Tutorial outline

- Overview (this)
- Image representation (60 mins, 9:15 - 10:30)
  - motivation, local features, global features, break
- Learning (90 mins, 10:30 - 12:30)
  - discriminative models, tea-break, generative models, break
- Object detection and recognition (90 mins, 12:30 - 2:00)
  - Dalal & Triggs, lunch-break, PASCAL challenge, *poselets* and their applications, tea-break
- Cross-modal search (60 mins, 2:30 - 3:30)

lunch-break 60 mins, break 15 mins, tea-break 20-30 mins

ICVGIP 2012, IIT Bombay

The eight Indian Conference on Computer Vision, Graphics and Image Processing

Tutorial

SVMs: linear, non-linear and additive

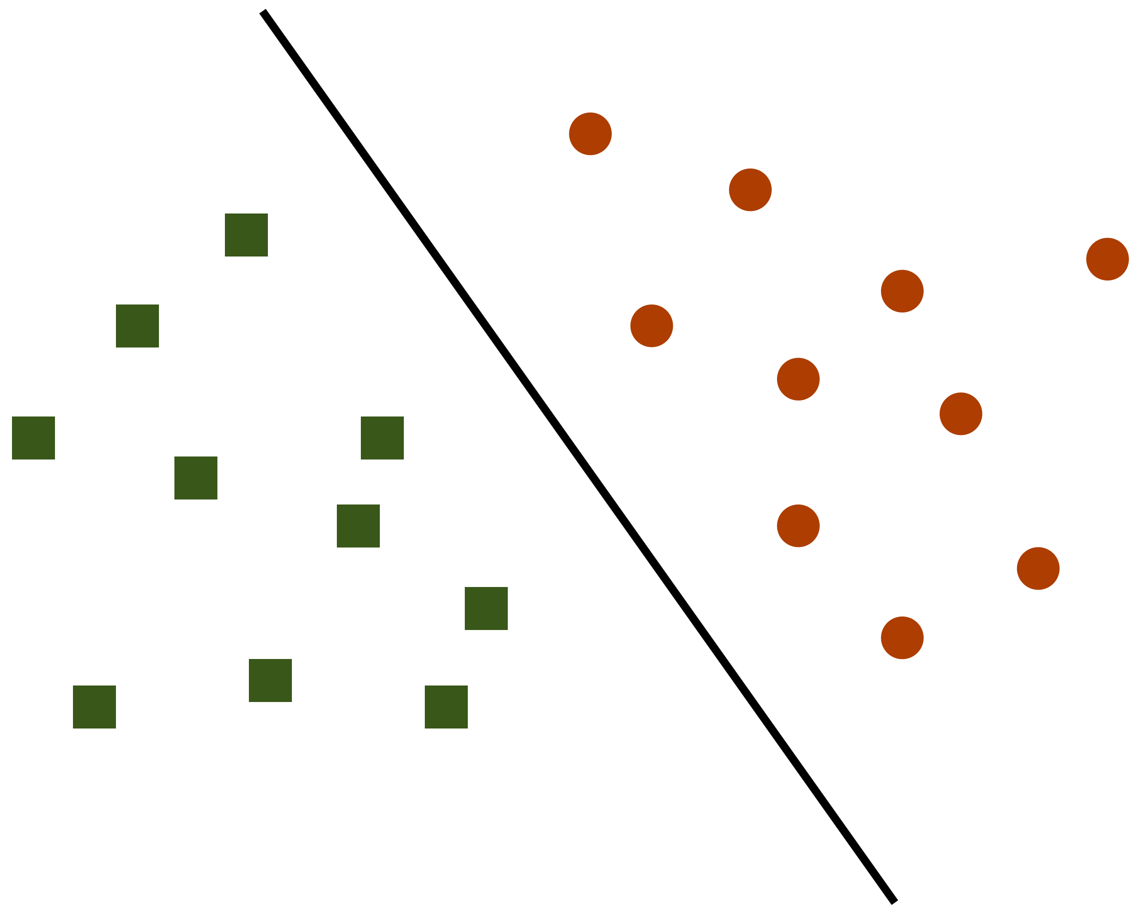
Subhransu Maji
Toyota Technological Institute at Chicago

Overview

- Linear SVMs
  - margins, learning, representer theorem
- Non-linear kernel SVMs
  - non-linear kernels, learning, classification complexity
- Kernels in computer vision
  - Examples
  - Histogram intersection kernel
  - Efficient evaluation and experiments
  - Efficient training using explicit embeddings
- Conclusions

