Support Vector Machines

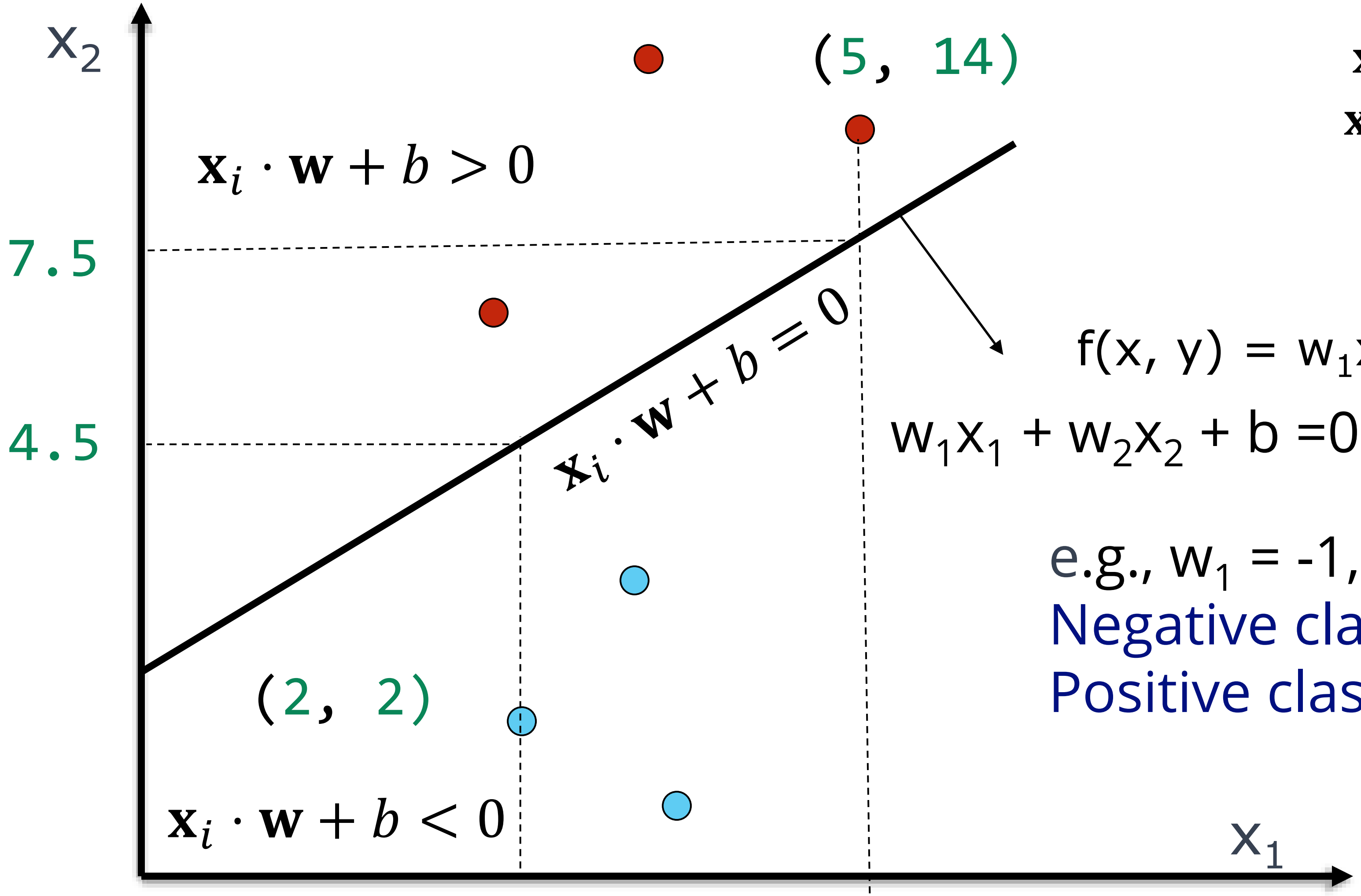
Slides by Svetlana Lazebnik
(with minor modification)

