parameters.

Cons:

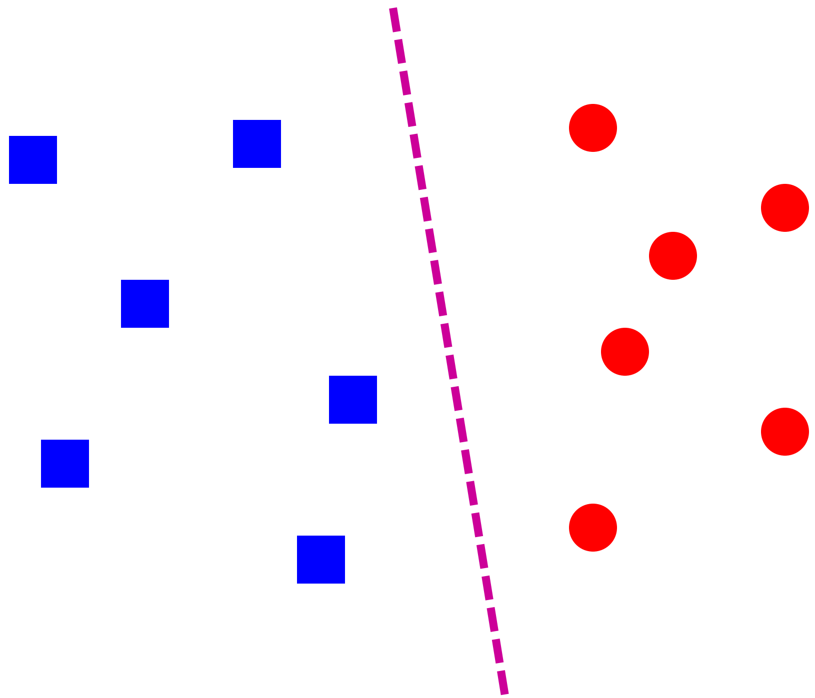
- The classifier must ***remember*** **all of the training data** and store it for future comparisons with the test data.
- This is **space inefficient** because datasets may easily be gigabytes in size.
- **Slow at test time** (need to compute distances between test example and every training example)
- Optimum value of **K is not known**.

LINEAR CLASSIFIERS – 2 CLASS PROBLEM



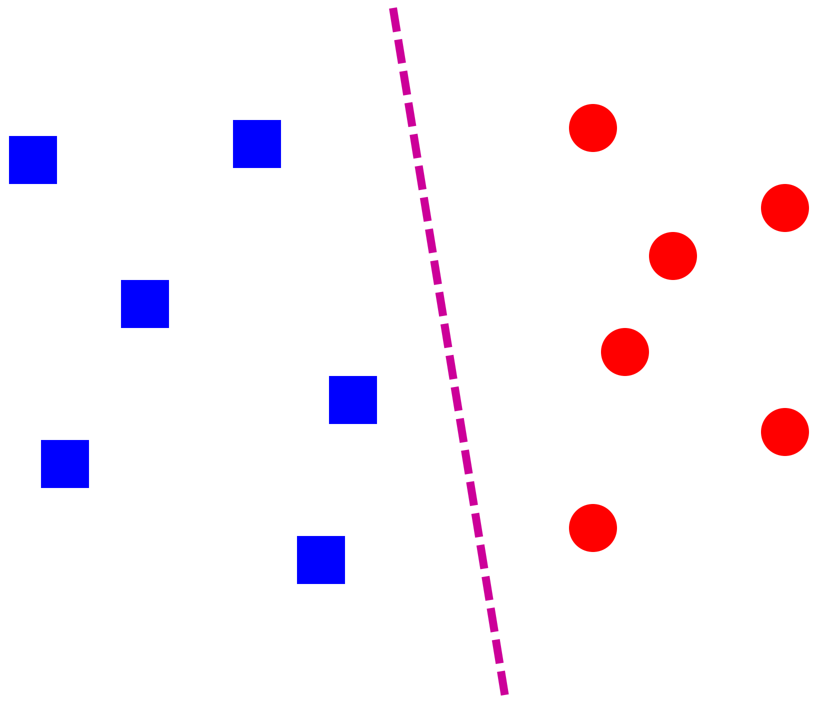
- Find a *linear function* to separate the classes:

LINEAR CLASSIFIERS – 2 CLASS PROBLEM



- Find a *linear function* to separate the classes:

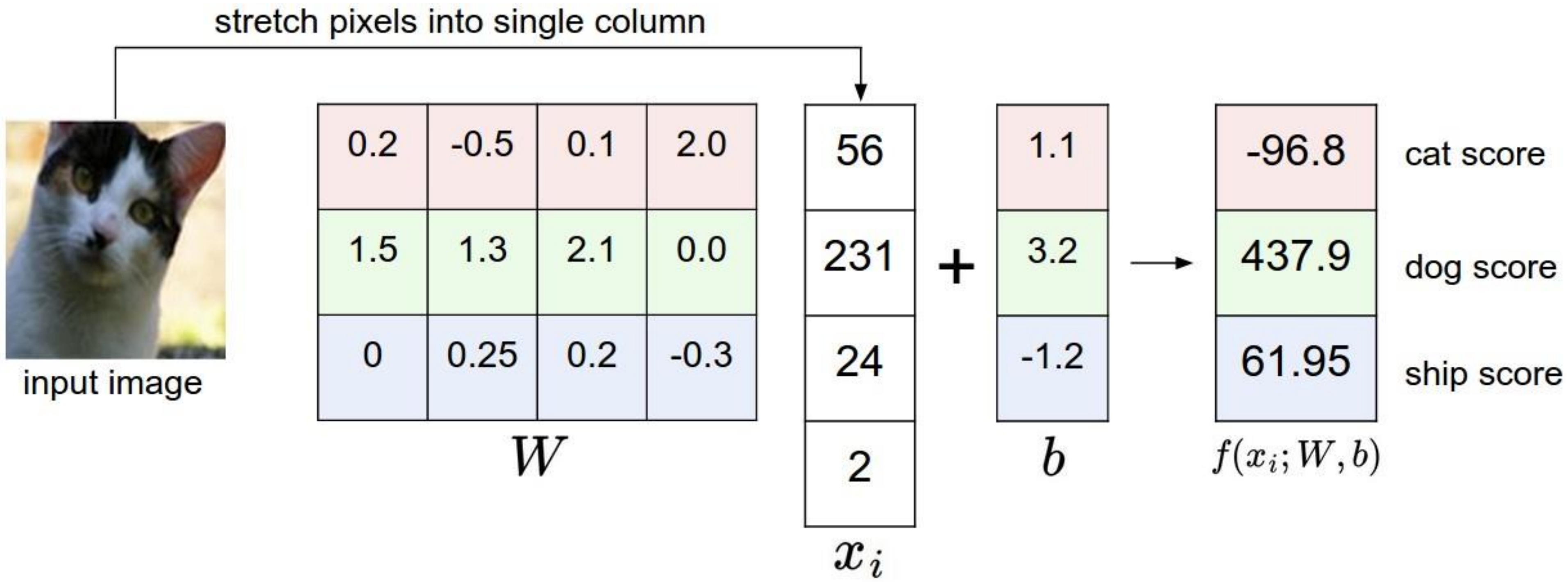
LINEAR CLASSIFIERS – 2 CLASS PROBLEM



- Find a *linear function* to separate the classes:

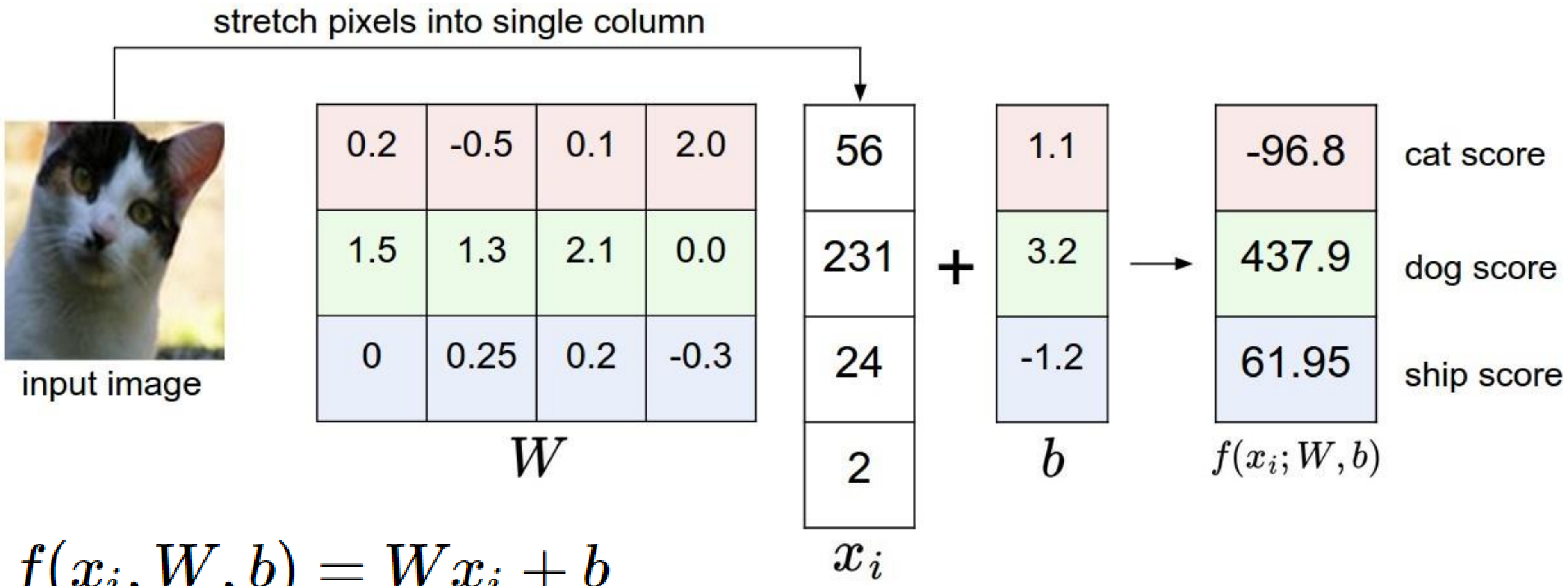
$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