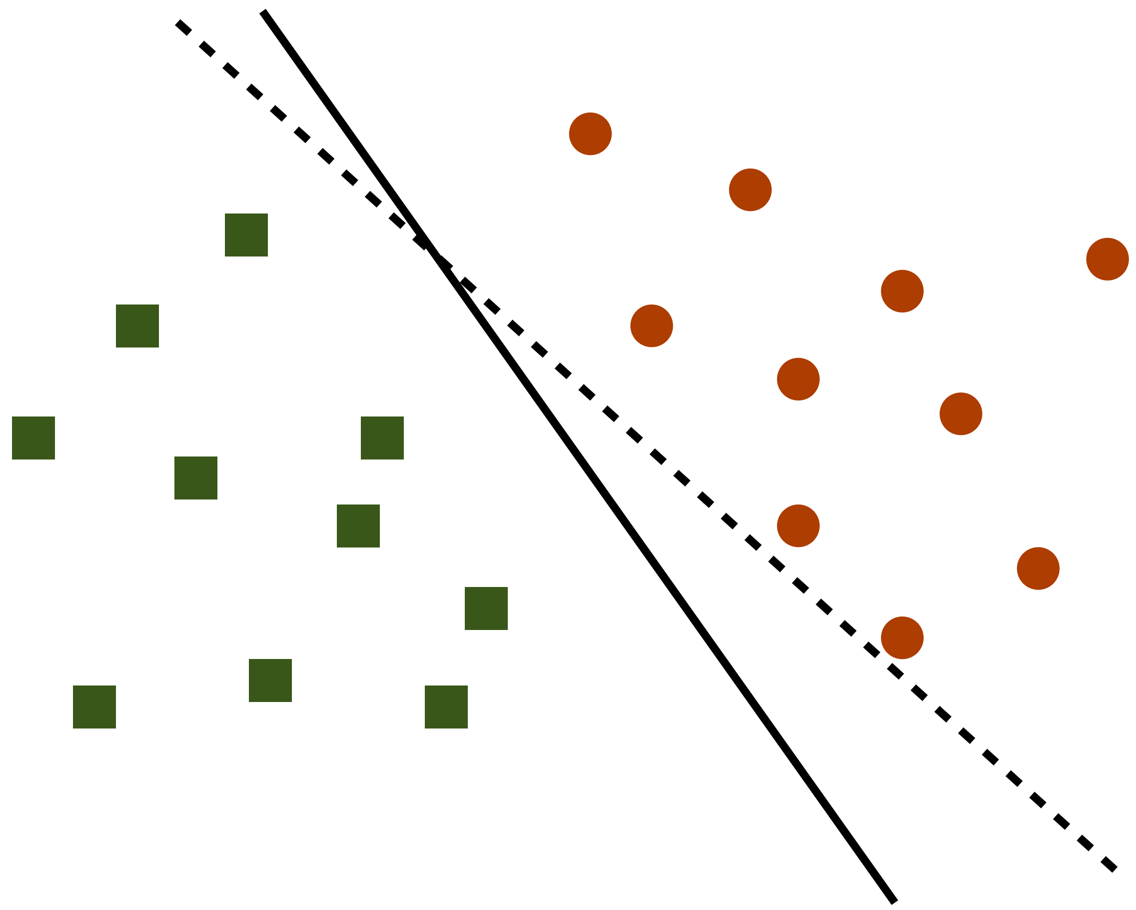
Linear separators

binary classification with a hyperplane



Linear separators

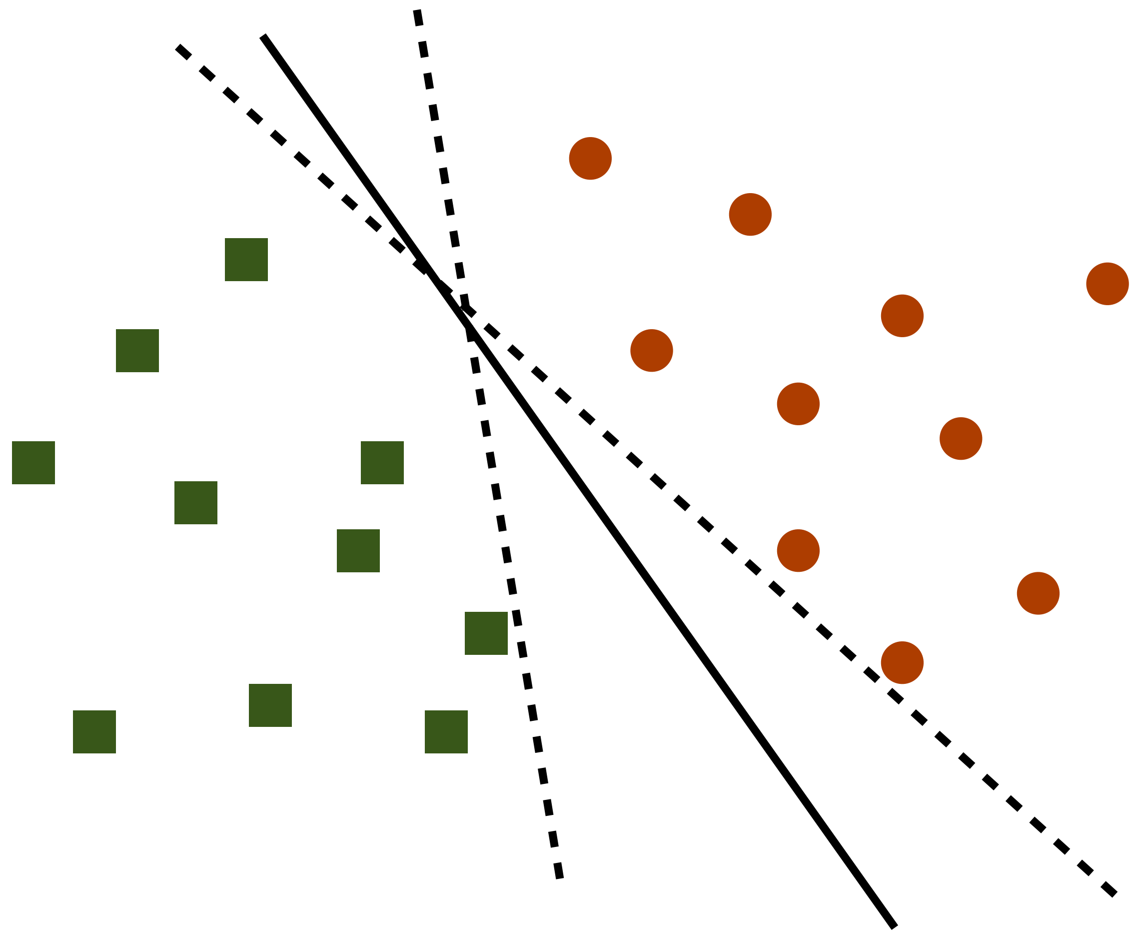
binary classification with a hyperplane



many possible solutions

Linear separators

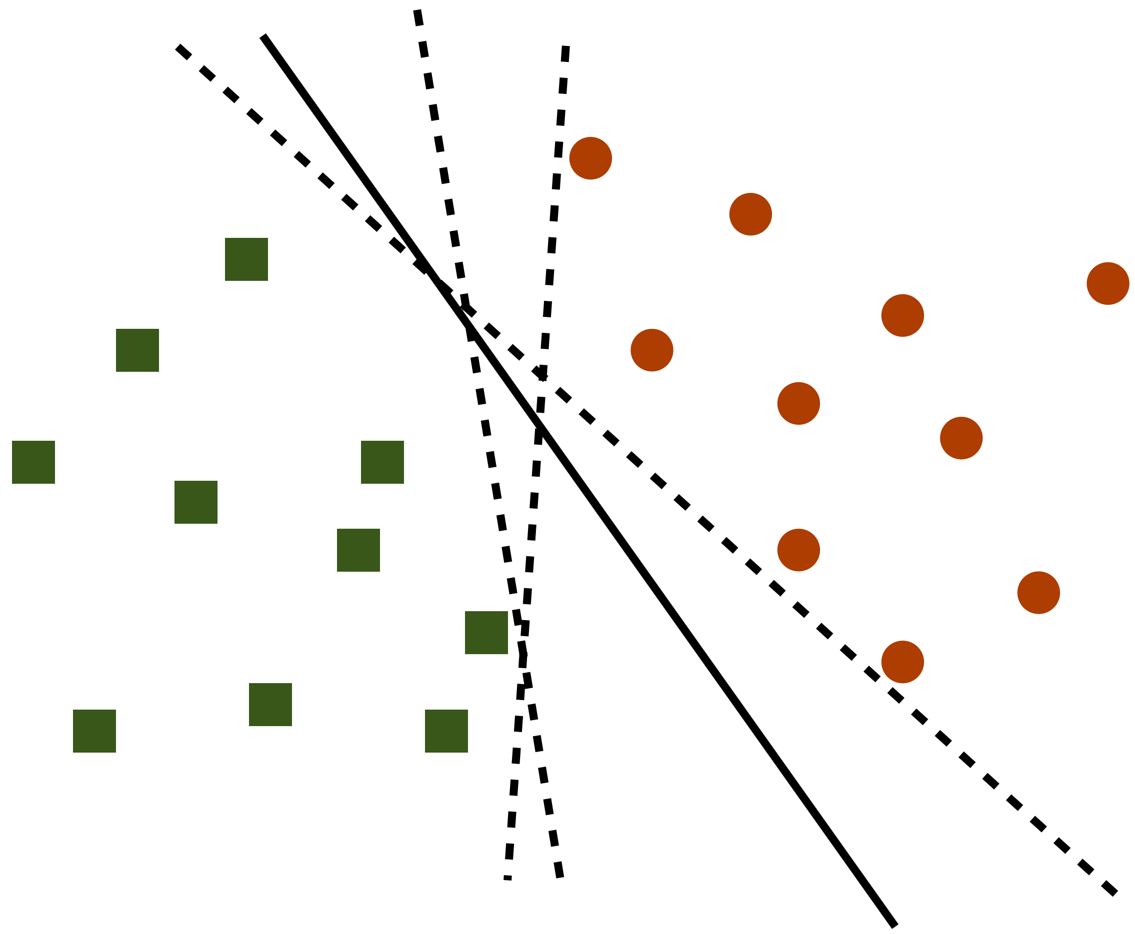
binary classification with a hyperplane



many possible solutions

Linear separators

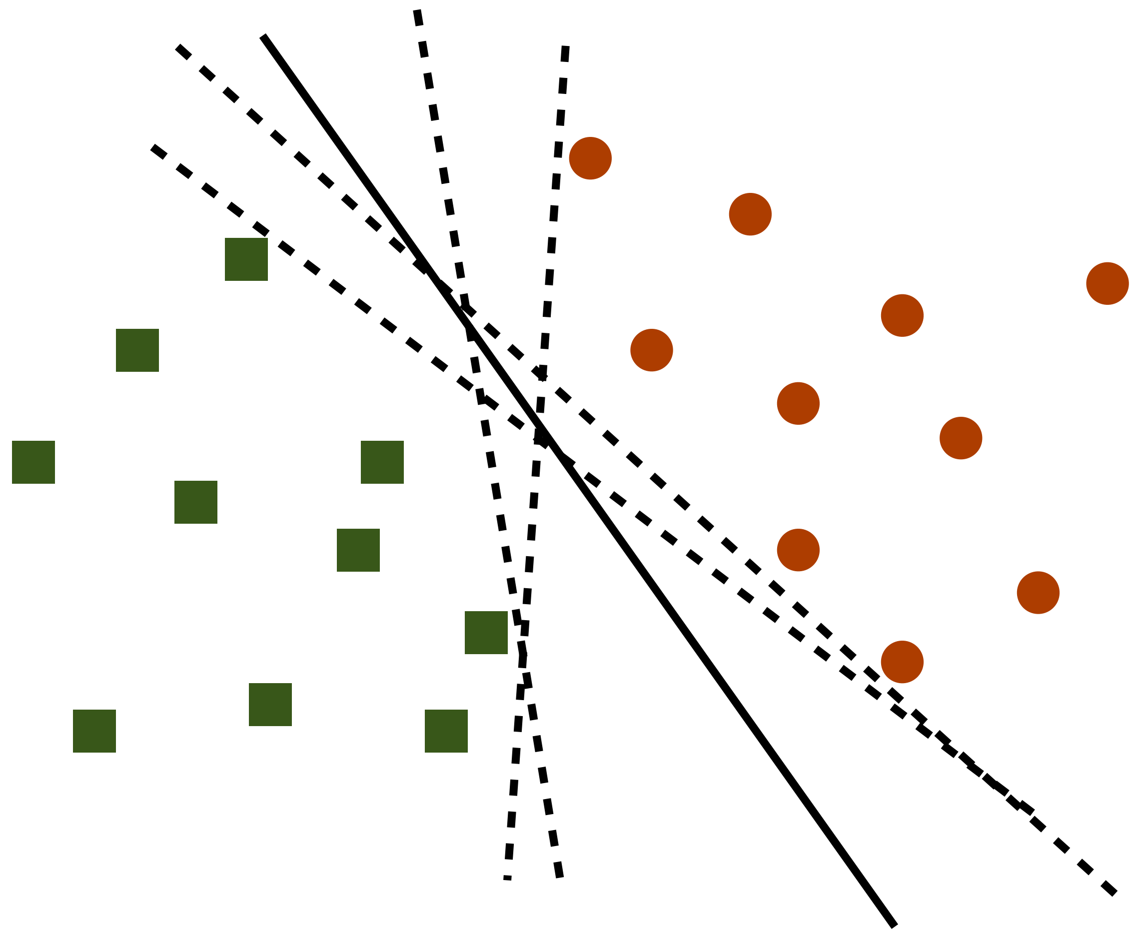
binary classification with a hyperplane



many possible solutions

Linear separators

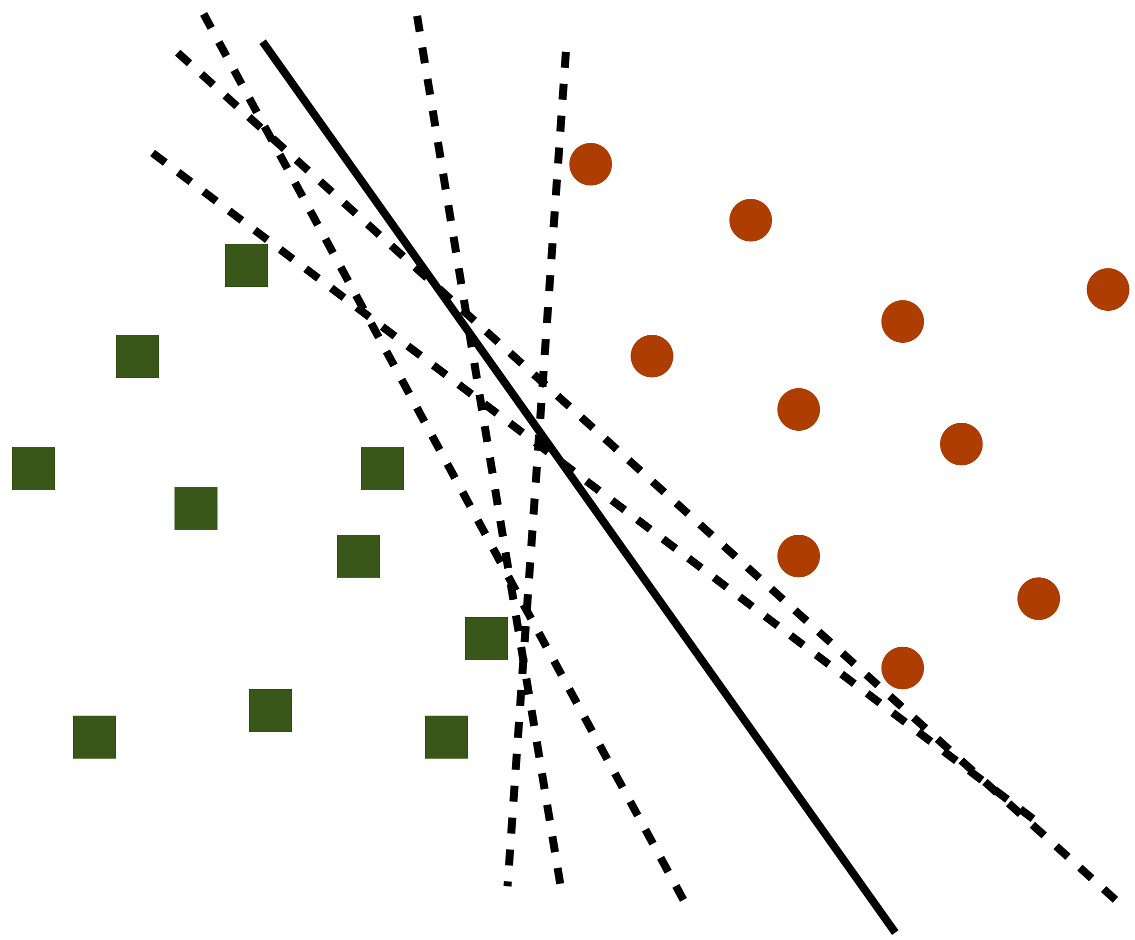
binary classification with a hyperplane



many possible solutions