Classifiers

- Given a feature representation for images/data, how do we learn a model for distinguishing features from different classes?
- Today:
  - Linear classifiers: support vector machines

Linear Classifiers

Find linear function (*hyperplane*) to separate positive and negative examples



$\mathbf{x}_i$ positive: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative: $\mathbf{x}_i \cdot \mathbf{w} + b < 0$

$f(x, y) = w_1 x_1 + w_2 x_2 + b$

$w_1 x_1 + w_2 x_2 + b = 0$ or $x_2 = -(w_1/w_2)\,x_1 - (b/w_2)$

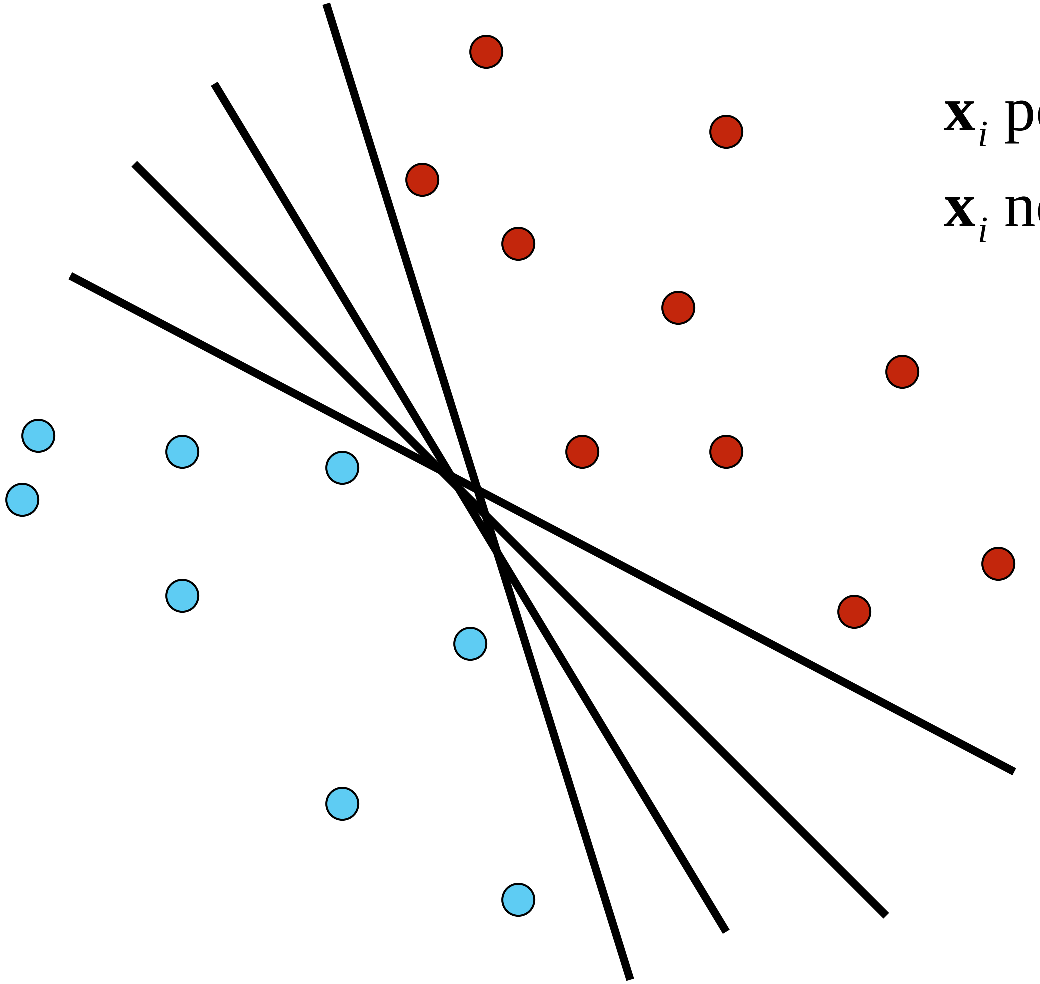
e.g., $w_1 = -1$, $w_2 = 1$, $b = -2.5$

Negative class data = [(2, 2)]