LINEAR CLASSIFIERS – MORE THAN 2 CLASS



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

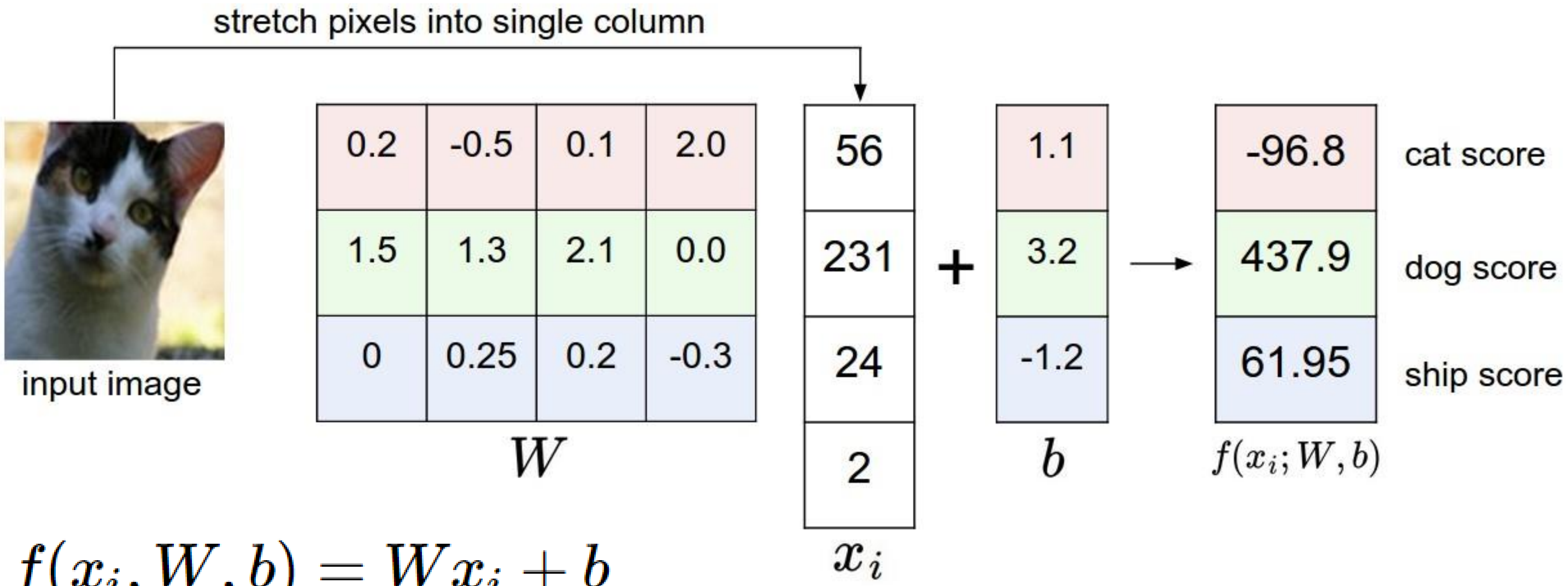
LINEAR CLASSIFIERS – MORE THAN 2 CLASS



$$f(x_i, W, b) = Wx_i + b$$

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

LINEAR CLASSIFIERS – MORE THAN 2 CLASS



$$f(x_i, W, b) = W x_i + b$$

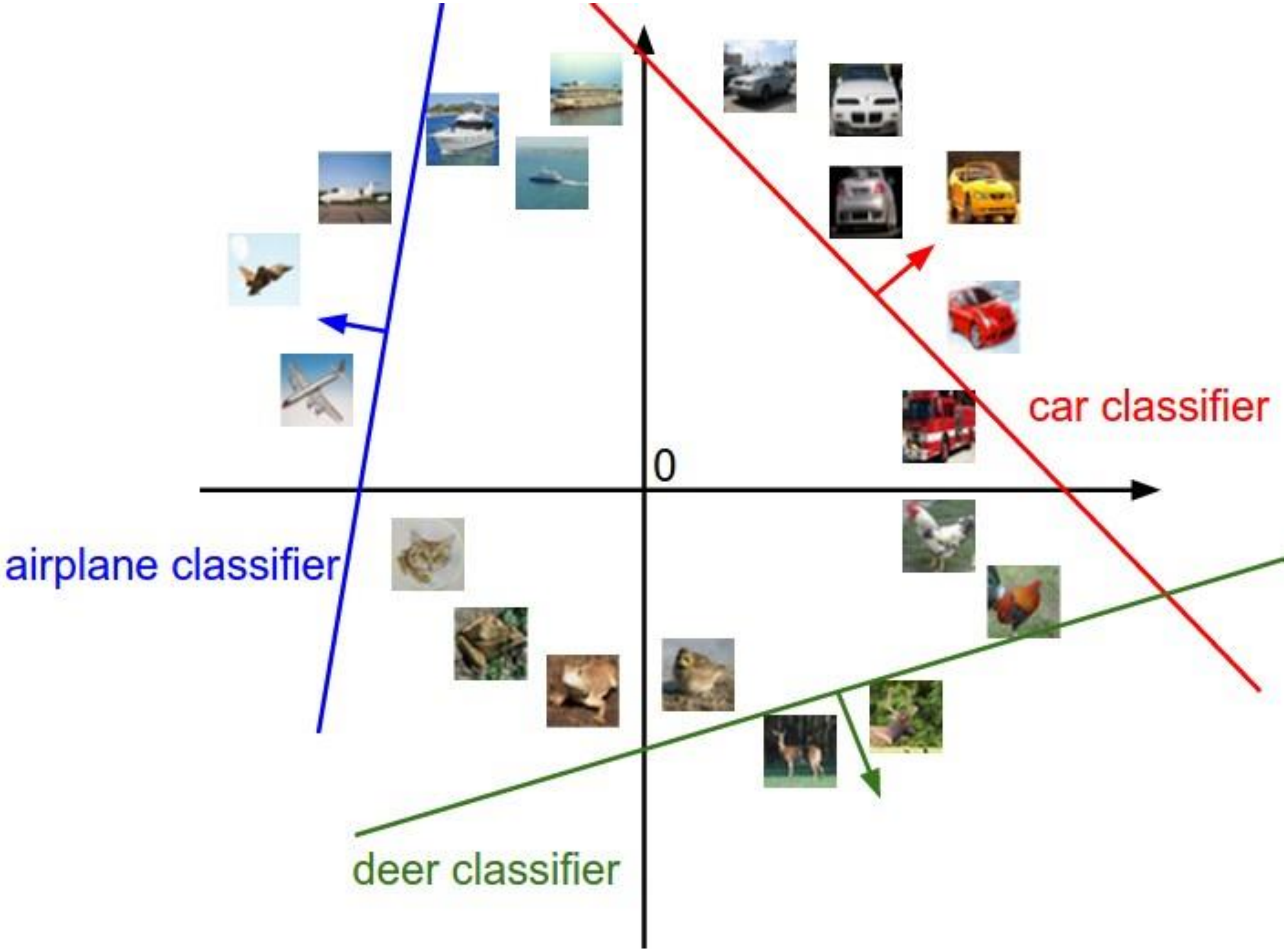
Image x_i has all of its pixels flattened out to a single column vector of shape [D x 1].

Matrix **W** (of size [K x D]), and vector **b** (of size [K x 1]) are the **parameters**.

K is the number of classes.

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

ANALOGY OF IMAGES AS HIGH-DIMENSIONAL POINTS



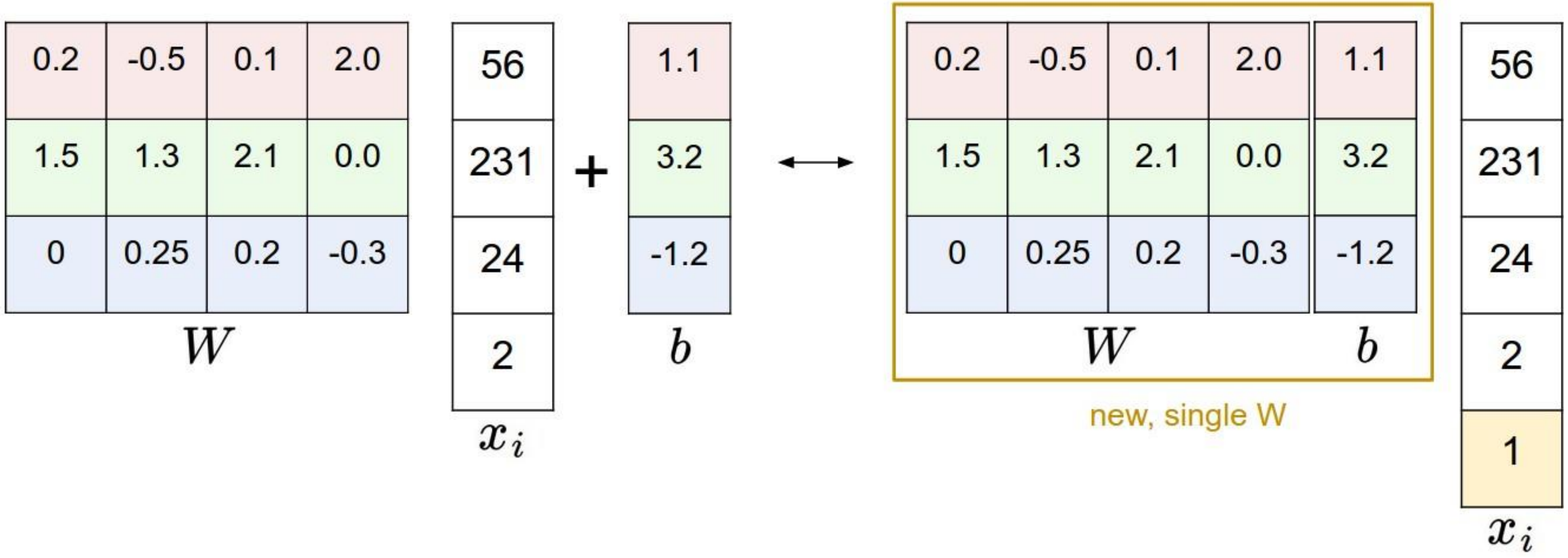
Source: cs231n, http://cs231n.github.io/linear-classify/

INTERPRETATION OF LINEAR CLASSIFIERS AS TEMPLATE MATCHING

Example learned weights at the end of learning for CIFAR-10. Note that, for example, the ship template contains a lot of blue pixels as expected. This template will therefore give a high score once it is matched against images of ships on the ocean with an inner product.

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

BIAS TRICK



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

LINEAR CLASSIFIERS

"Parametric" classifier: model defined by a small number of parameters (w, b)

Pros:

- Very fast at test time

Cons:

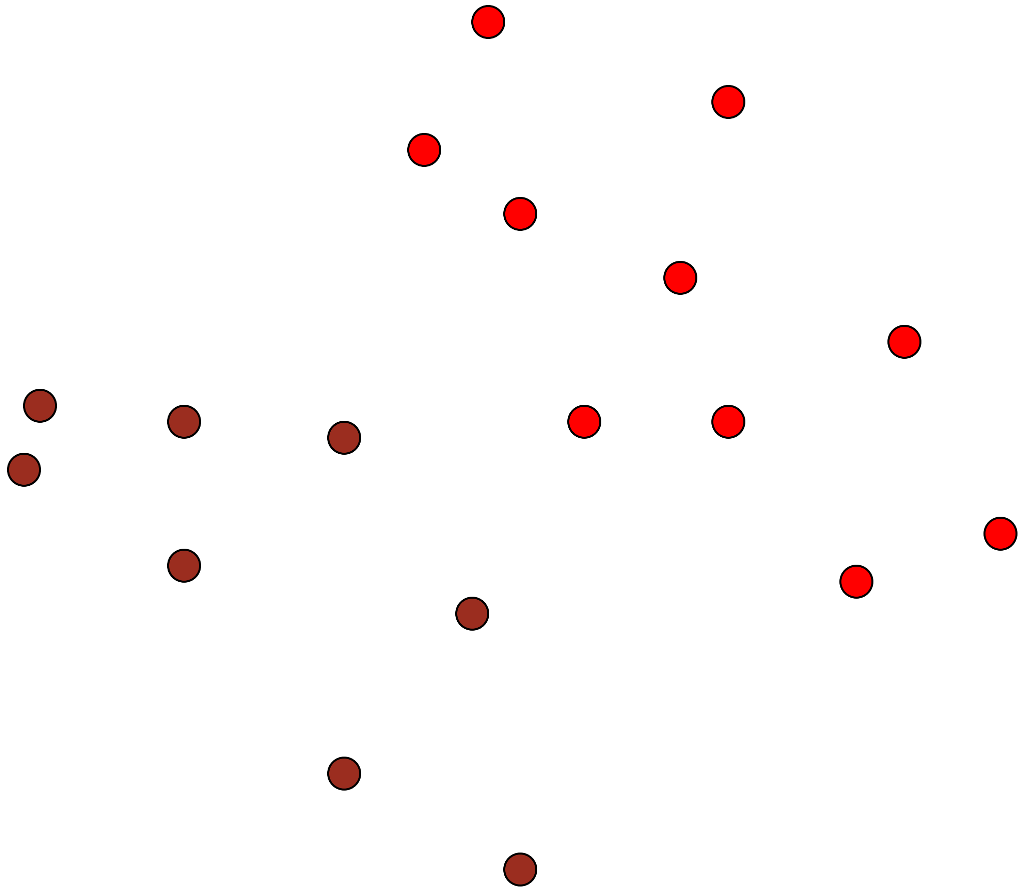
- Slow at training time: need to estimate the parameters
- Data may not be linearly separable