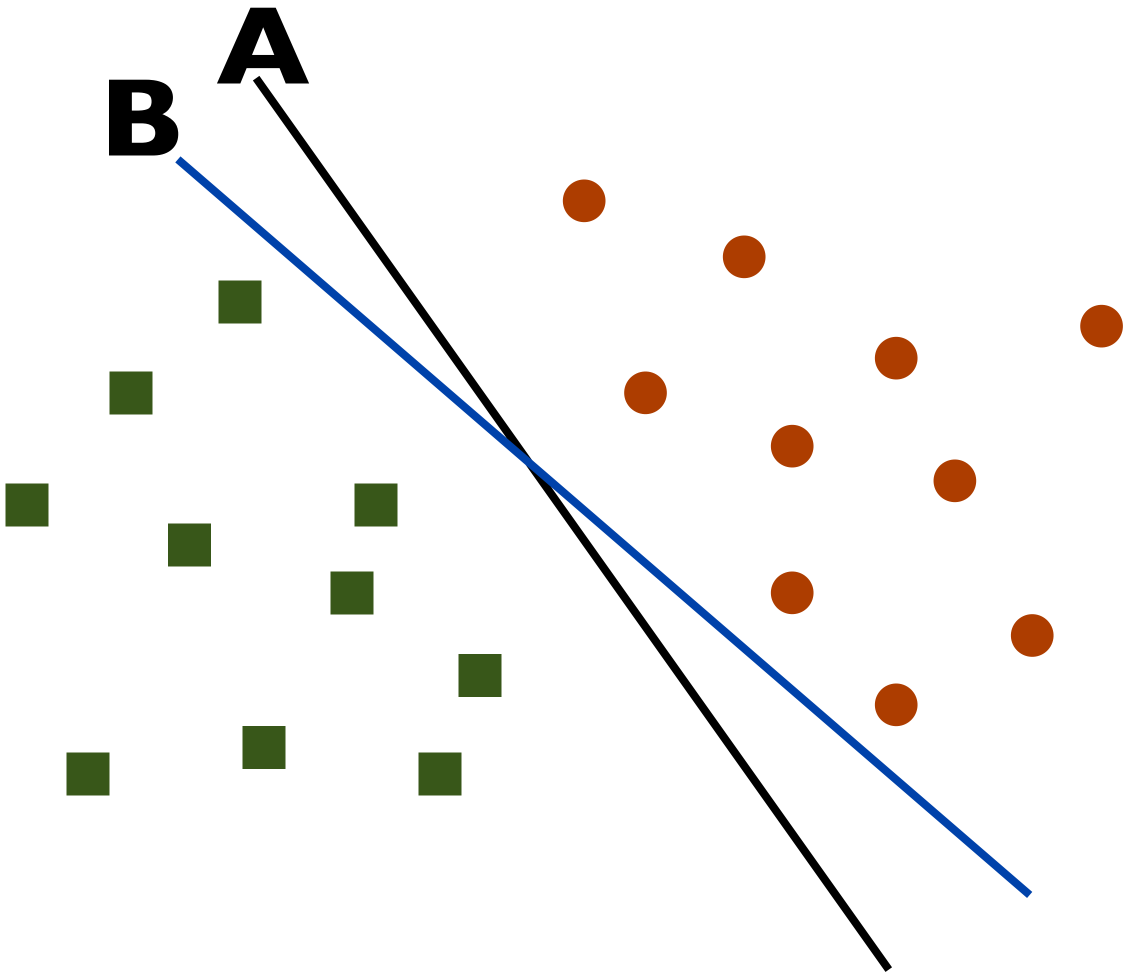
Linear separators

binary classification with a hyperplane



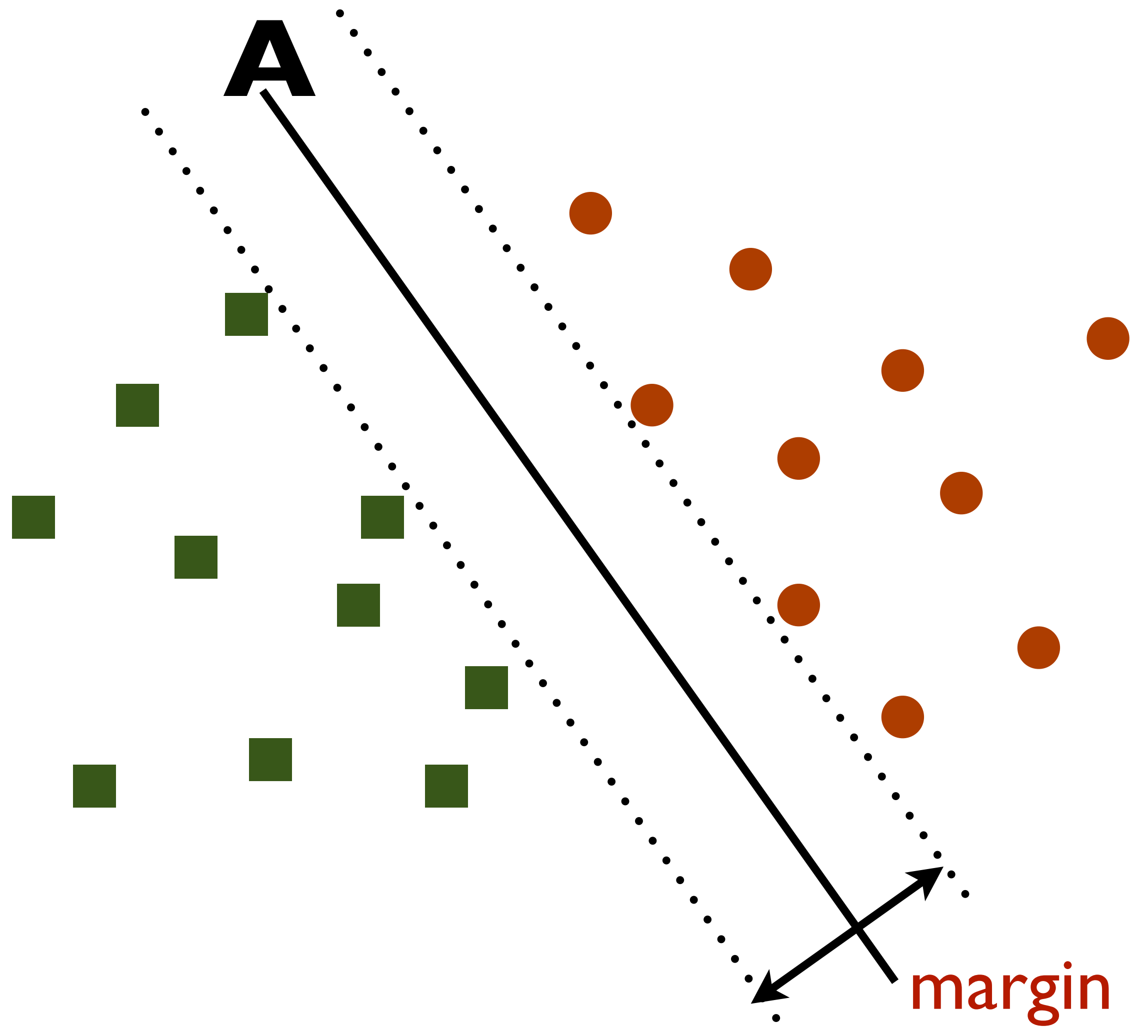
many possible solutions

Linear separators

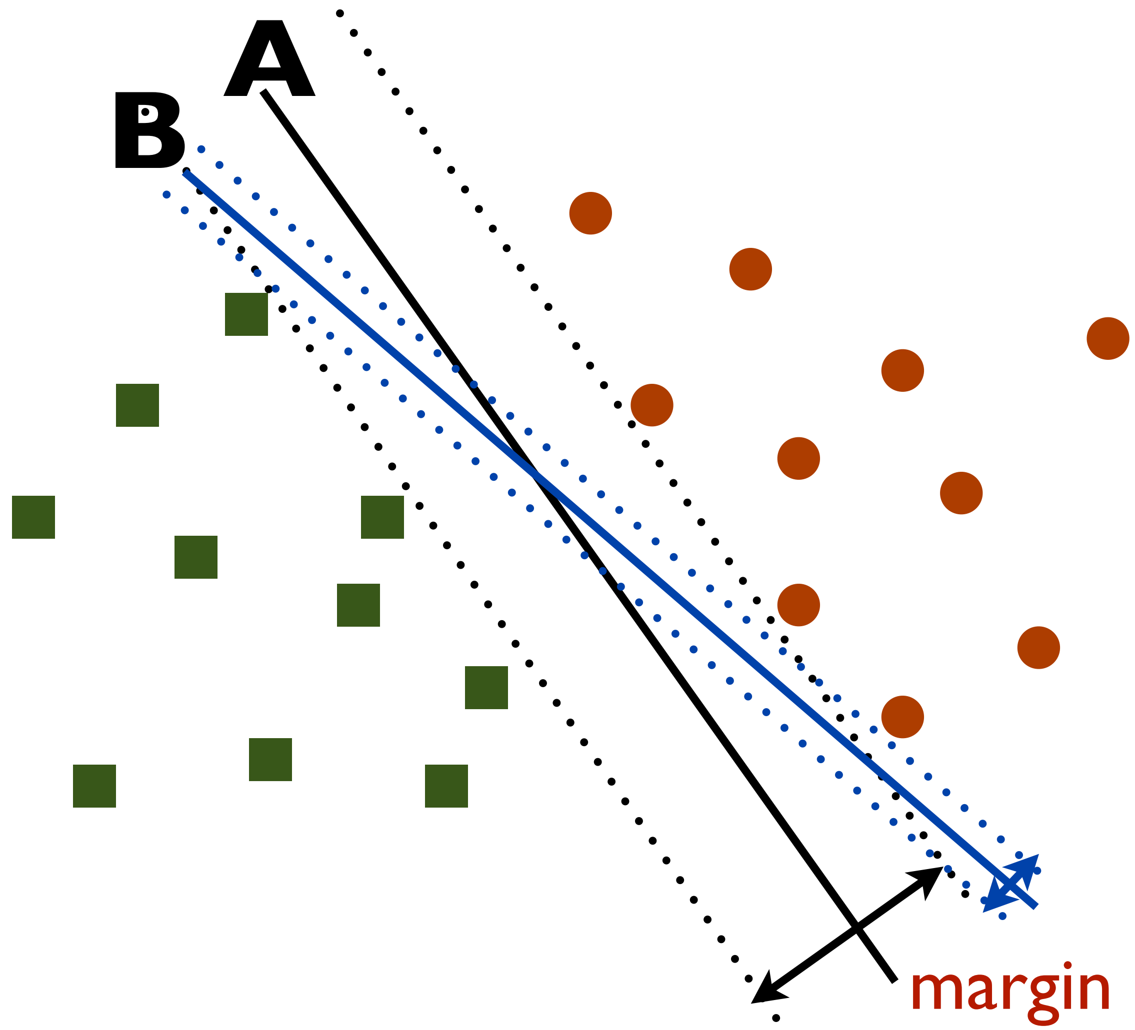


- Which one is better **A** or **B** ?
- How do you define better?

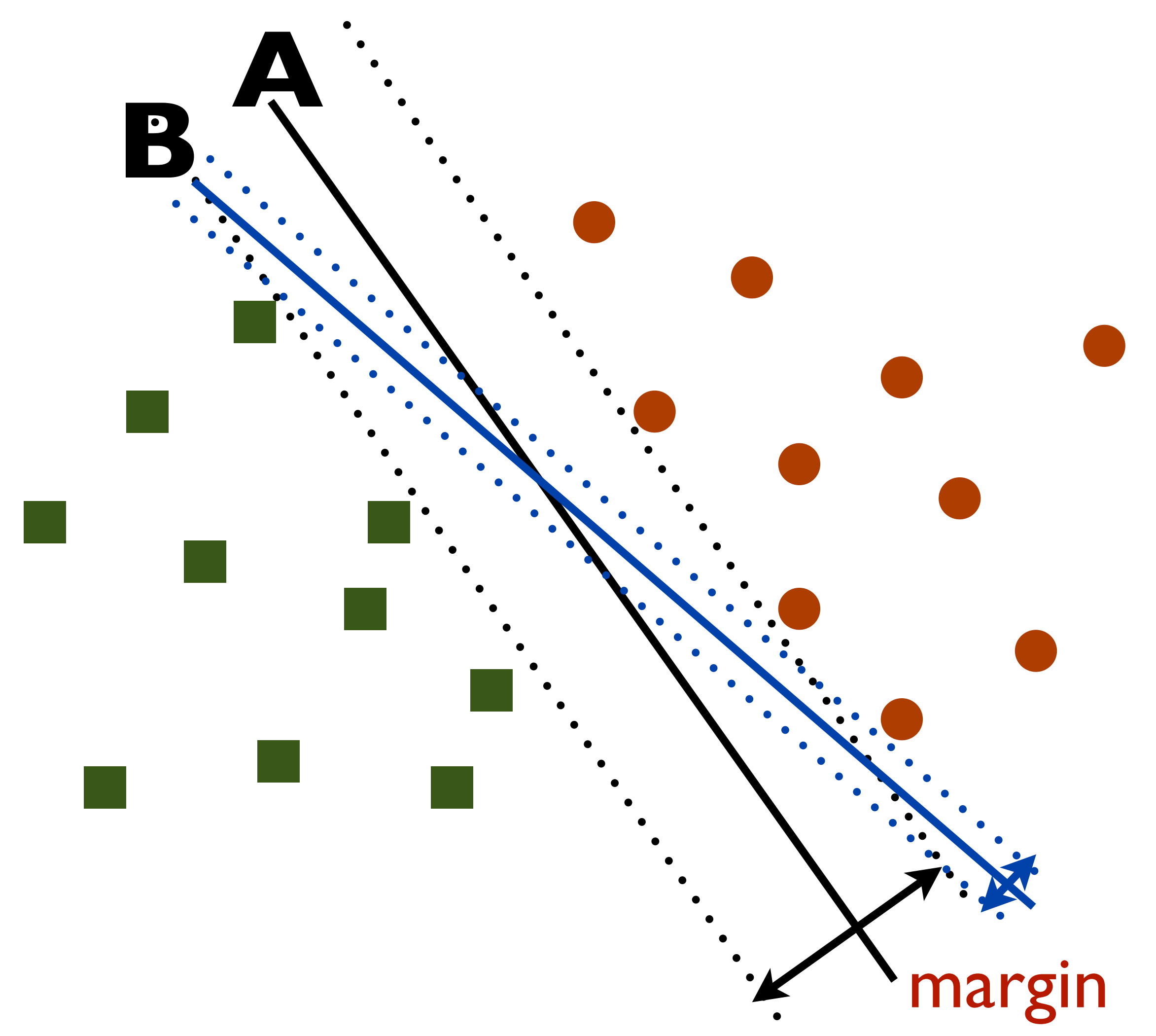
Linear separators : margin



Linear separators : margin



Linear separators : margin



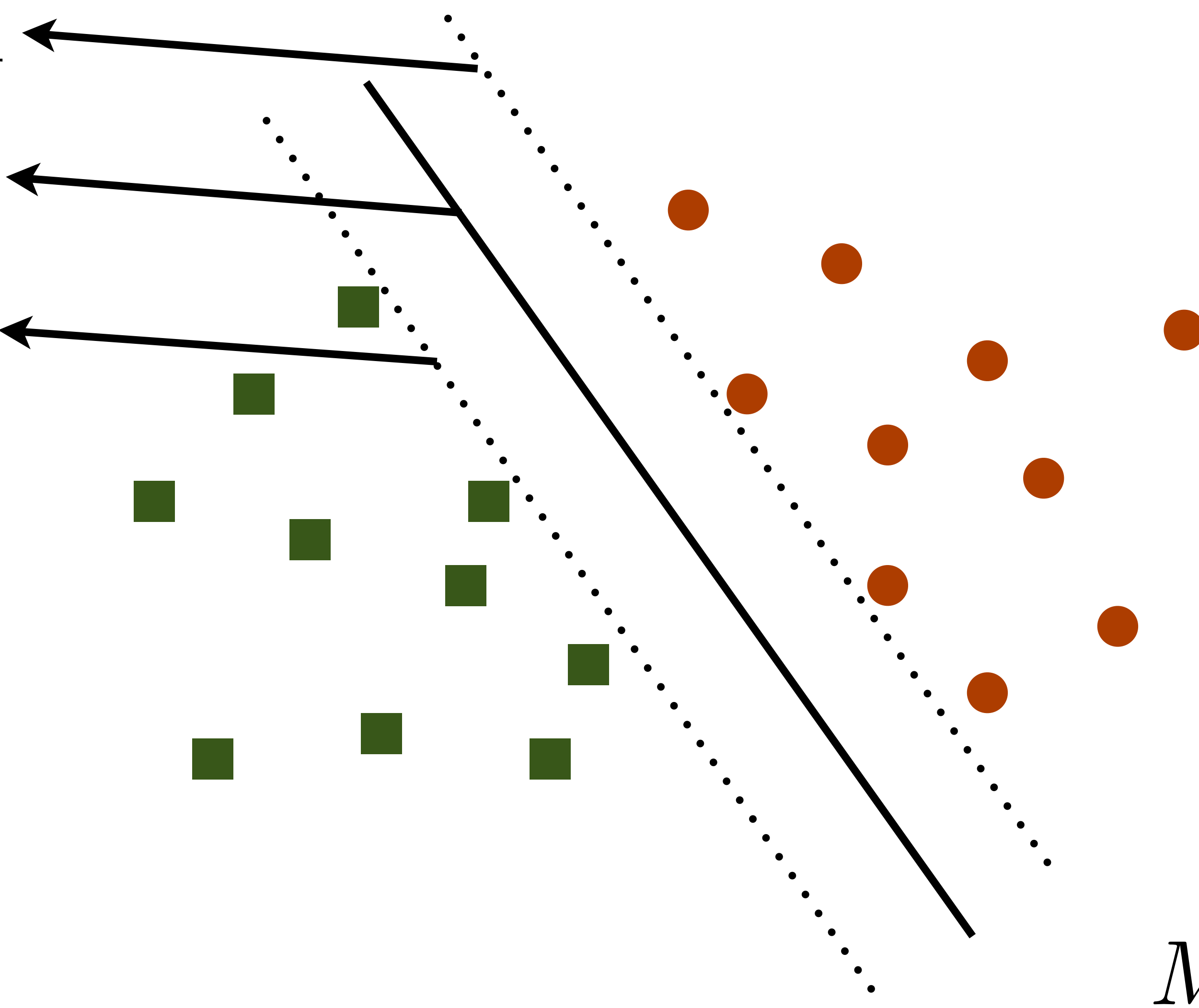
- Find a hyperplane that maximizes the margin
- Bigger margin is better, i.e., classifier **A** is better than **B**
- Why is bigger margin better?