Positive class data = [(5, 14)]

Linear Classifiers

Find linear function (*hyperplane*) to separate positive and negative examples



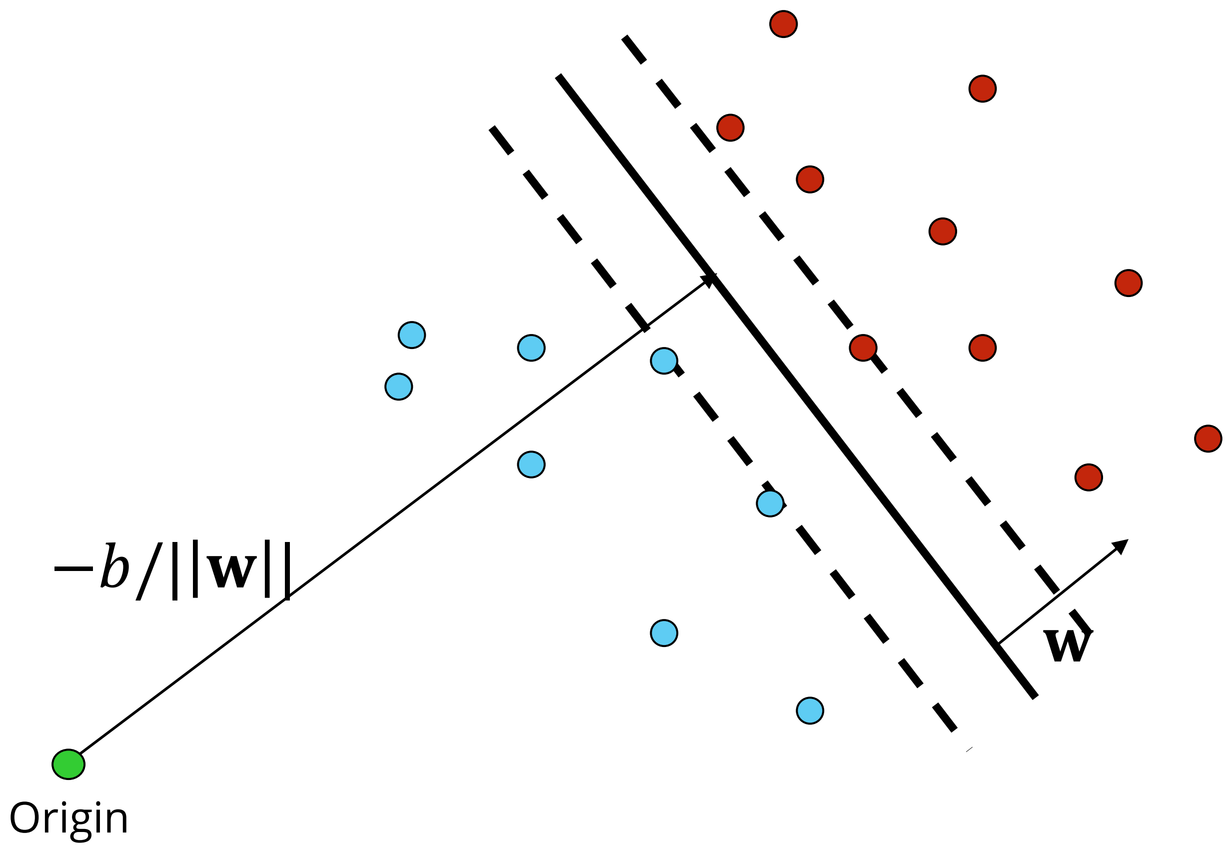
$\mathbf{x}_i$ positive : $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$

$\mathbf{x}_i$ negative : $\quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$

Which hyperplane is best?