NEAREST NEIGHBOR VS. LINEAR CLASSIFIERS

- NN pros:
  - Simple to implement
  - Decision boundaries not necessarily linear
  - Works for any number of classes
  - *Nonparametric* method
- NN cons:
  - Need good distance function
  - Slow at test time, Memory in-efficient
- Linear pros:
  - Low-dimensional *parametric* representation
  - Very fast at test time
- Linear cons:
  - How to train the linear function?
  - What if data is not linearly separable?

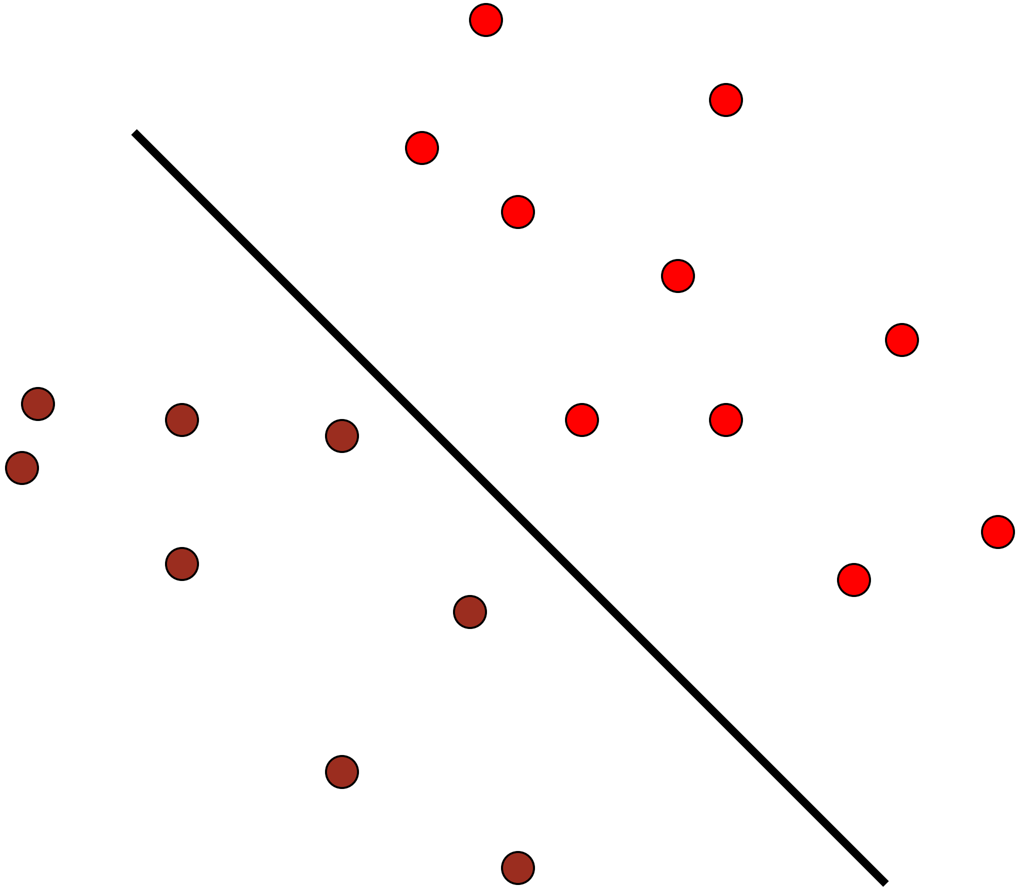
SUPPORT VECTOR MACHINES

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



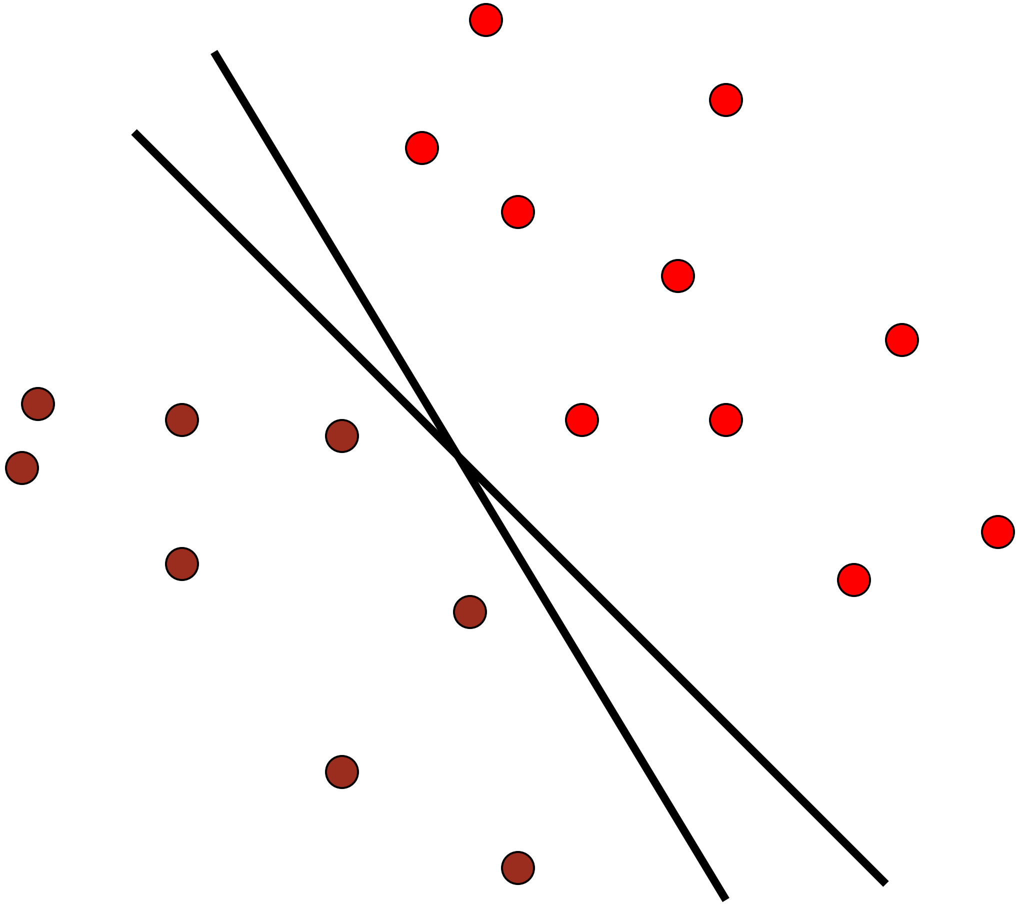
SUPPORT VECTOR MACHINES

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