Linear Support Vector Machines (SVMs)

$$\mathbf{w}^T\mathbf{x} + b = +1$$

$$\mathbf{w}^T\mathbf{x} + b = 0$$

$$\mathbf{w}^T\mathbf{x} + b = -1$$

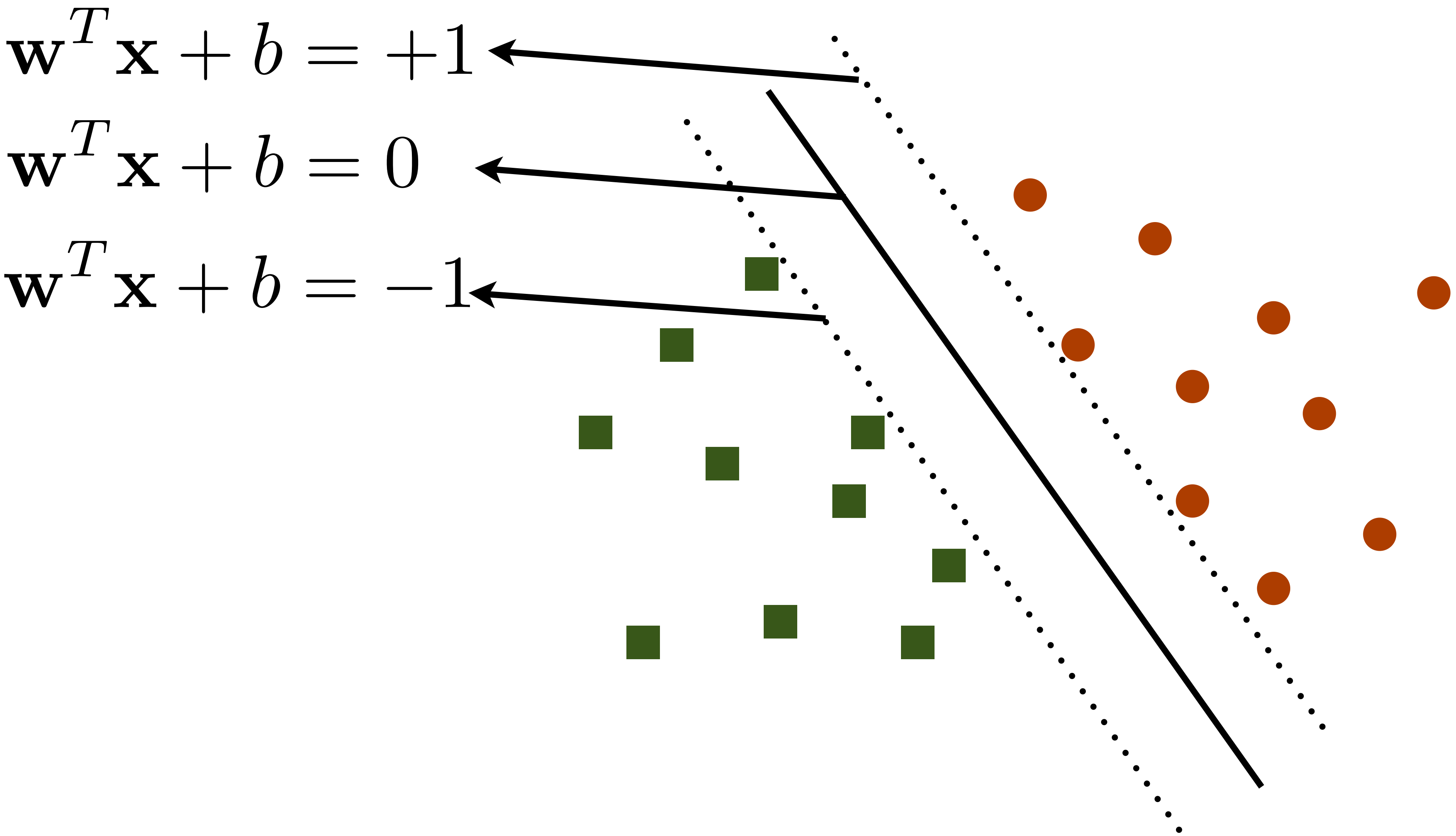


$$Margin = \frac{2}{||\mathbf{w}||^2}$$

classification function

$$f(\mathbf{x}) = sign\left(\mathbf{w}^T\mathbf{x} + b\right)$$

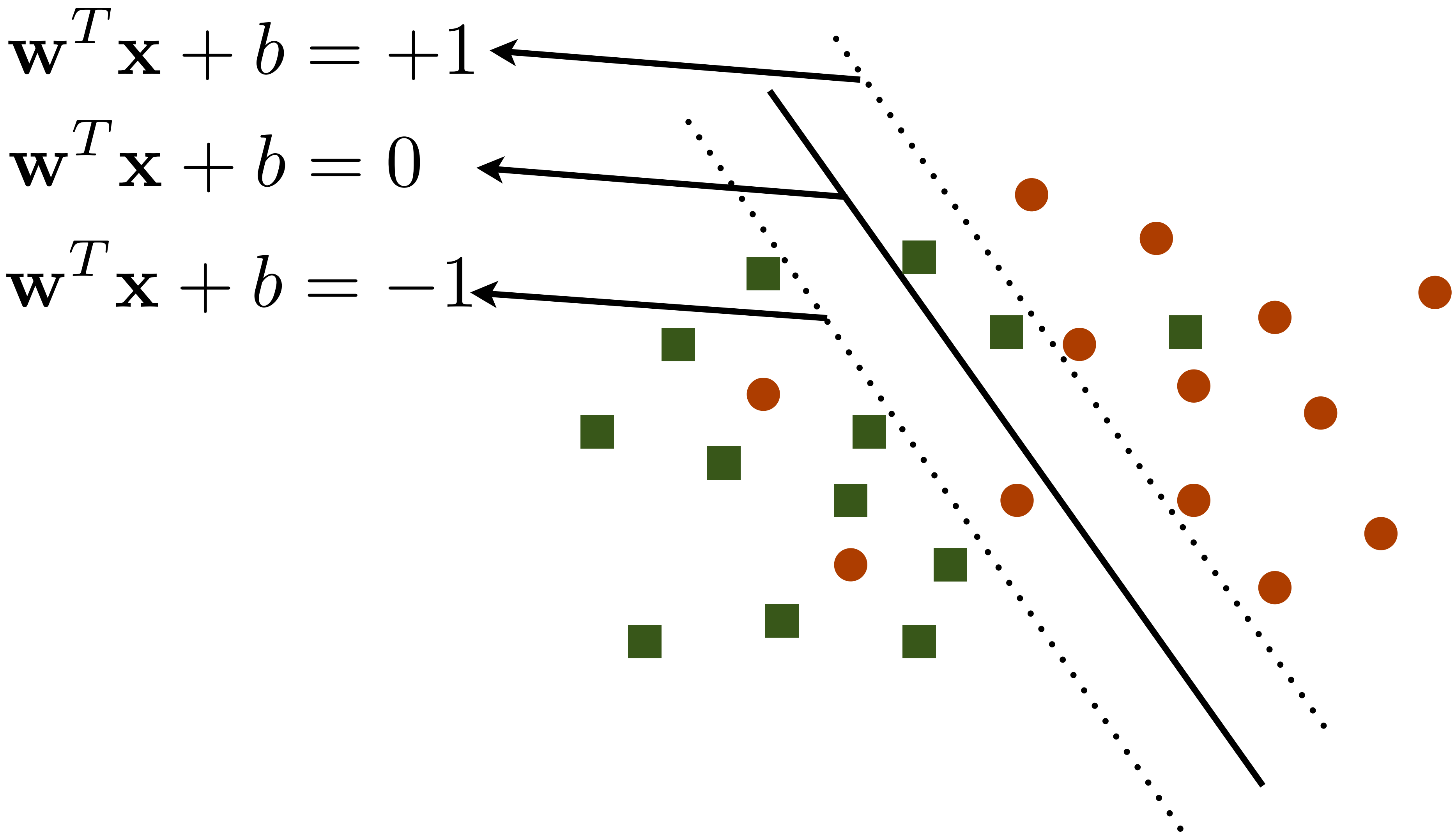
Linear SVMs



maximize: $\frac{2}{||\mathbf{w}||^2}$ or minimize: $||\mathbf{w}||^2$

subject to: $y_i \left(\mathbf{w}^T\mathbf{x}_i + b \right) \geq 1$

Linear SVMs : non-separable data



minimize: $$\frac{||\mathbf{w}||^2}{2} + C\sum_i \xi_i$$

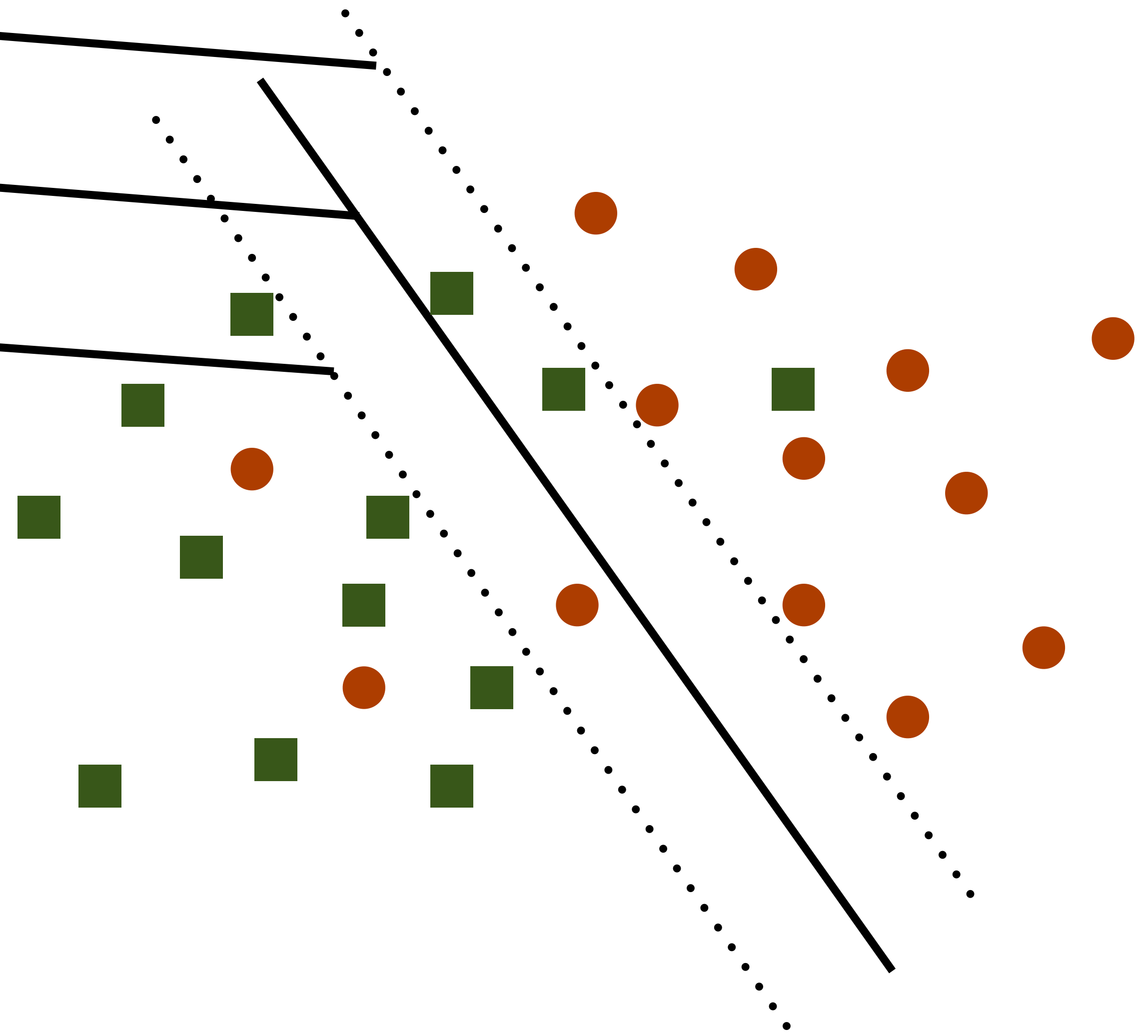
subject to: $$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