Support Vector Machines

Find hyperplane that maximizes the *margin* between the positive (say y_i=1) and negative (say y_i=-1) examples

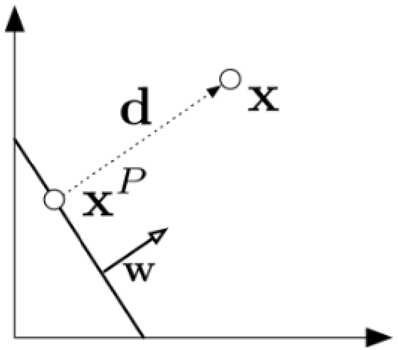


$\mathbf{x}_i$ positive $(y_i = 1)^*$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

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

$\Rightarrow y_i\,(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0\ \forall$ i

distance of a point to the hyperplane



$$\frac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$$

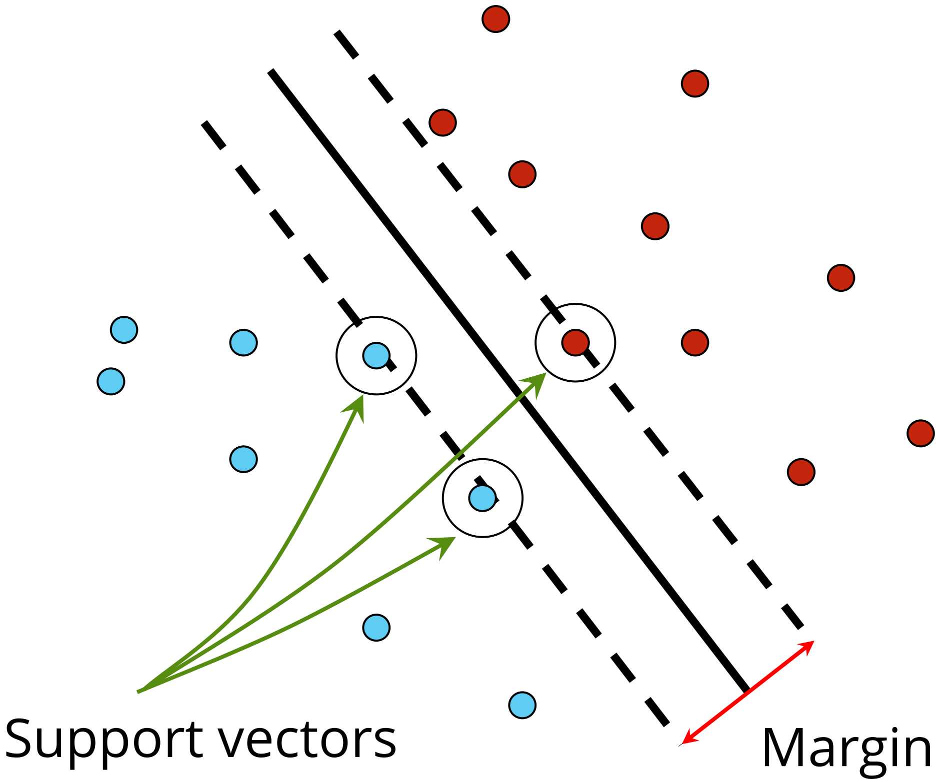
(for derivation see, e.g., here)

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

* we replace 0 with 1 and -1 By setting the thresholds at -1 and 1, it ensures that the data points on the margin, which are called support vectors, have a margin of exactly 1 unit from the decision boundary

Support Vector Machines

Find hyperplane that maximizes the *margin* between the positive (say y_i=1) and negative (say y_i=-1) 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 $2 / \|\mathbf{w}\|$ for support vectors

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:

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

can be solve efficiently with QCQP (Quadratically Constrained Quadratic Program) solver

Objective-quadratic

constraints are linear

we made use of the fact $f(z)=z^2$ is a monotonically increasing function for $z \geq 0$

Finding the Maximum-Margin Hyperplane

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

Learned weight
(nonzero only for support vectors)

•Support vectors are critical elements of the training set in SVM.

•They are the data points that lie closest to the decision boundary.

•If all other training points were removed or moved within their respective classes, the same separating hyperplane would be found during the training process.