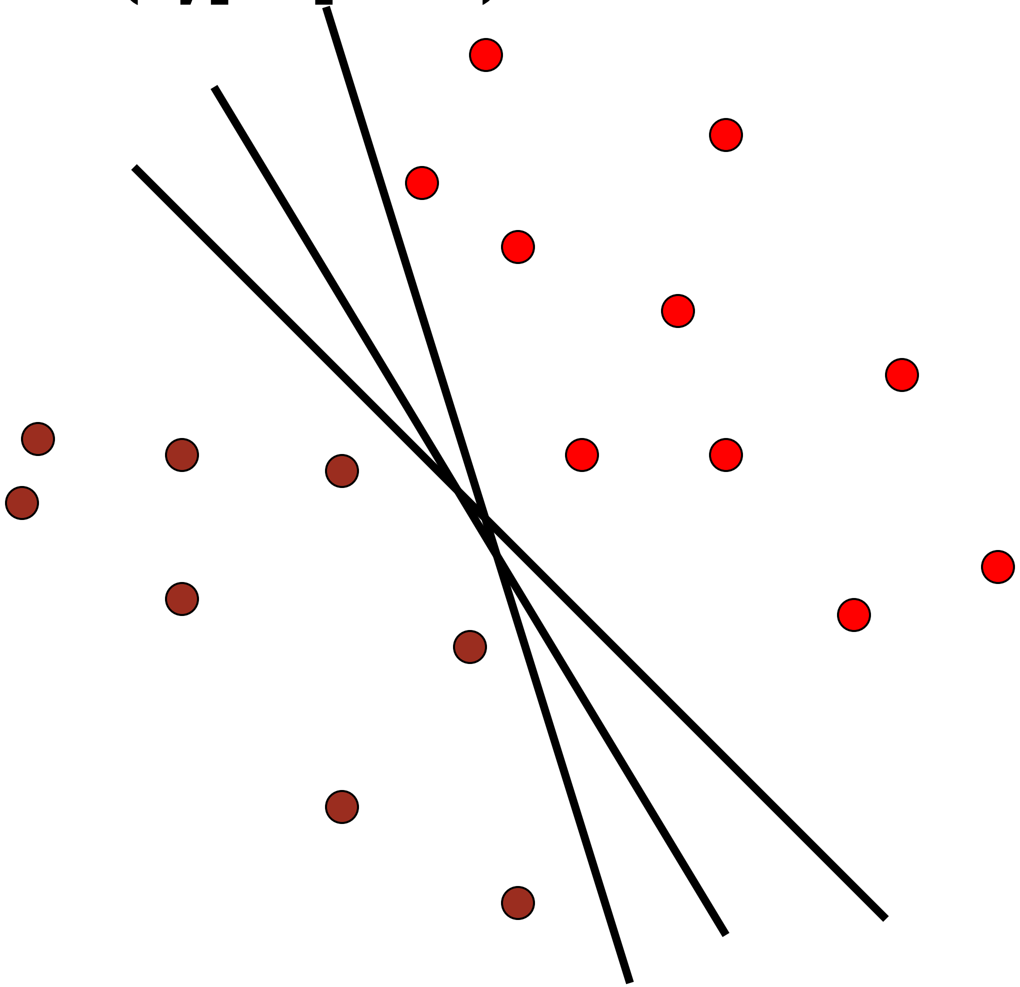
SUPPORT VECTOR MACHINES

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



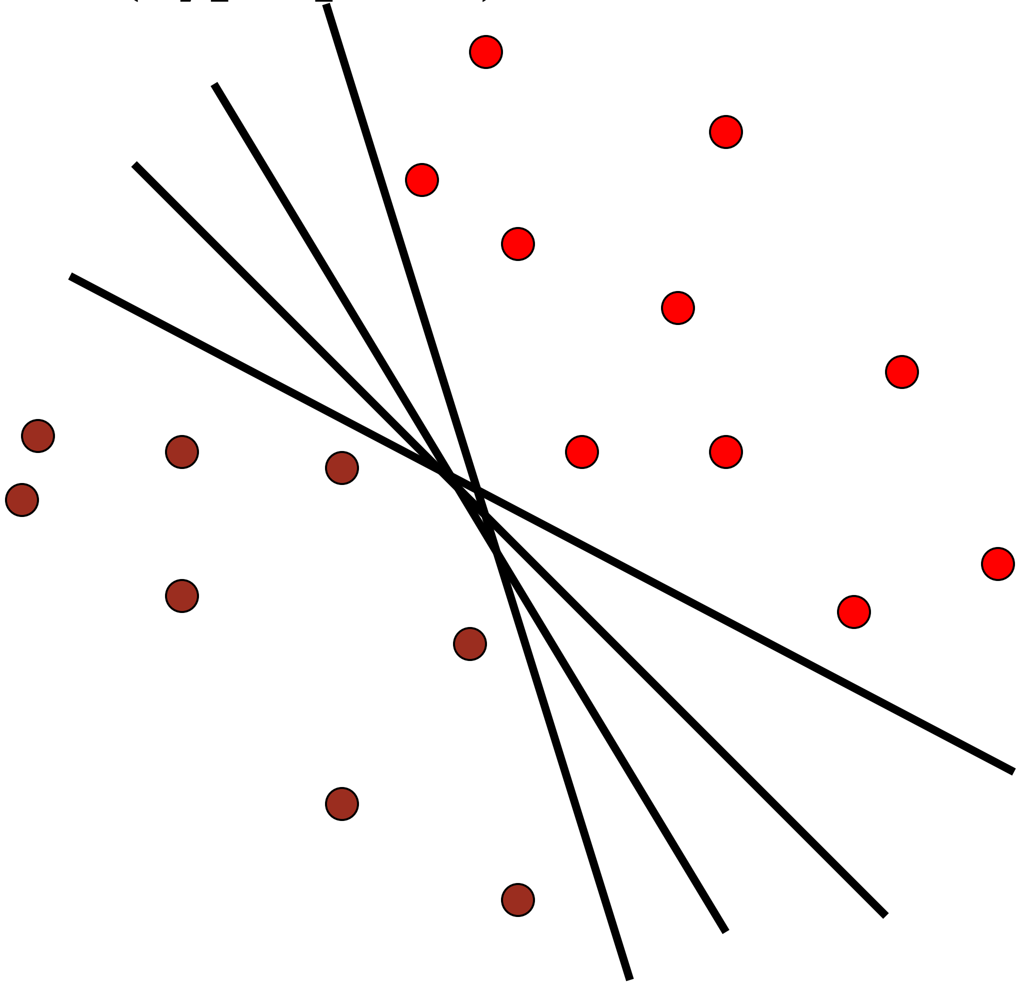
SUPPORT VECTOR MACHINES

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



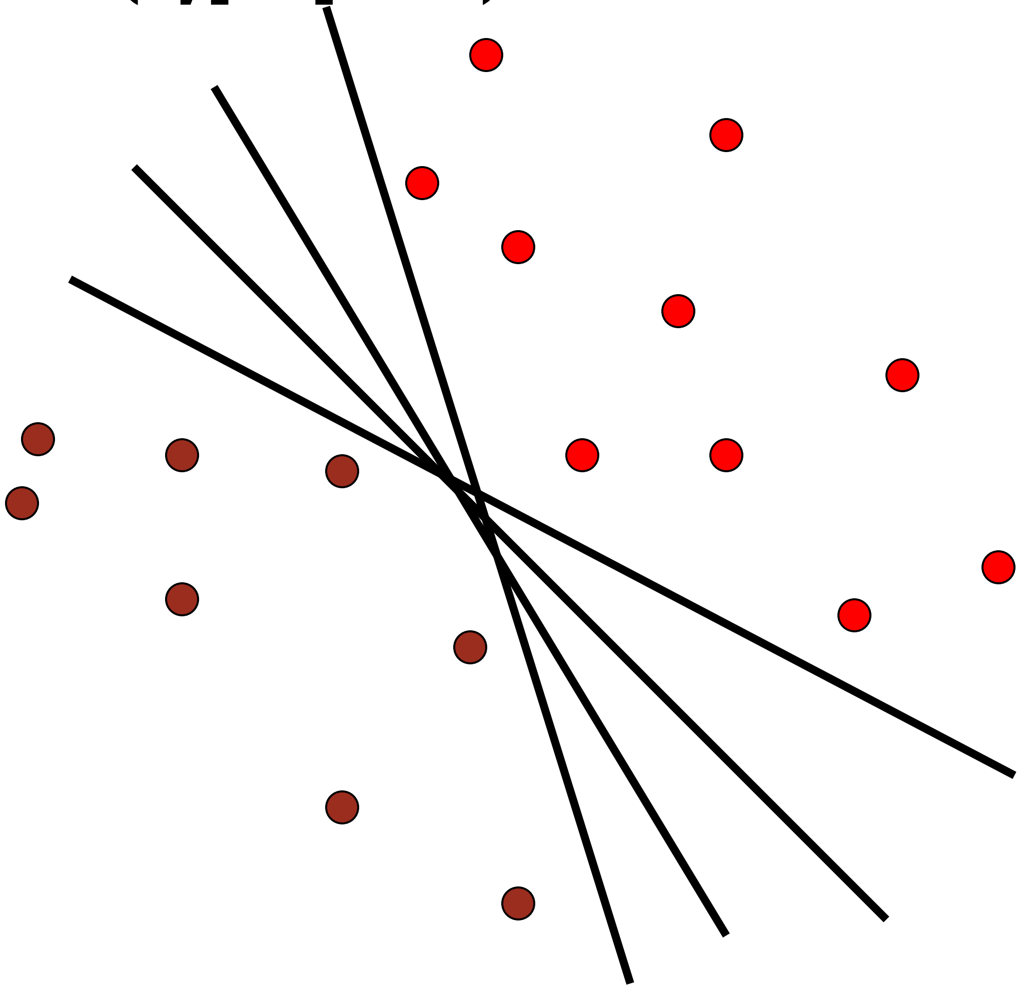
SUPPORT VECTOR MACHINES

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



SUPPORT VECTOR MACHINES

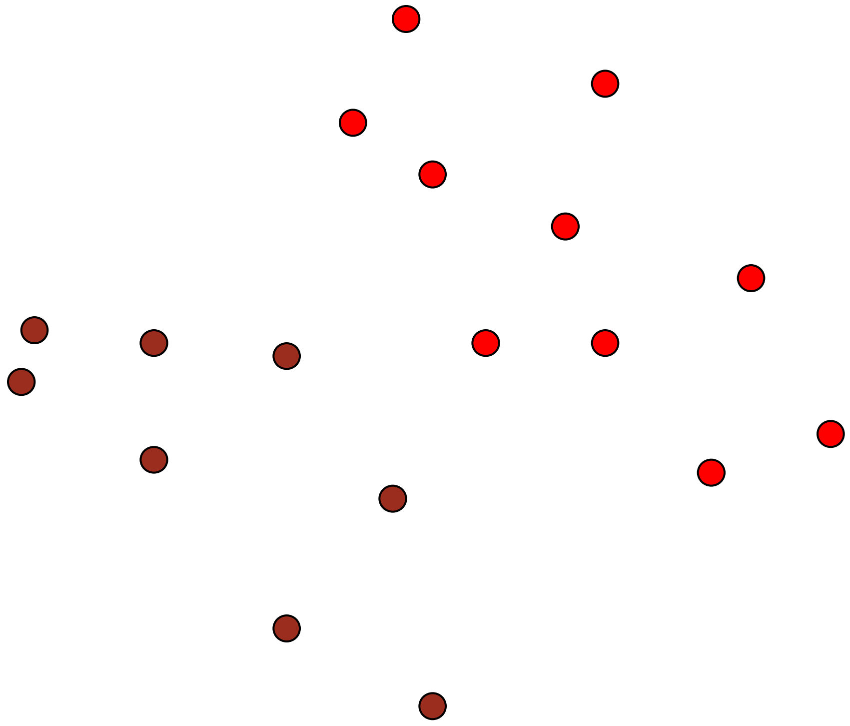
- When the data is linearly separable, there may be more than one separator (hyperplane)



Which separator is best?

SUPPORT VECTOR MACHINES

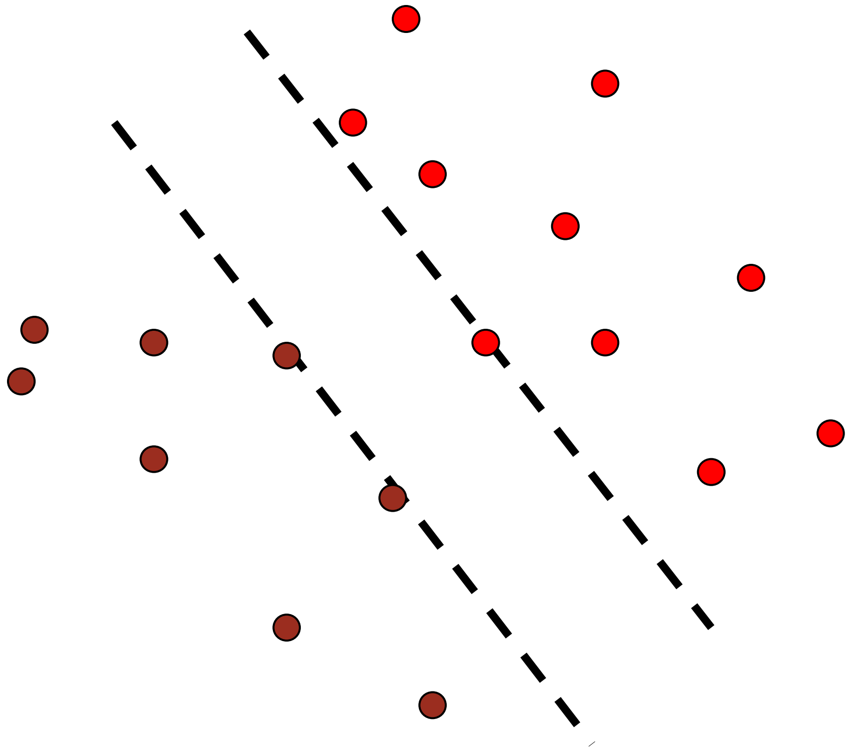
- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