Linear SVMs : non-separable data

$$\mathbf{w}^T\mathbf{x} + b = +1$$

$$\mathbf{w}^T\mathbf{x} + b = 0$$

$$\mathbf{w}^T\mathbf{x} + b = -1$$



minimize: $$\frac{||\mathbf{w}||^2}{2} + C\sum_i \xi_i$$

slack

subject to: $$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i$$

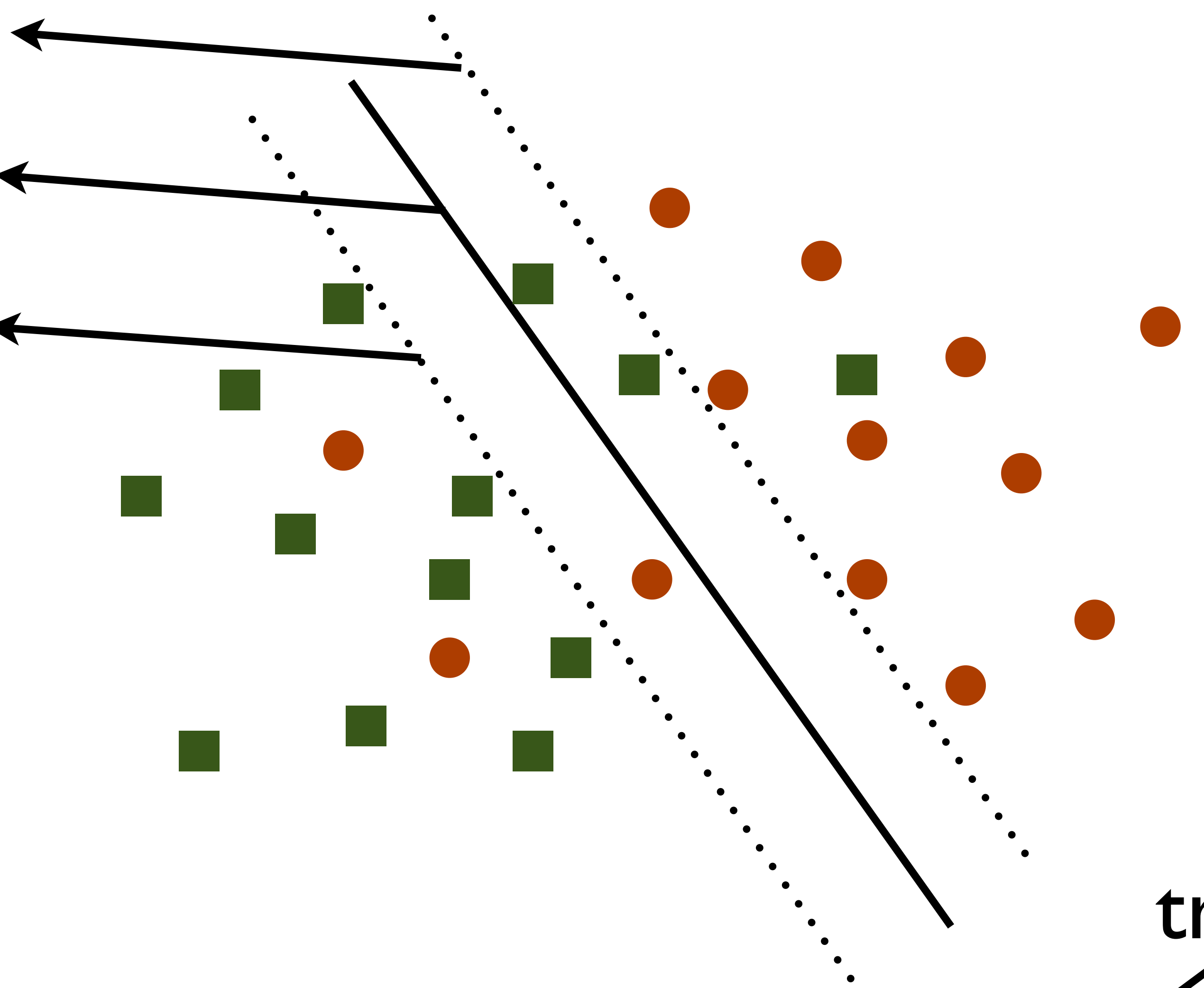
$$\xi_i \geq 0$$

Linear SVMs : non-separable data

$$\mathbf{w}^T\mathbf{x} + b = +1$$

$$\mathbf{w}^T\mathbf{x} + b = 0$$

$$\mathbf{w}^T\mathbf{x} + b = -1$$



tradeoff parameter

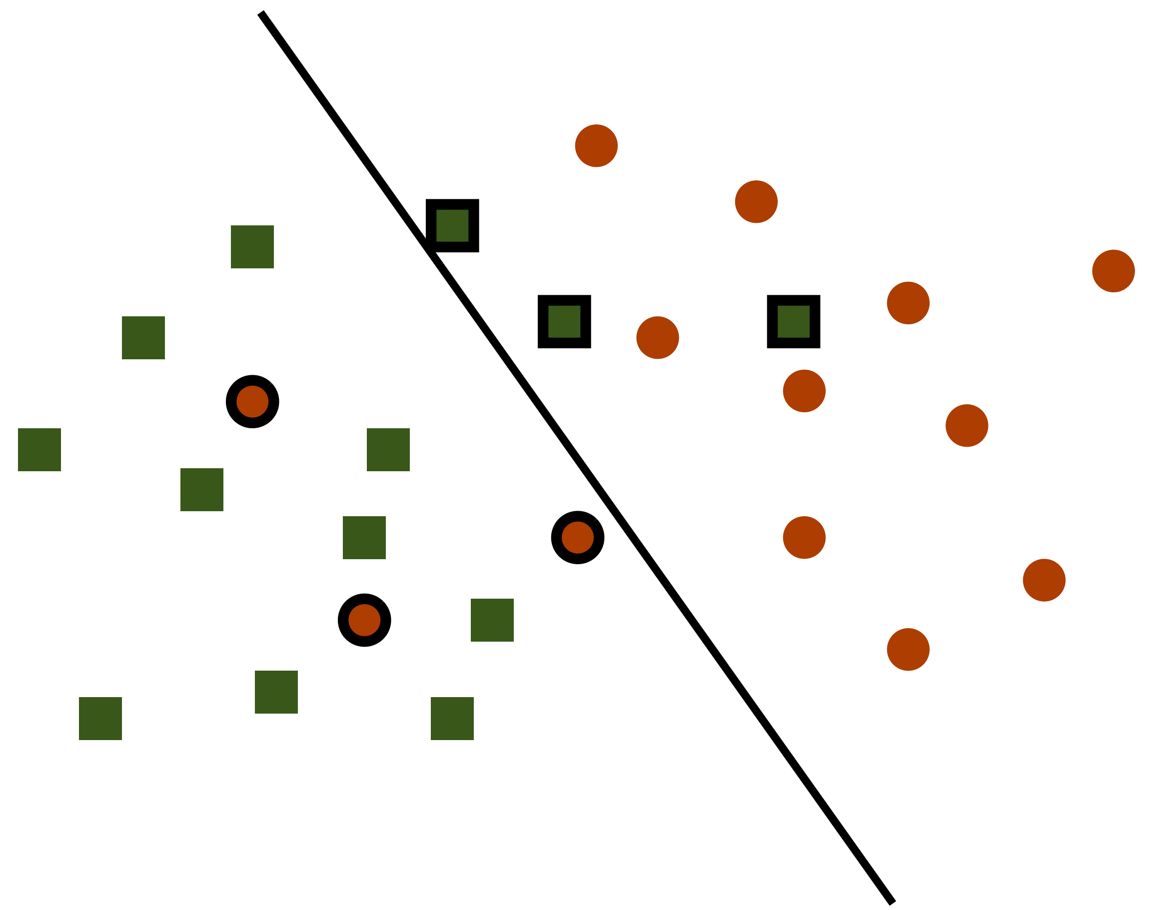
slack

minimize: $$\frac{||\mathbf{w}||^2}{2} + C\sum_i \xi_i$$

subject to: $$y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

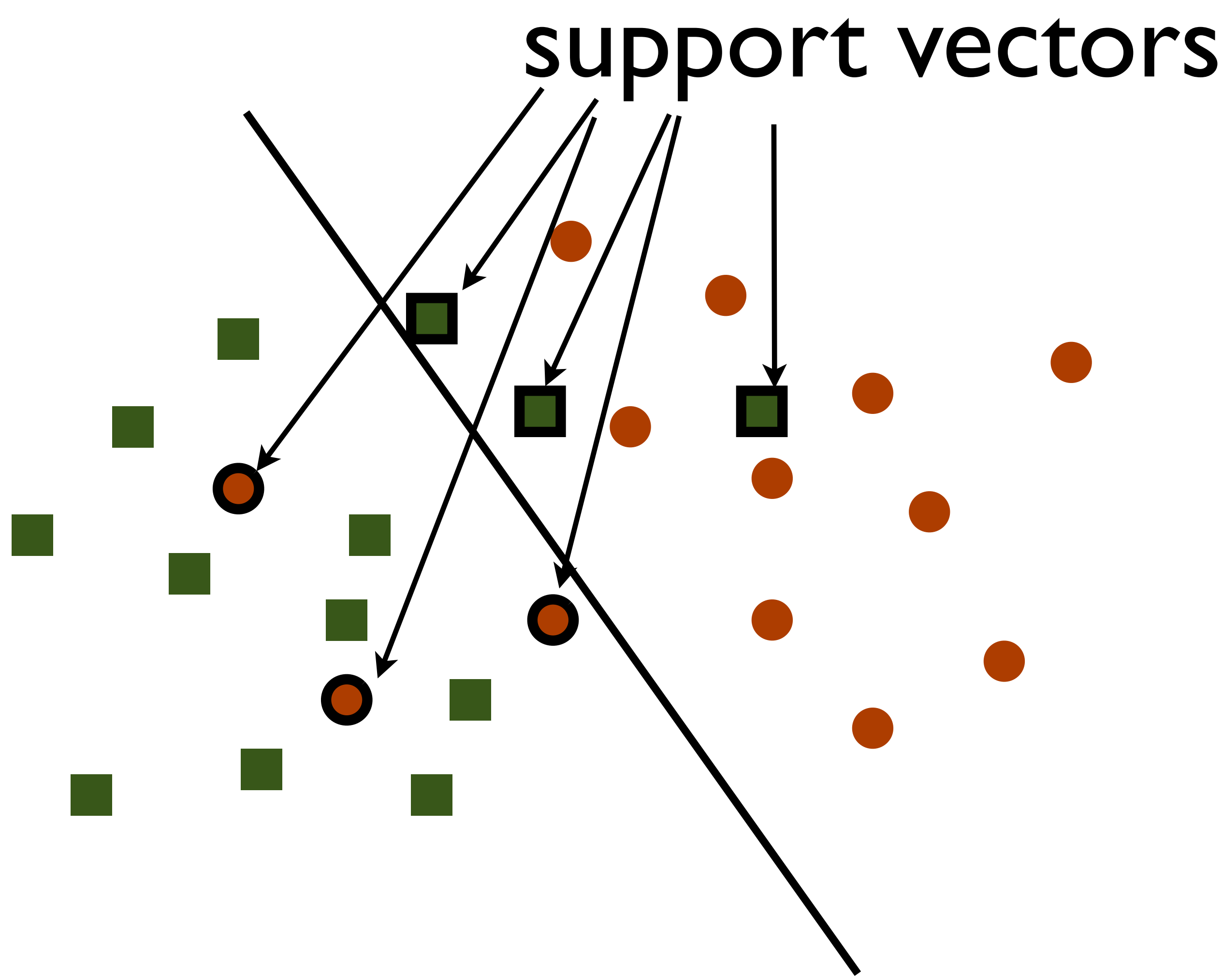
Representer theorem



classifier is a linear combination of *support vectors*

$$\mathbf{w} = \sum_{i \in \{i : y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq -1\}} \alpha_i \mathbf{x}_i$$

Representer theorem



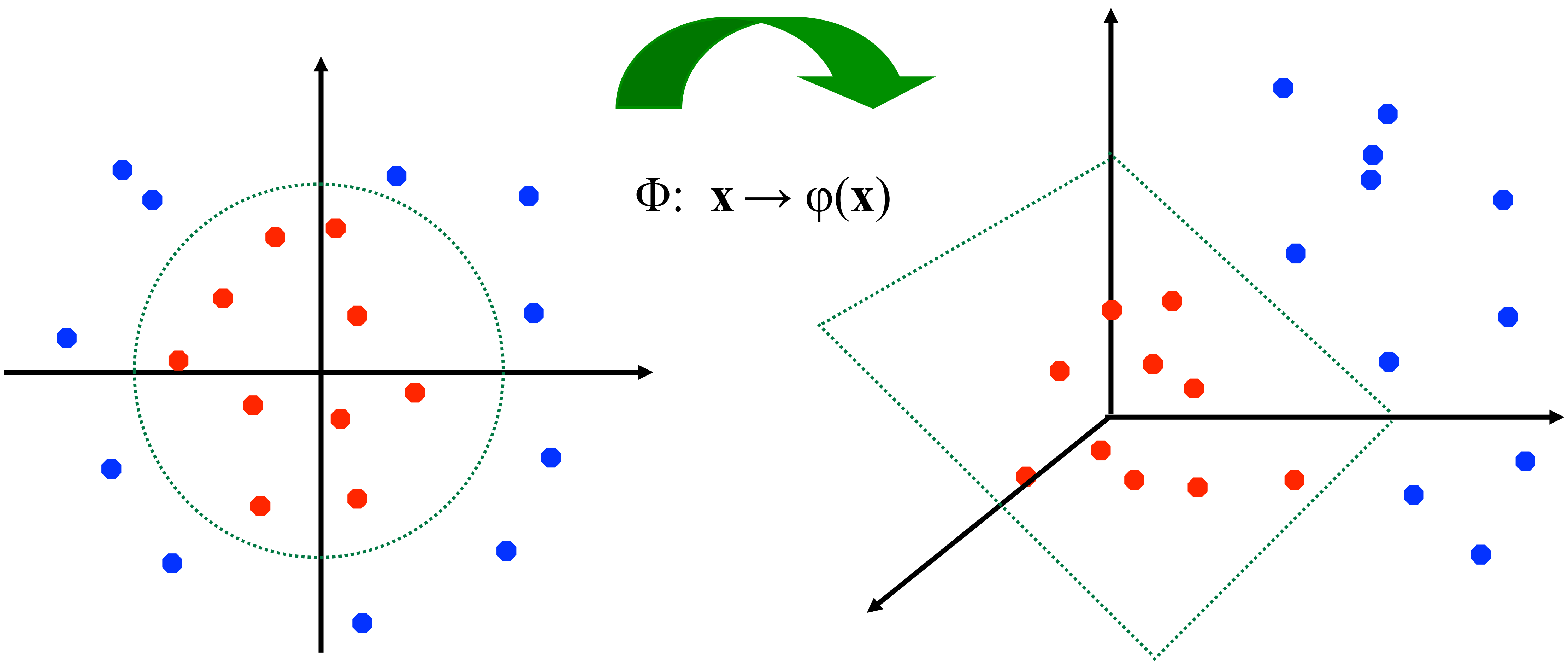
classifier is a linear combination of *support vectors*

$$\mathbf{w} = \sum_{i \in \{i : y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq -1\}} \alpha_i \mathbf{x}_i$$

support vectors

Feature maps

- Data may be too hard to separate using a linear classifier
- Map features to a higher dimensional space and use a linear classifier



Feature maps via non-linear kernels

- Use a kernel function to represent the dot product

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y})$$

- With the representer theorem we have:

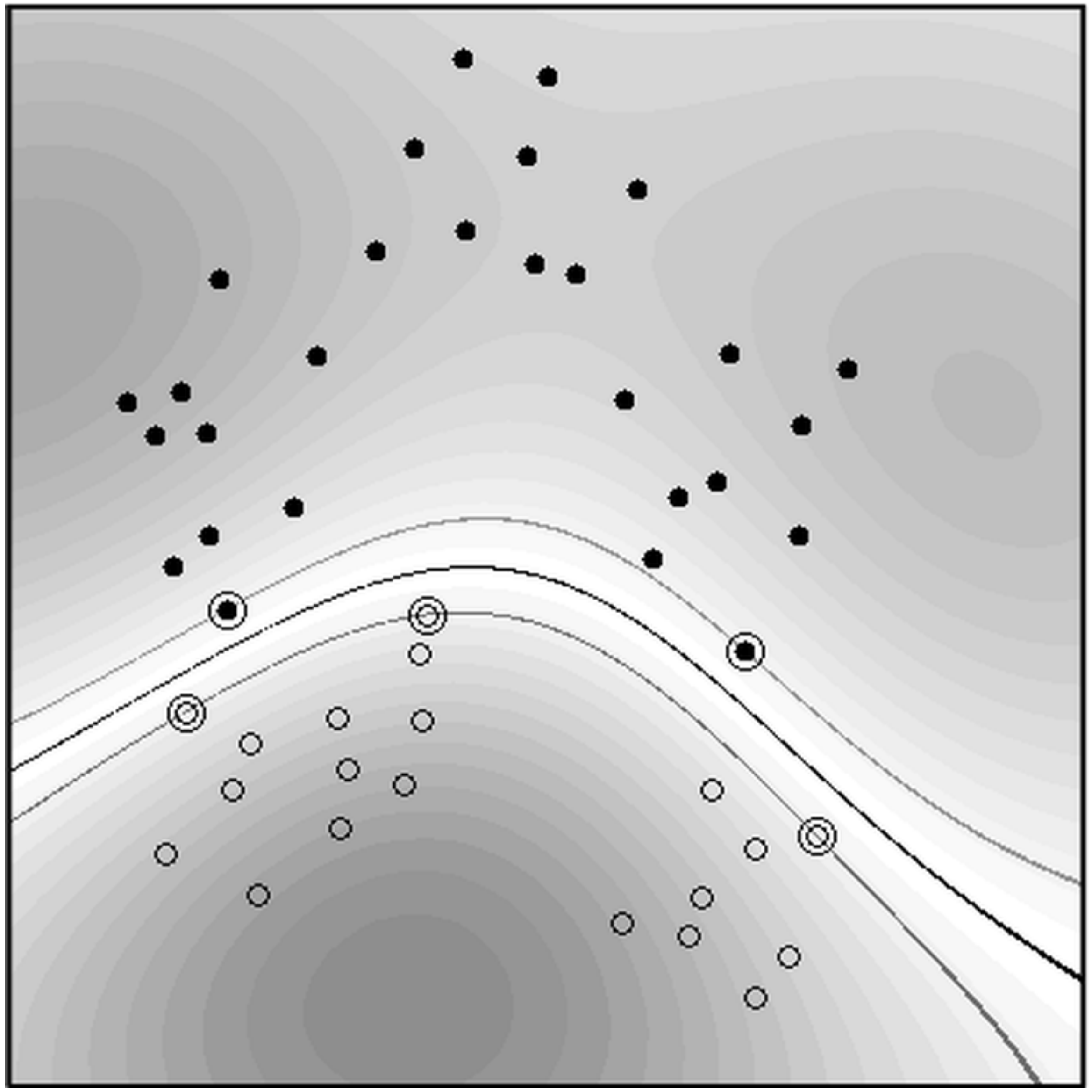
$$\mathbf{w}^T \Phi(\mathbf{x}) = \sum_{s_i \in Sup.Vec.} \alpha_i K(\mathbf{s}_i, \mathbf{x})$$

- Can use arbitrary feature maps as long as we have a kernel function (also called the "kernel trick"), e.g.,

$$K(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T \mathbf{y})^2$$

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

Even infinite dimensional features..



Gaussian kernel

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

infinite dimensional map
(via Taylor expansion)

learn non-linear boundaries

Any function can be used as long as it is a dot product of a feature map (positive definiteness)

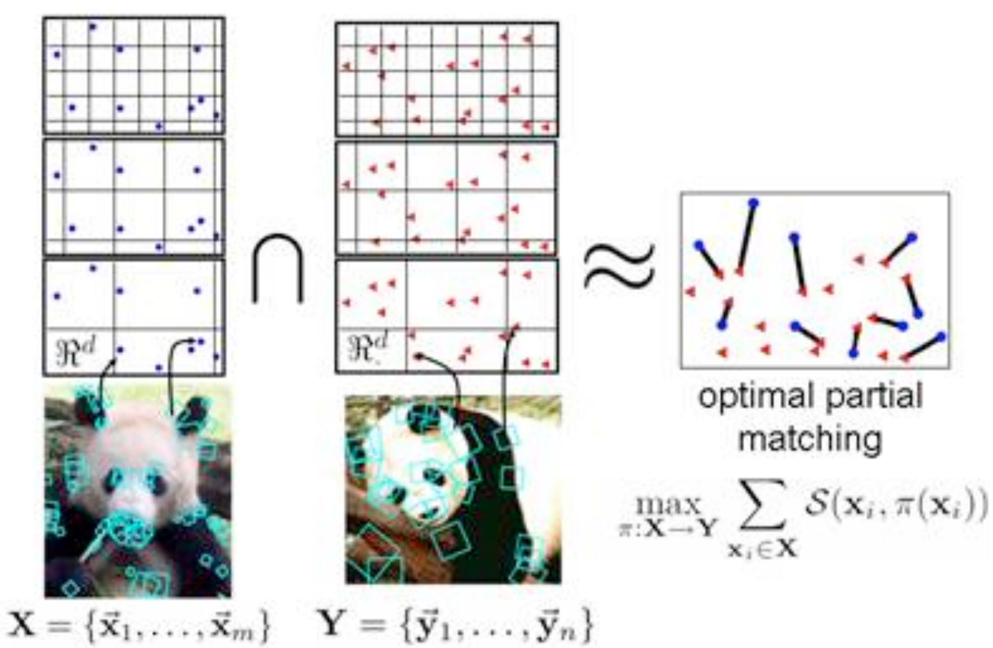
Kernels in computer vision

- Images are represented as histograms of low level features such as color and texture [Swain and Ballard 01, Odone et al. 05]

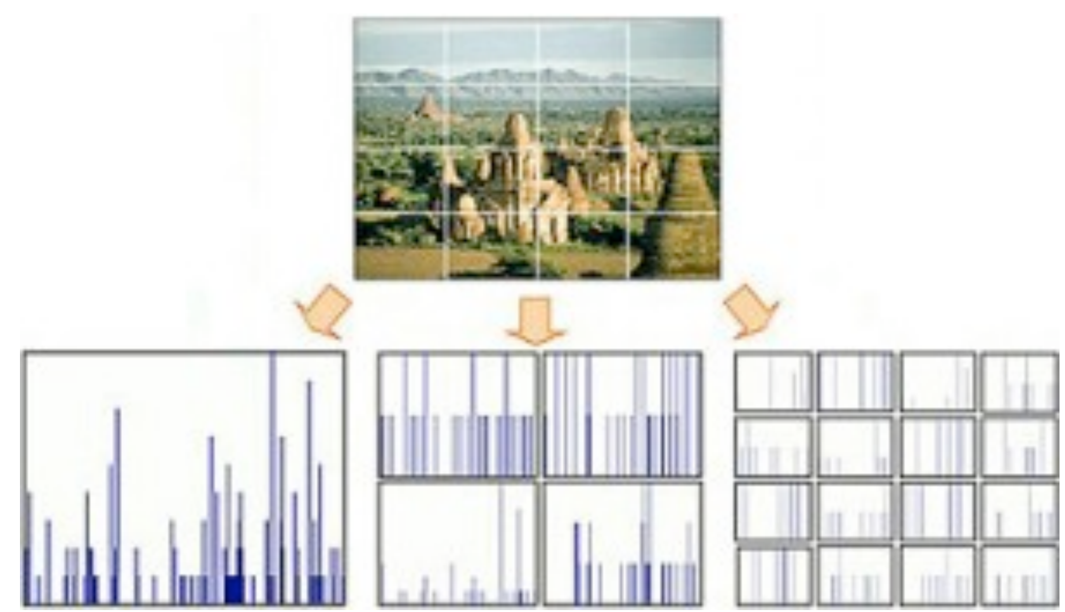
- Histogram based similarity measures are typically additive

$$K_{\min}(\mathbf{x},\mathbf{y}) = \sum \min(x_i, y_i) \qquad K_{\chi^2}(\mathbf{x},\mathbf{y}) = \sum \frac{2x_i y_i}{x_i + y_i}$$

- Other examples of additive kernels based on approximate correspondence :



Pyramid Match Kernel,
Grauman and Darrell, CVPR'05



Spatial Pyramid Match Kernel,
Lazebnik, Schmidt and Ponce, CVPR'06

The *histogram intersection* kernel

- A measure of similarity between histograms **a, b**

$$K(\mathbf{a},\mathbf{b}) = \sum_i \min(\mathbf{a}_i, \mathbf{b}_i) \qquad\qquad \mathbf{a}_i \geq 0, \mathbf{b}_i \geq 0$$

The *histogram intersection* kernel

- A measure of similarity between histograms **a, b**

$$K(\mathbf{a},\mathbf{b}) = \sum_i \min(\mathbf{a}_i, \mathbf{b}_i) \qquad\qquad \mathbf{a}_i \geq 0, \mathbf{b}_i \geq 0$$

K large : histograms are similar

K small : histograms are different

The *histogram intersection* kernel

- A measure of similarity between histograms **a, b**

$$K(\mathbf{a},\mathbf{b}) = \sum_i \min(\mathbf{a}_i, \mathbf{b}_i) \qquad \mathbf{a}_i \geq 0, \mathbf{b}_i \geq 0$$

K large : histograms are similar
K small : histograms are different

Introduced by Swain & Ballard 1991 to compare color histograms
Odone et al., 2005 proved *positive definiteness*
Hence can be used as a kernel directly with SVMs