SUPPORT VECTOR MACHINES

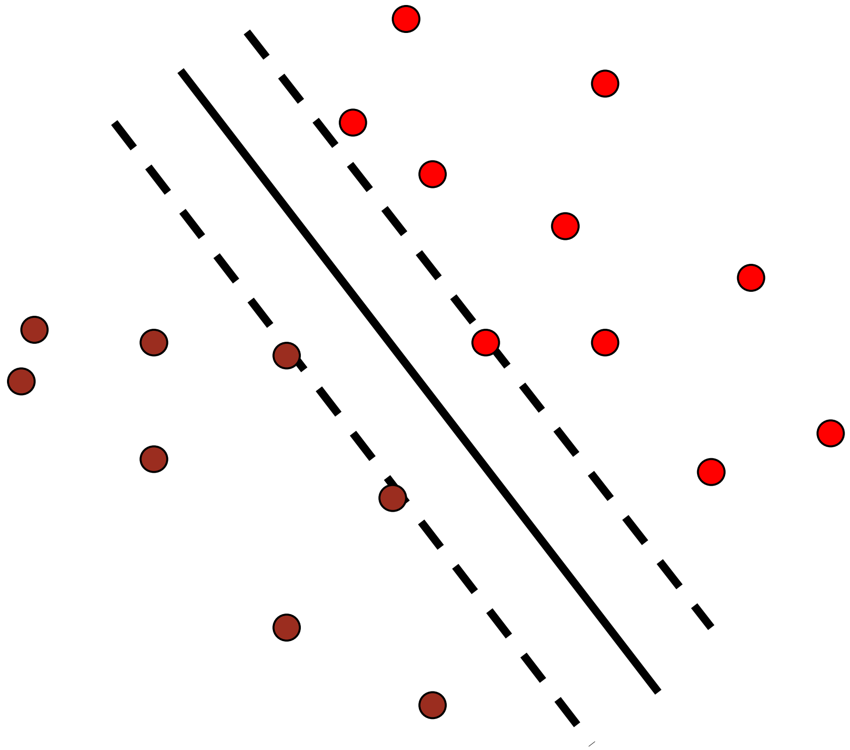
- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

SUPPORT VECTOR MACHINES

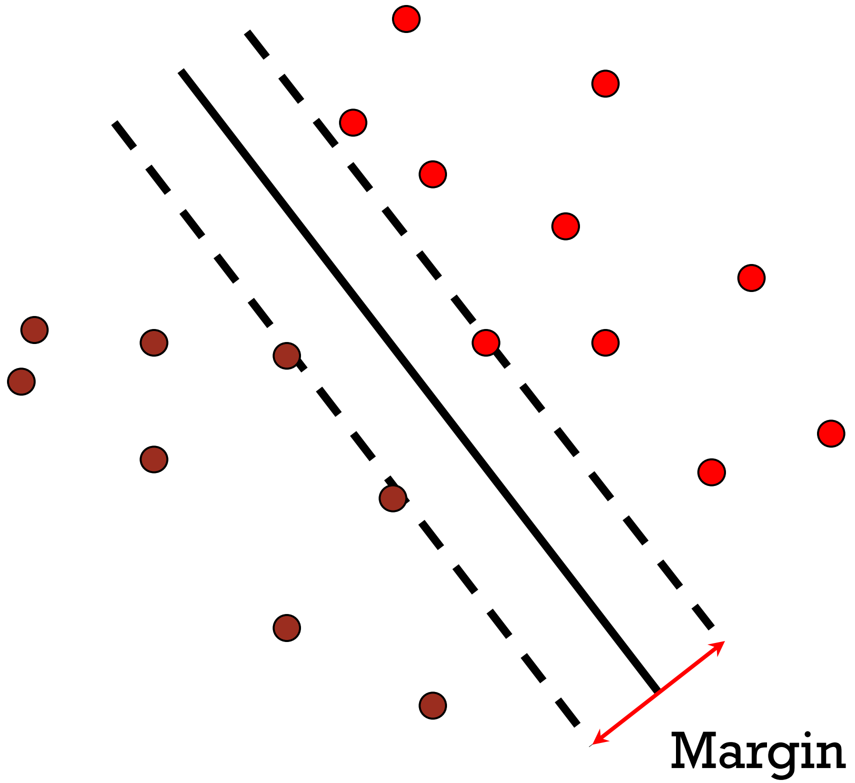
- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

SUPPORT VECTOR MACHINES

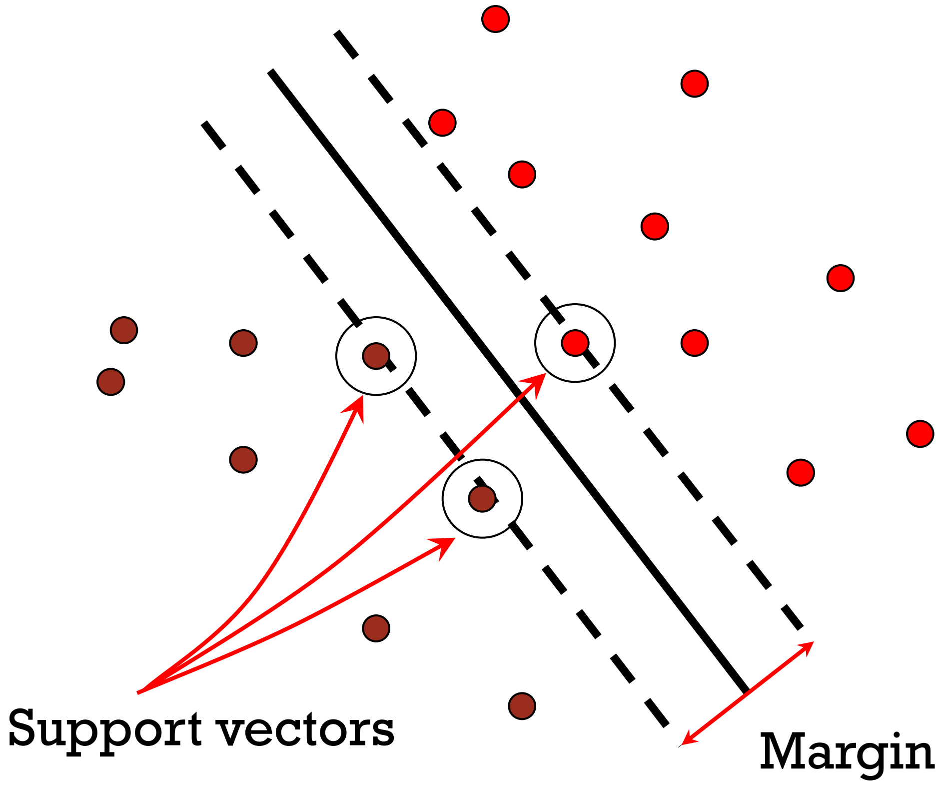
- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

SUPPORT VECTOR MACHINES

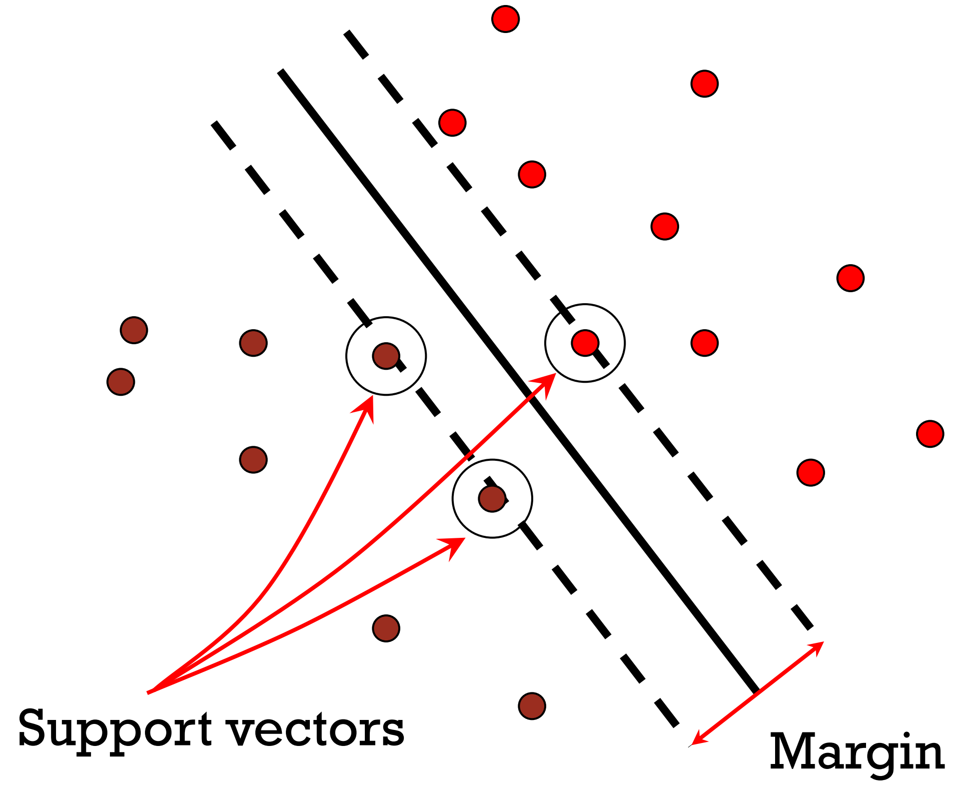
- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

SUPPORT VECTOR MACHINES

- Find hyperplane that maximizes the *margin* between the positive and negative examples



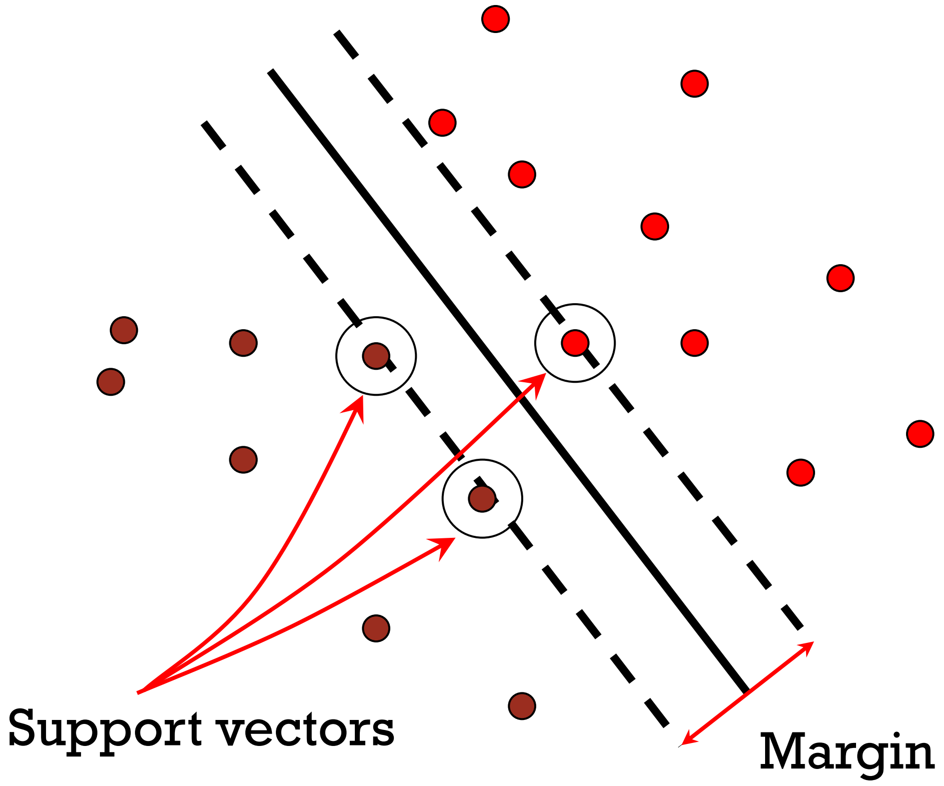
$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