The *histogram intersection* kernel: positive definiteness

- A measure of similarity between histograms **a, b**

$$K(\mathbf{a}, \mathbf{b}) = \sum_i \min(\mathbf{a}_i, \mathbf{b}_i) \qquad \mathbf{a}_i \geq 0, \mathbf{b}_i \geq 0$$

To see $\min(\mathbf{a}_i, \mathbf{b}_i)$ is positive definite, represent $\mathbf{a}_i, \mathbf{b}_i$ in unary

Unary representation: *n* written as *n* ones in a row

$$\min(\mathbf{a}_i, \mathbf{b}_i) =< \mathbf{a}_i^{unary}, \mathbf{b}_i^{unary} >$$

$$\min(3,5) =< [1,1,1,0,0], [1,1,1,1,1] >= 3$$

The *histogram intersection* kernel: positive definiteness

- A measure of similarity between histograms **a, b**

$$K(\mathbf{a}, \mathbf{b}) = \sum_i \min(\mathbf{a}_i, \mathbf{b}_i) \qquad \mathbf{a}_i \geq 0, \mathbf{b}_i \geq 0$$

To see $\min(\mathbf{a}_i, \mathbf{b}_i)$ is positive definite, represent $\mathbf{a}_i, \mathbf{b}_i$ in unary

Unary representation: *n* written as *n* ones in a row

$$\min(\mathbf{a}_i, \mathbf{b}_i) =< \mathbf{a}_i^{unary}, \mathbf{b}_i^{unary} >$$

$$\min(3,5) =< [1,1,1,0,0], [1,1,1,1,1] >= 3$$

Also positive definite for reals

Kernel SVMs are slow to evaluate

- The decision function is *sign(h(x))*

linear SVM
$$h(\mathbf{x}) = \sum_{i=1}^{\#dim} \mathbf{w}_i \mathbf{x}_i + b$$

kernel SVM
$$h(\mathbf{x}) = \sum_{j=1}^{\#sv} \alpha_j K(\mathbf{x}, \mathbf{s}_j) + b$$

intersection kernel SVM
$$h(\mathbf{x}) = \sum_{j=1}^{\#sv} \alpha_j \sum_{i=1}^{\#dim} \min(\mathbf{x}_i, \mathbf{s}_{i,j}) + b$$

Kernel SVMs are slow to evaluate

- The decision function is *sign(h(x))*

linear SVM
$$h(\mathbf{x}) = \sum_{i=1}^{\#dim} \mathbf{w}_i \mathbf{x}_i + b$$

kernel SVM
$$h(\mathbf{x}) = \sum_{j=1}^{\#sv} \alpha_j K(\mathbf{x}, \mathbf{s}_j) + b$$

intersection kernel SVM
$$h(\mathbf{x}) = \sum_{j=1}^{\#sv} \alpha_j \sum_{i=1}^{\#dim} \min(\mathbf{x}_i, \mathbf{s}_{i,j}) + b$$

linear SVM = O(#dim)
kernel SVM = O(#dim X #sv)

Can be orders of magnitude slower

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign($h(x)$)

$$
\begin{aligned}
h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign($h(x)$)

$$
\begin{aligned}
h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

Evaluating each dimension **O(#sv)**

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}
$$

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign($h(x)$)

$$\begin{aligned} h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\ &= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\ &= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i) \end{aligned}$$

Evaluating each dimension **O(#sv)**

$$\begin{aligned} h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\ &= \boxed{\sum_{x_i^j < x_i} \alpha^j x_i^j} + \boxed{\left(\sum_{x_i^j \geq x_i} \alpha^j \right)} x_i \end{aligned}$$

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign(*h*(*x*))

$$
\begin{aligned}
h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

Independent of input
Can be precomputed

Evaluating each dimension **O(#sv)**

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \boxed{\sum_{x_i^j < x_i} \alpha^j x_i^j} + \boxed{\left(\sum_{x_i^j \geq x_i} \alpha^j \right)} x_i
\end{aligned}
$$

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign(h(x))

$$
\begin{aligned}
h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

Evaluating each dimension ~~**O(#sv)**~~

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \boxed{\sum_{x_i^j < x_i} \alpha^j x_i^j} + \boxed{\left(\sum_{x_i^j \geq x_i} \alpha^j \right)} x_i
\end{aligned}
$$

Independent of input
Can be precomputed

To evaluate, find the position of input in the sorted list of support vectors. Can be done using binary search in **O(log #sv)** time

Maji, Berg, Malik '08
Herbster '01

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign(*h*(*x*))

$$\begin{aligned} h(x) &= \sum_{j=1}^{\#\text{sv}} \alpha^j \left(\sum_{i=1}^{\#\text{dim}} \min(x_i, x_i^j) \right) + b \\ &= \sum_{i=1}^{\#\text{dim}} \left(\sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\ &= \sum_{i=1}^{\#\text{dim}} h_i(x_i) \end{aligned}$$

Evaluating each dimension **~~O(#sv)~~**

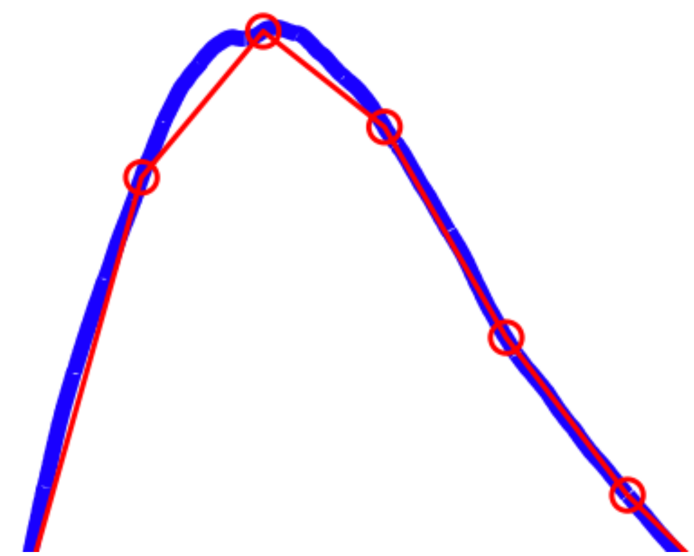
$$\begin{aligned} h_i(x_i) &= \sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) + b \\ &= \boxed{\sum_{x_i^j < x_i} \alpha^j x_i^j} + \boxed{\left(\sum_{x_i^j \geq x_i} \alpha^j \right)} x_i \end{aligned}$$

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign($h(x)$)

$$
\begin{aligned}
h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

O(log #sv)



Evaluating each dimension ~~**O(#sv)**~~

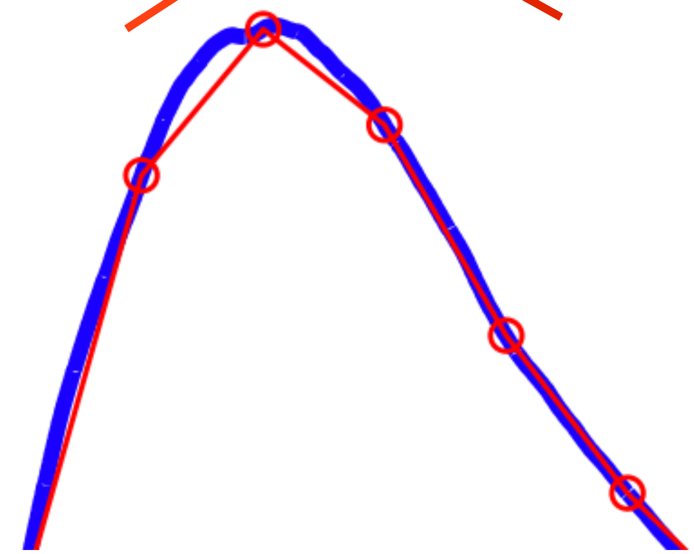
$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}
$$

Additive kernel SVMs can be efficiently evaluated

The decision function of the classifier is sign($h(x)$)

$$
\begin{aligned}
h(x) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \left(\sum_{i=1}^{\#\mathrm{dim}} \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} \left(\sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\mathrm{dim}} h_i(x_i)
\end{aligned}
$$

~~**O(log #sv)**~~



Evaluating each dimension ~~**O(#sv)**~~

$$
\begin{aligned}
h_i(x_i) &= \sum_{j=1}^{\#\mathrm{sv}} \alpha^j \min(x_i, x_i^j) + b \\
&= \sum_{x_i^j < x_i} \alpha^j x_i^j + \left(\sum_{x_i^j \geq x_i} \alpha^j \right) x_i
\end{aligned}
$$

Consider a piecewise polynomial approximation **O(1)** time. Saves time and memory!

Works for any additive kernel

Maji, Berg, Malik '08

Timing results

- Time to classify 10,000 features

Dataset	Model parameters		SVM kernel type		fast IKSVMs		
	#SVs	#features	linear	intersection	binary search	piecewise-const	piecewise-lin
INRIA Ped	3363	1360	0.07±0.00	659.1±1.92	2.57±0.03	0.34±0.01	0.43±0.01
DC Ped	5474±395	656	0.03±0.00	459.1±31.3	1.43±0.02	0.18±0.01	0.22±0.00
Caltech 101	175±46	1360	0.07±0.01	24.77±1.22	1.63±0.12	0.33±0.03	0.46±0.03

Linear SVM are fastest, but usually have worse performance than non-linear kernels

IKSVM with multi-scale HOG features beat Dalal&Triggs, also work well for DC ped. and Caltech 101 datasets

Up to 3 orders of magnitude faster

Maji, Berg & Malik, 08

Timing results

- Time to classify 10,000 features

saves memory

Dataset	Model parameters		SVM kernel type		fast IKSVMs		
	#SVs	#features	linear	intersection	binary search	piecewise-const	piecewise-lin
INRIA Ped	3363	1360	0.07±0.00	659.1±1.92	2.57±0.03	0.34±0.01	0.43±0.01
DC Ped	5474±395	656	0.03±0.00	459.1±31.3	1.43±0.02	0.18±0.01	0.22±0.00
Caltech 101	175±46	1360	0.07±0.01	24.77±1.22	1.63±0.12	0.33±0.03	0.46±0.03

Linear SVM are fastest, but usually have worse performance than non-linear kernels

IKSVM with multi-scale HOG features beat Dalal&Triggs, also work well for DC ped. and Caltech 101 datasets

Up to 3 orders of magnitude faster

Maji, Berg & Malik, 08

Shape of 1-d functions

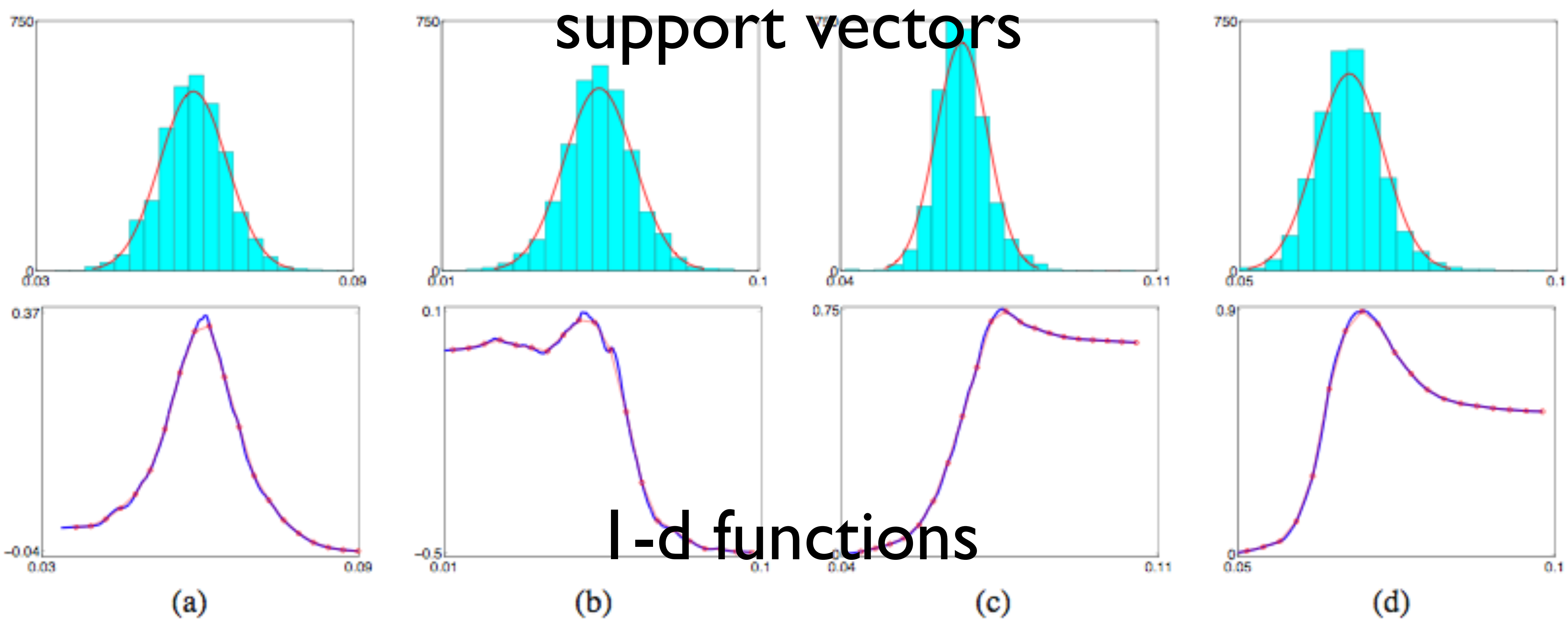


Figure 1. Each column (a-d) shows the distribution of the support vectors values along a dimension with a Gaussian fit (top) and the function $h_i(x)$ vs. x with a piecewise linear fit using 20 uniformly spaced points (bottom) of an IKSVM model trained on the INRIA dataset. Unlike the distribution of the training data which are heavy tailed, the values of the support vectors tend to be clustered.