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

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

SUPPORT VECTOR MACHINES

- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

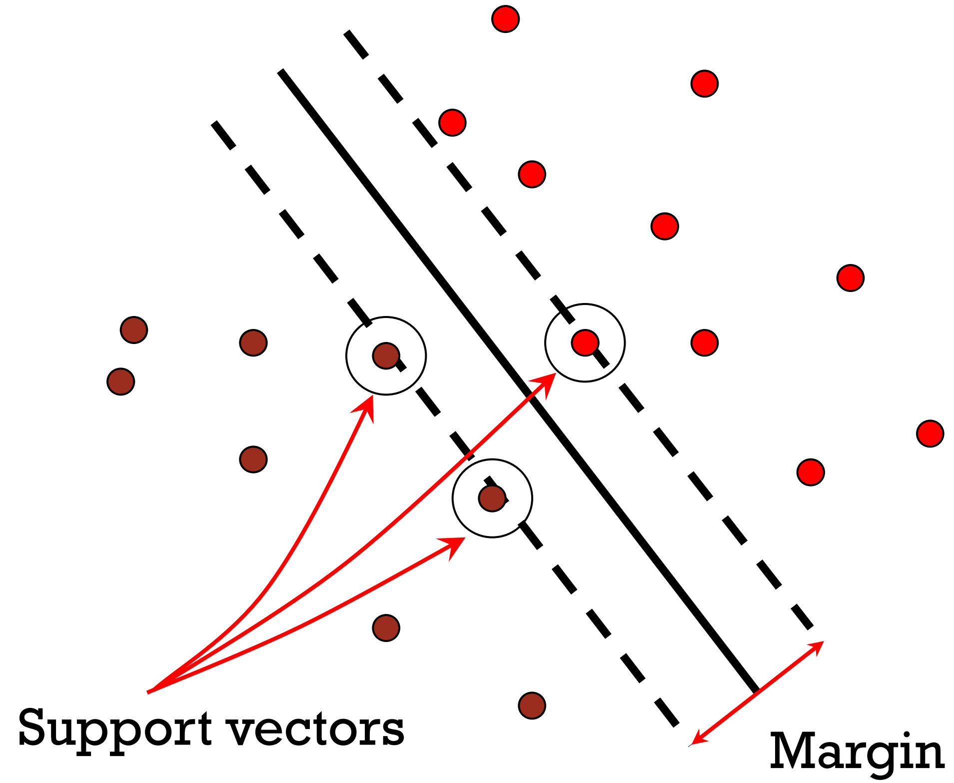
$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

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

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

SUPPORT VECTOR MACHINES

- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

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

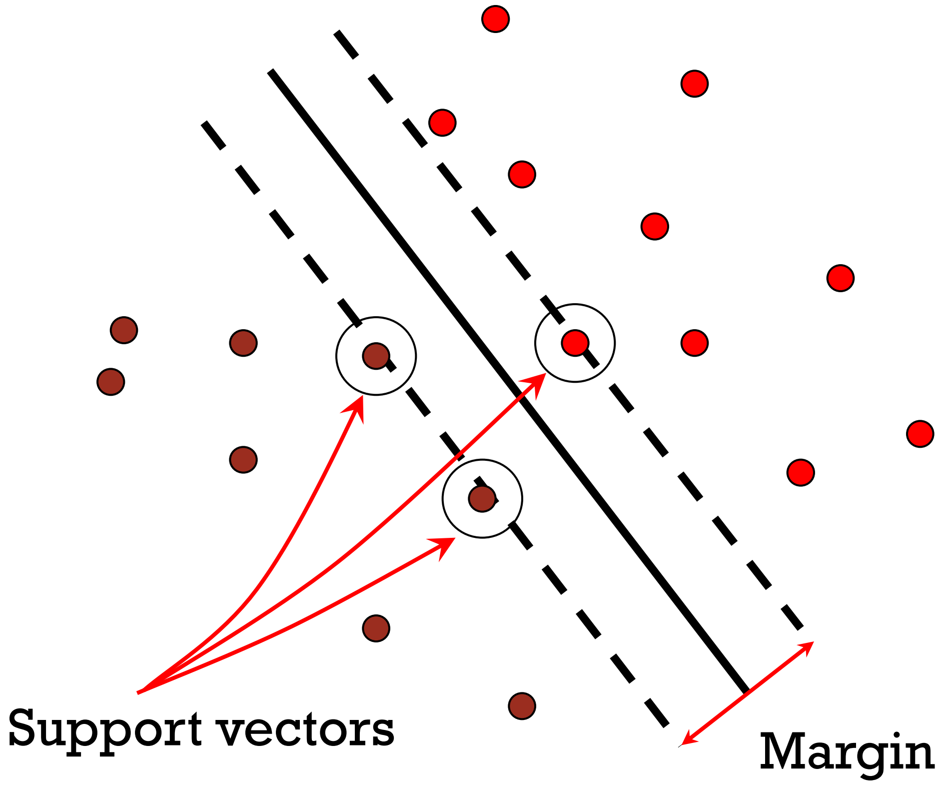
For support vectors, $\quad \mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

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

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

SUPPORT VECTOR MACHINES

- Find hyperplane that maximizes the *margin* between the positive and negative examples



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

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

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

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

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:

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

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

- *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

FINDING THE MAXIMUM MARGIN HYPERPLANE

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

learned weight

Support vector

- 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:

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

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

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:

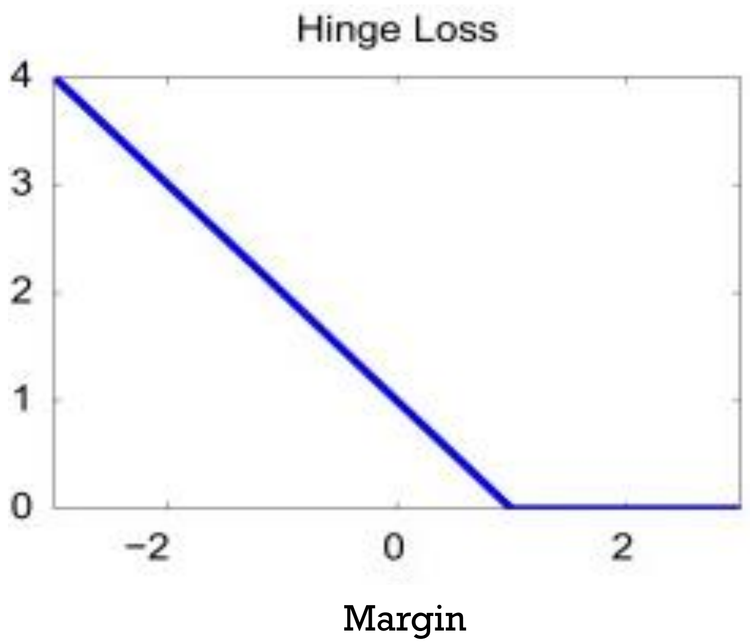
$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b))$$