learned functions are usually smooth, i.e., need small number of bins to approximate the classifier well

Learn additive classifiers directly

- Additive kernel SVMs are additive classifiers, i.e., the method for intersection kernel works for *any additive kernel*
- Optimize the hinge loss with a parametric representation of *h(x)*

$$
\begin{aligned}
h(x) &= \sum_{i=1}^{\#\text{dim}} \left(\sum_{j=1}^{\#\text{sv}} \alpha^j \min(x_i, x_i^j) \right) + b \\
&= \sum_{i=1}^{\#\text{dim}} h_i(x_i)
\end{aligned}
$$

- For piecewise linear functions, we obtain just a bigger linear model, and most linear solvers can be adapted to solve this problem

Additive kernel SVMs and Generalized Additive Models

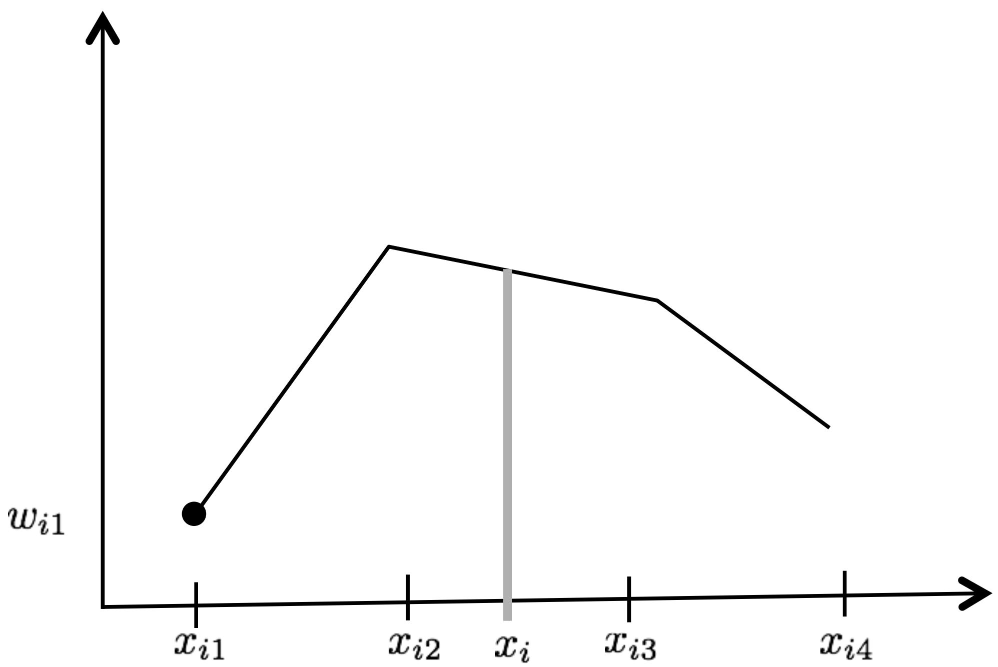
$$f(x_1, x_2, \ldots, x_n) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$$

- Why use them?
  - **Efficiency** : can be efficiently evaluated
  - **Interpretability** : Simple generalization of linear classifiers, i.e., may lead to models that are interpretable
- Well known in the statistics community
  - Generalized Additive Models (Hastie & Tibshirani '90)
- However traditional learning algorithms do not scale well (e.g. "backfitting algorithm")

Generalization of a linear classifier

$$\min w'w + c\sum \xi^j$$
$$y^j\left(w'x^j + b\right) > 1 - \xi^j$$
$$\xi^j > 0$$

$$\min \hat{w}' H \hat{w} + c\sum \xi^j$$
$$y^j\left(\hat{w}'\hat{x}^j + b\right) > 1 - \xi^j$$
$$\xi^j > 0$$



$$\sum h_i(x_i) \qquad \sum \hat{h}_i(\hat{x}_i)$$

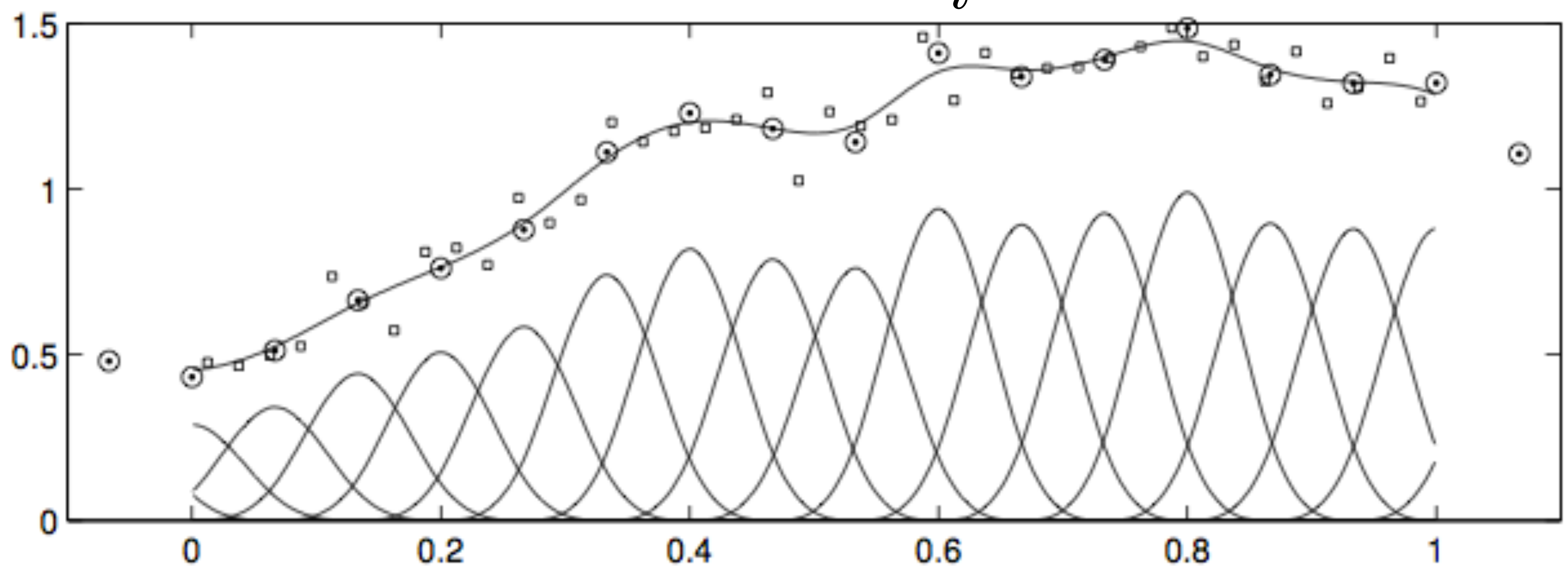
$$x_i = \alpha x_{i2} + (1-\alpha)x_{i3}$$
$$\hat{x}_i = \begin{bmatrix} 0 & \alpha & 1-\alpha & 0 \end{bmatrix}$$
$$\hat{w}_i = \begin{bmatrix} w_{i1} & w_{i2} & w_{i3} & w_{i4} \end{bmatrix}$$

H is tridiagonal
inv(*H*) is block diagonal
(for dual methods)

Feature maps via basis expansions

$$f(x_1, x_2, \ldots, x_n) = f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n)$$

$$\sum_i w_i \phi_i$$



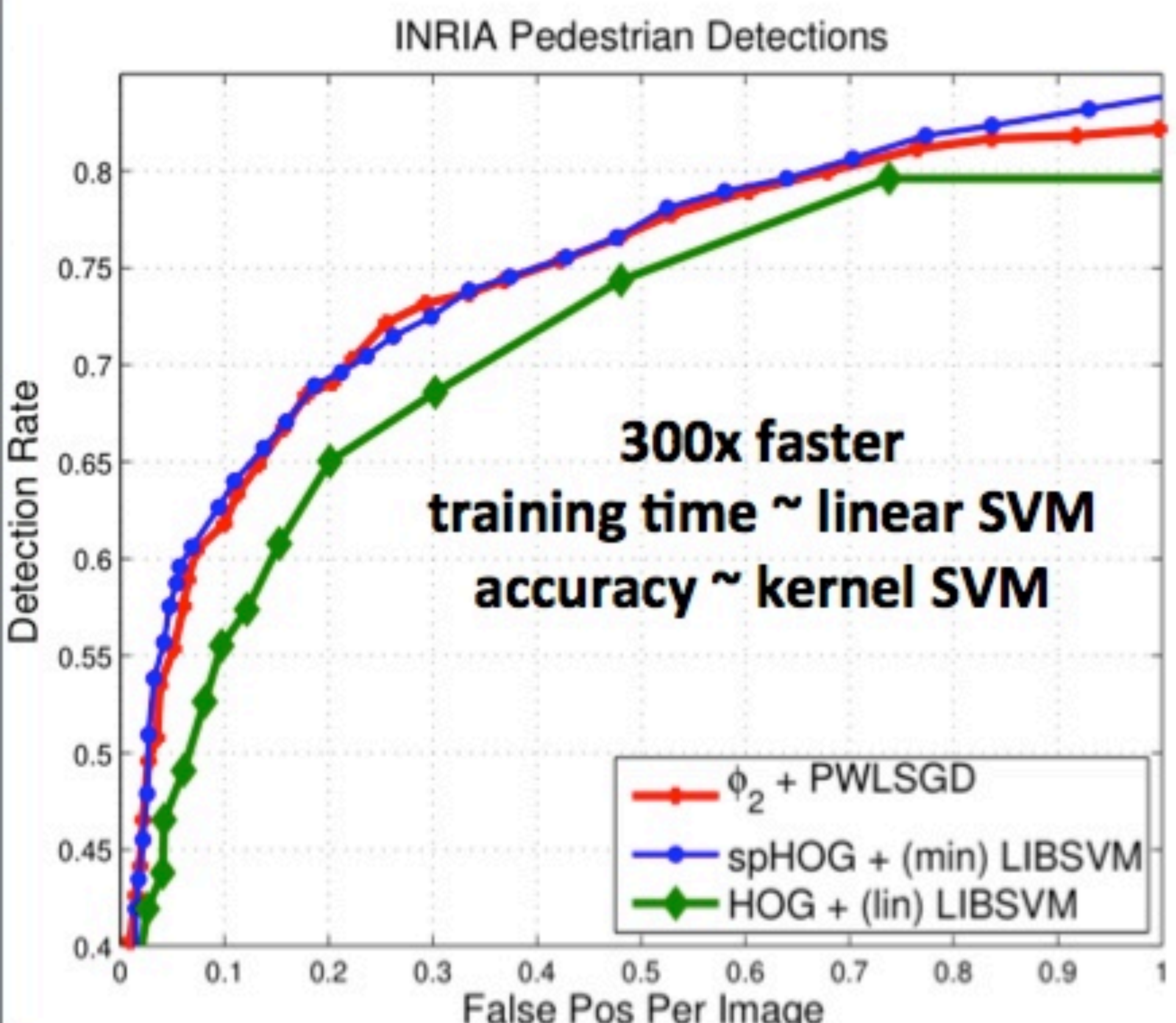
local splines are the basis

In general can choose any orthonormal basis

Maji & Berg '09, Maji '12

Faster training times

Dataset	Linear		Piecewise Linear		IK SVM	
	Time	Accuracy	Time	Accuracy	Time	Accuracy
INRIA pedestrians	20s	see curve	**76s**	**see curve**	~ 3 hr	see curve
Caltech101, 15 examples	18.6s	41.2%	**238s**	**49.9%**	844s	50.1%
Caltech101, 30 examples	40.5s	46.2%	**291s**	**55.4%**	2686s	56.5%



Conclusions

- Linear SVMs are the fastest to evaluate and train, are the classifier of choice for large scale tasks

- Kernel SVMs are more expressive but often significantly slower

- Additive kernels are ***widely used*** in computer vision and allow ***efficient evaluation*** *and* ***significantly faster training*** times than standard kernel SVMs

- *References:*

  - *Classification using Intersection Kernel SVMs is efficient",* S. Maji, A.C. Berg and J.Malik, CVPR 2009

  - *Efficient Classification for Additive Kernel SVMs*, S.Maji, A.C. Berg, and J.Malik, IEEE T-PAMI 2013 (to appear)

  - *Max-margin additive classifiers for detection*, S.Maji and A.C.Berg, ICCV 2009

  - *Linearized smooth additive classifiers,* S.Maji, ECCV Workshop on web-scale vision and social media, 2012