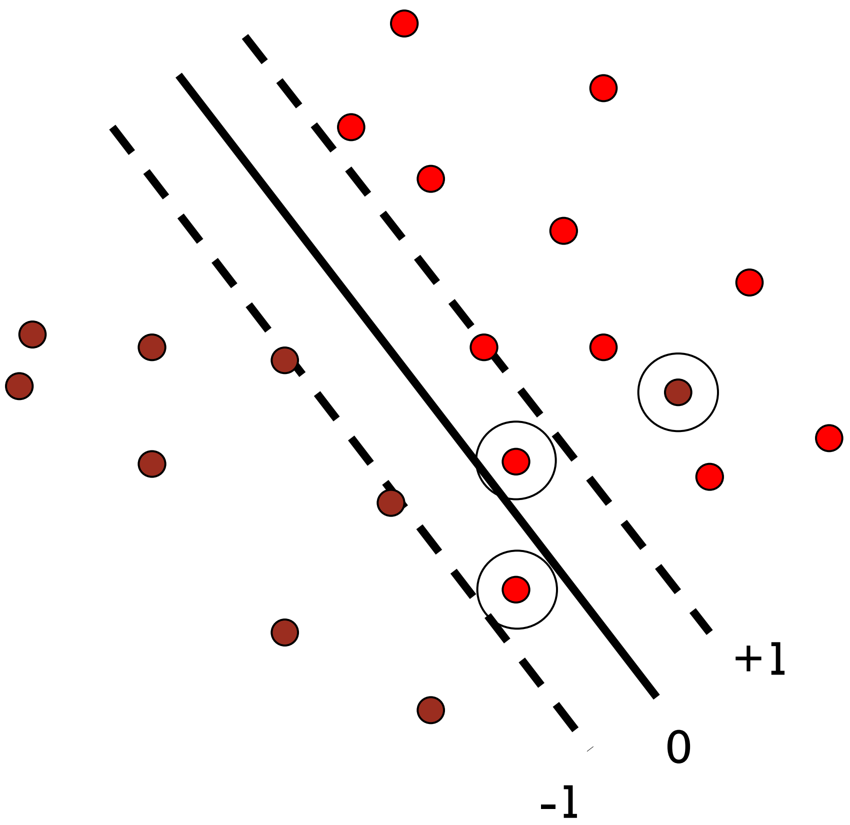
Maximize margin

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

NONLINEAR SVMS

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

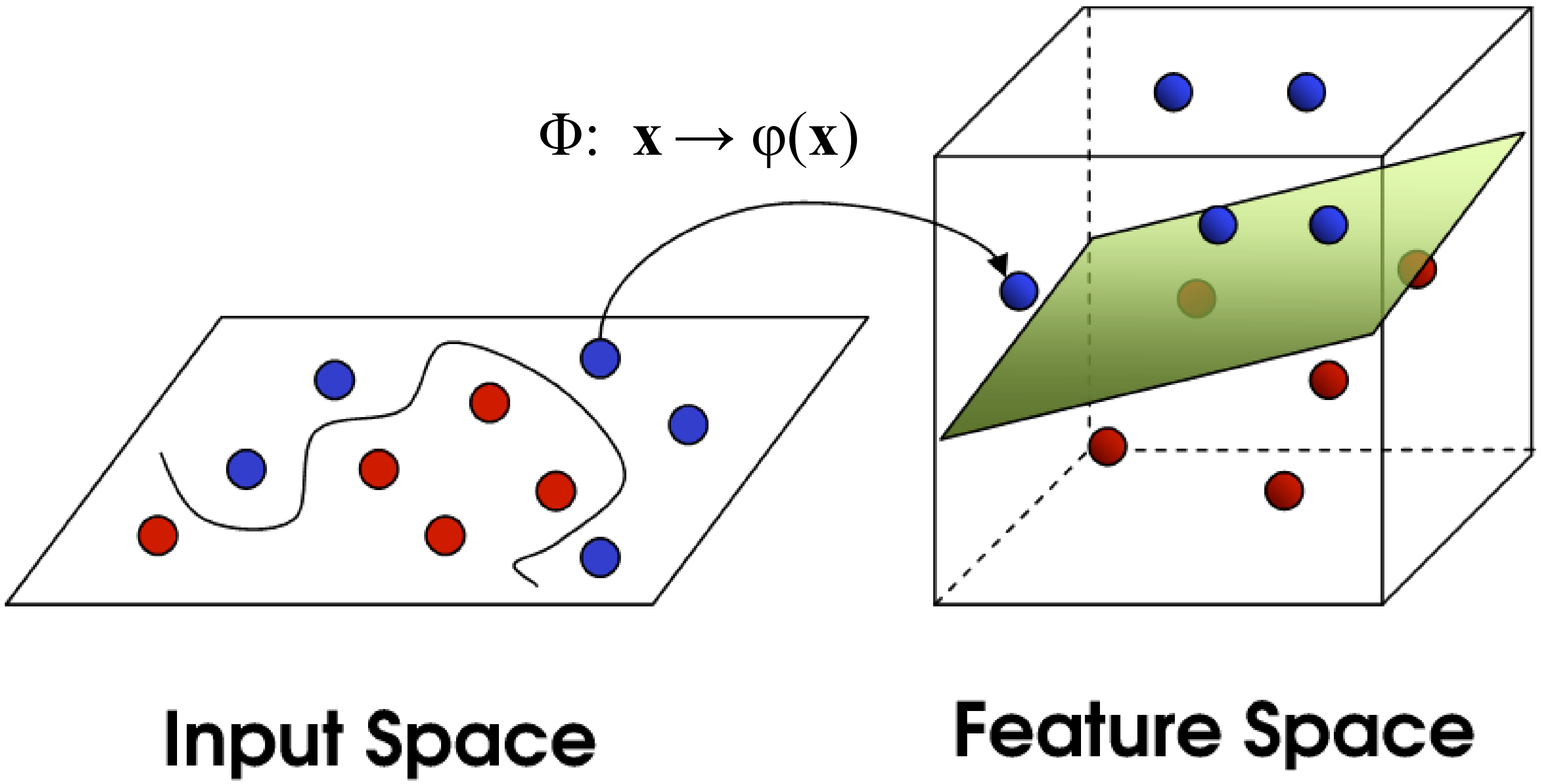
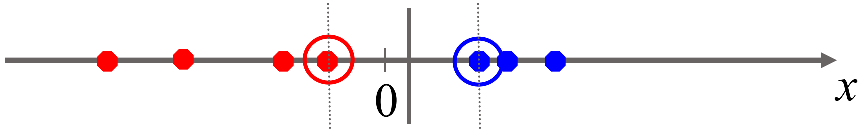


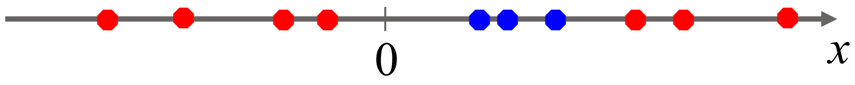
Image source

NONLINEAR SVMS

- Linearly separable dataset in 1D:



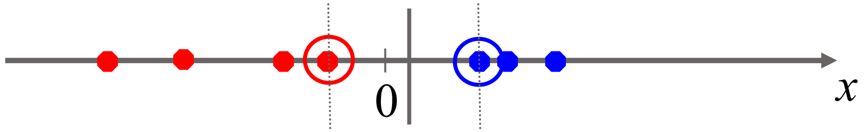
- Non-linearly separable dataset in 1D:



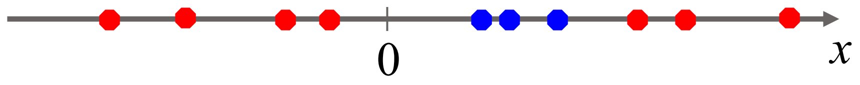
Slide credit: Andrew Moore

NONLINEAR SVMS

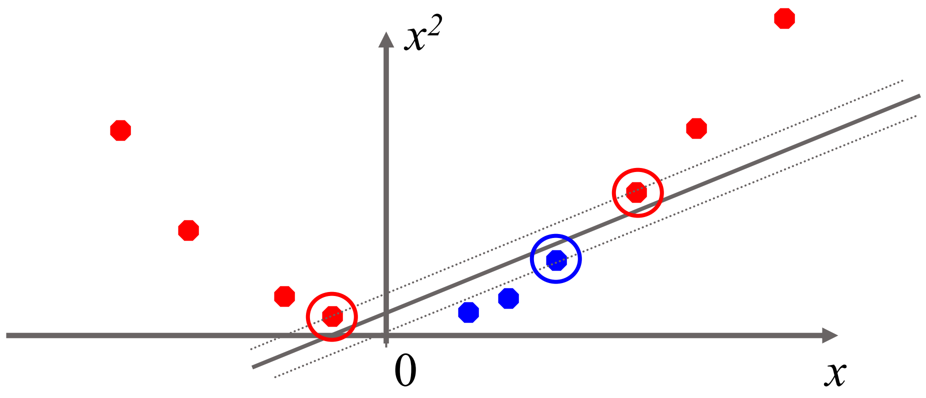
- Linearly separable dataset in 1D:



- Non-linearly separable dataset in 1D:



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



Slide credit: Andrew Moore

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 $\varphi(\mathbf{x})$, define a kernel function K such that

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

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

EXAMPLE

2-dimensional vectors x=[x_1 x_2];

let $K(\mathrm{x}_i,\mathrm{x}_j)=(1 + \mathrm{x}_i^T\mathrm{x}_j)^2$

EXAMPLE

2-dimensional vectors x=[x_1 x_2];

let $K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^{\mathrm{T}}\mathrm{x_j})^2$

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

EXAMPLE

2-dimensional vectors x=[x_1 x_2];

let $K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^{\mathrm{T}}\mathrm{x_j})^2$

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

$K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^{\mathrm{T}}\mathrm{x_j})^2$,

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

EXAMPLE

2-dimensional vectors x=[x_1 x_2];

let $K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^T\mathrm{x_j})^2$

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

$K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^T\mathrm{x_j})^2$,

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

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

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

EXAMPLE

2-dimensional vectors x=[x_1 x_2];

let $K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^T\mathrm{x_j})^2$

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

$$K(\mathrm{x_i},\mathrm{x_j})=(1 + \mathrm{x_i}^T\mathrm{x_j})^2,$$

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

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

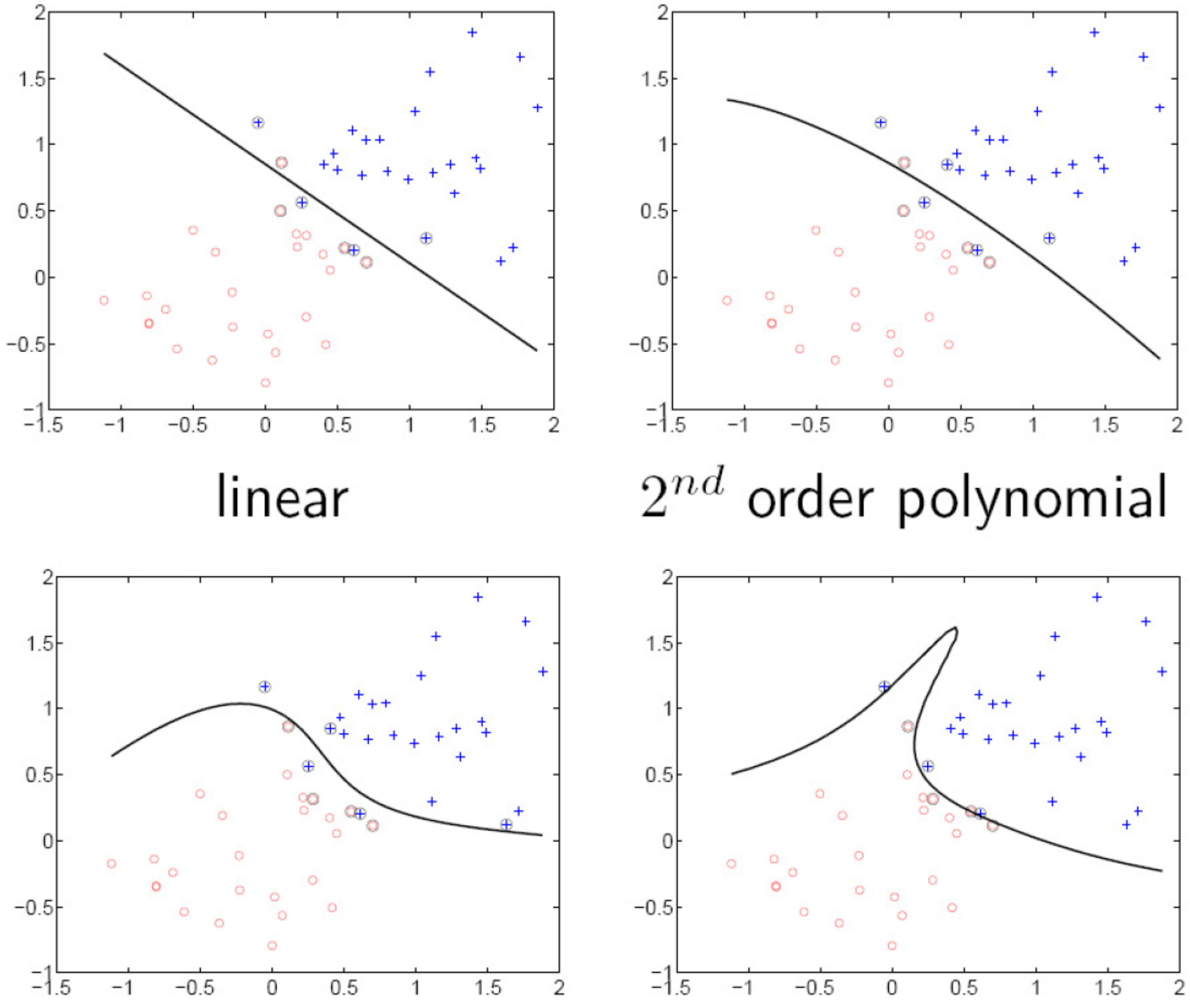
$$[1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$$

$$= \varphi(\mathrm{x_i})^T\varphi(\mathrm{x_j}),$$

where $\varphi(\mathrm{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

POLYNOMIAL KERNEL

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



linear

2^{nd} order polynomial

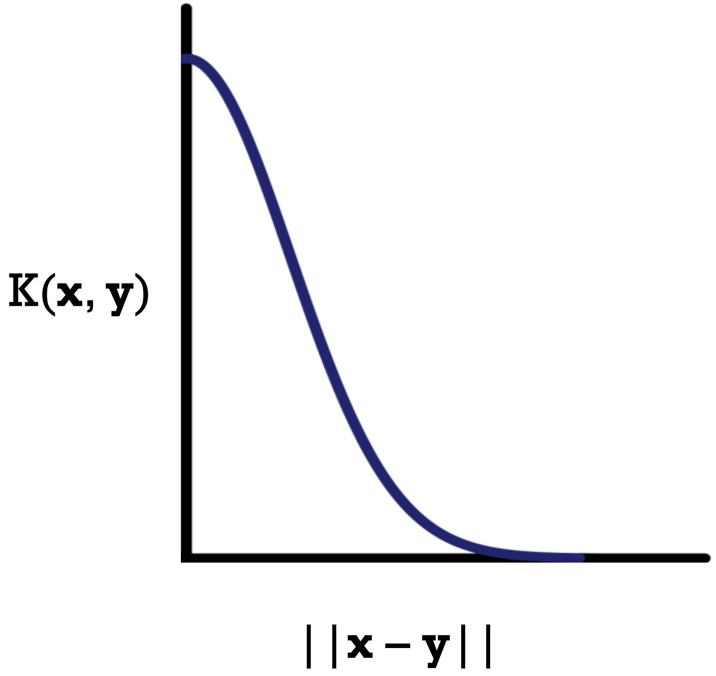
4^{th} order polynomial

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:

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

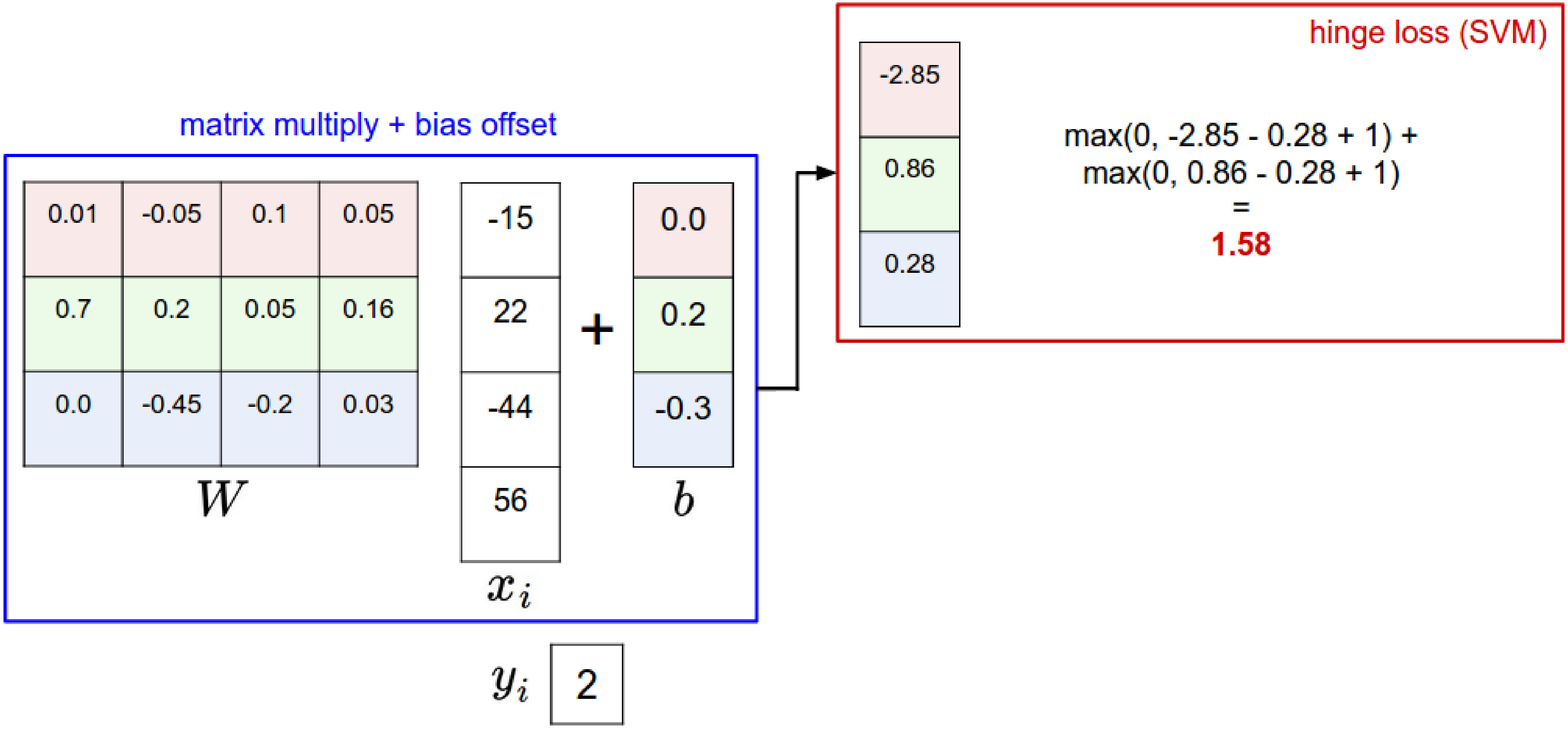
- Hinge Loss:

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

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

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:

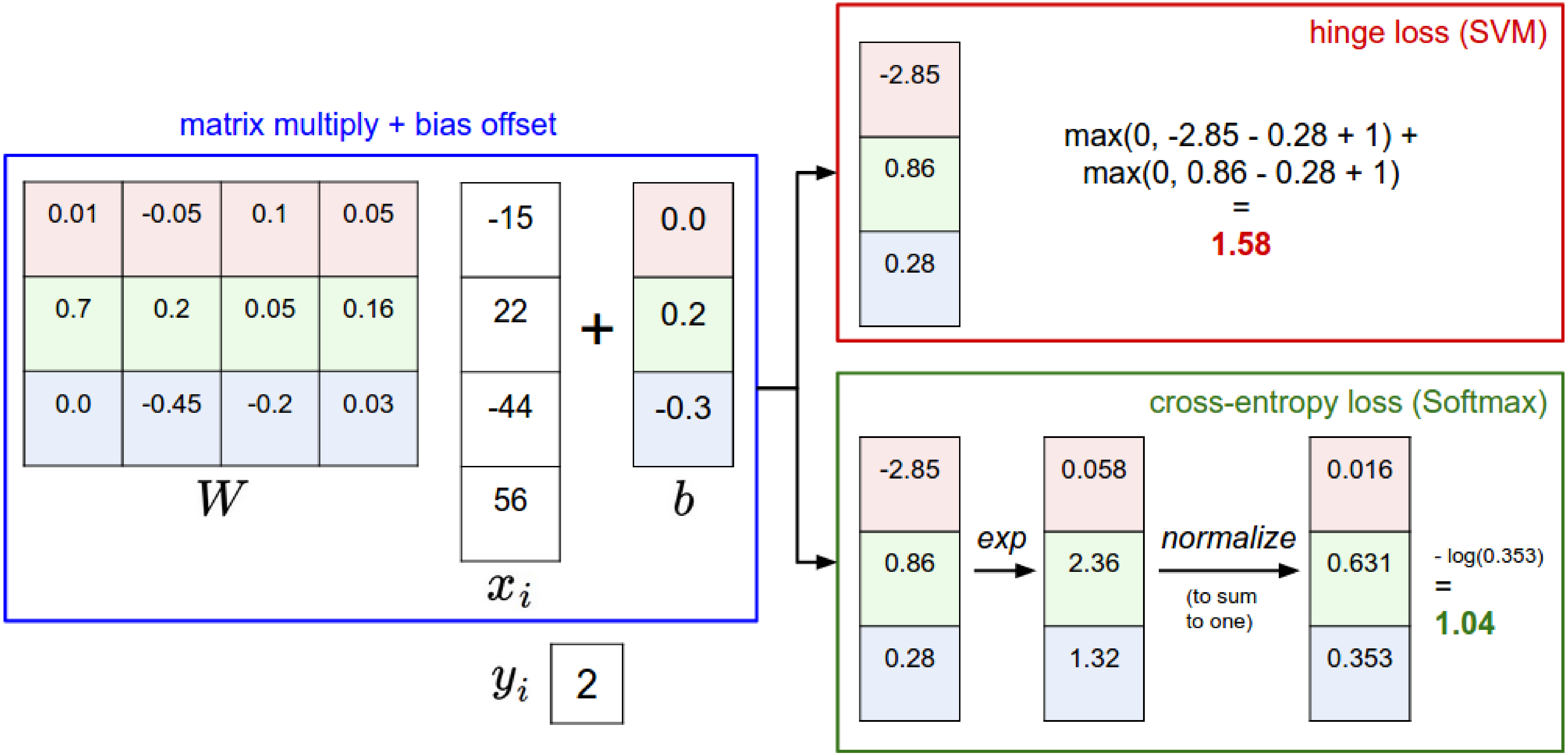
$$L_i = -log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) = -s_{y_i} + log\sum_j e^{s_j}$$

- 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/

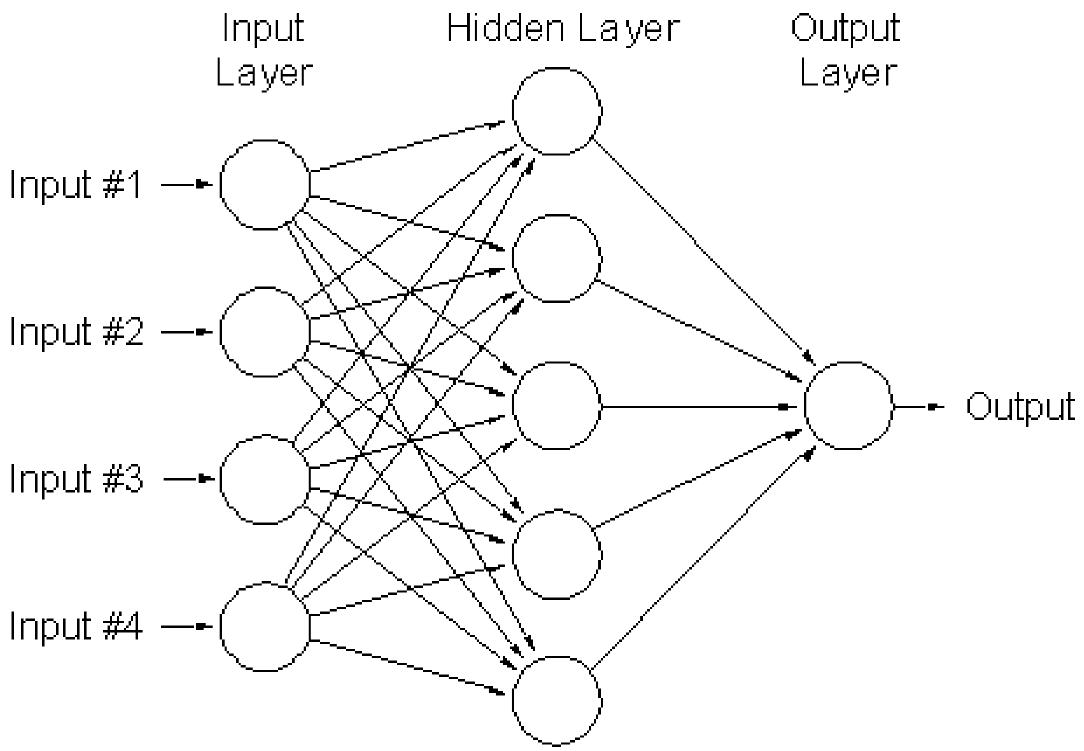
ACKNOWLEDGEMENT

Thanks to the following courses and corresponding researchers for making their teaching/research material online

- Convolutional Neural Networks for Visual Recognition, Stanford University
- Deep Learning, Stanford University
- Introduction to Deep Learning, University of Illinois at Urbana-Champaign
- Introduction to Deep Learning, Carnegie Mellon University
- Natural Language Processing with Deep Learning, Stanford University
- And Many More Publicly Available Resources ……

NEXT LECTURE

Neural Networks



Questions?