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*

- Classification function (decision boundary):

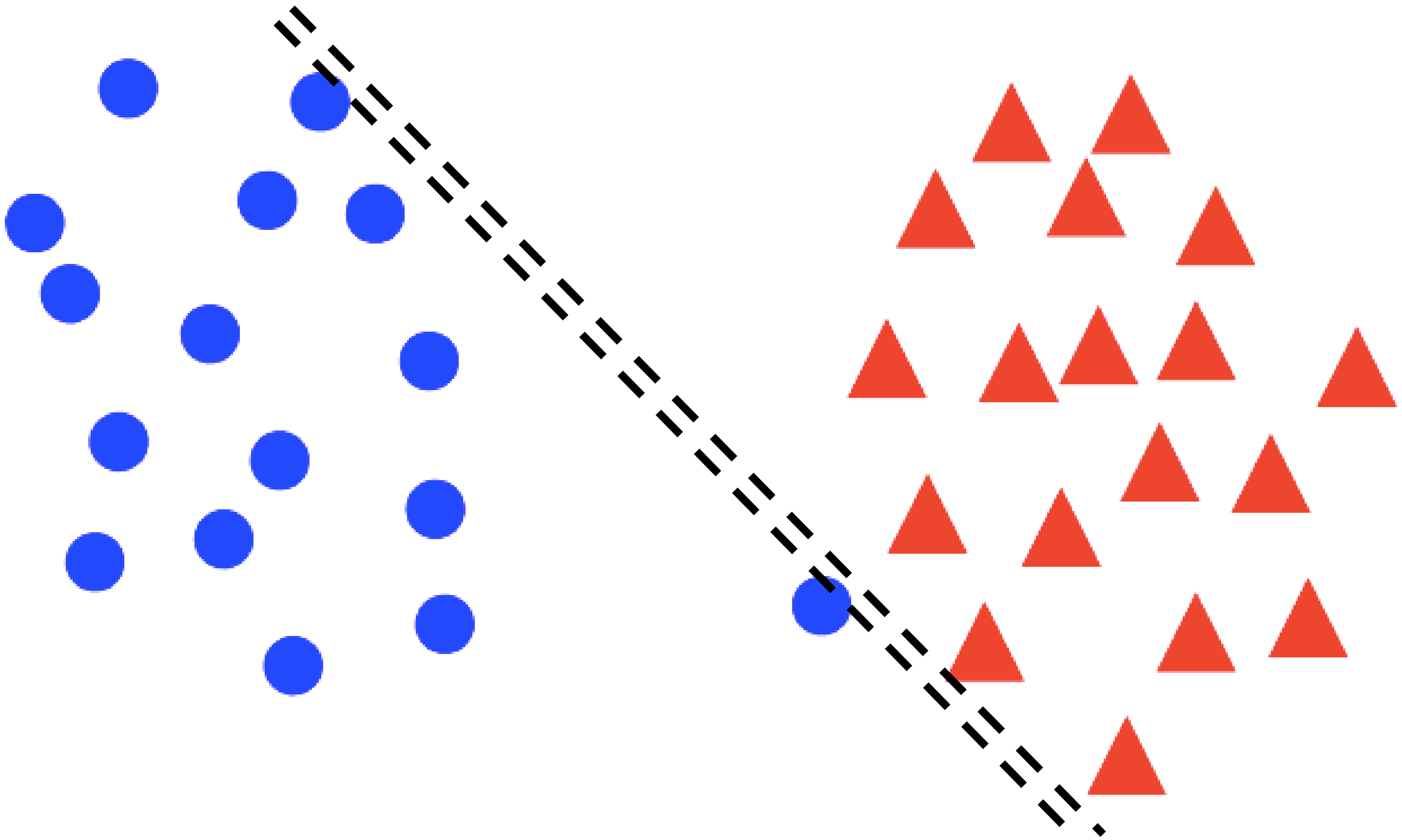
$$\mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point $\mathbf{x}$ and the support vectors $\mathbf{x}_i$
- Solving the optimization problem also involves computing the inner products $\mathbf{x}_i \cdot \mathbf{x}_j$ between all pairs of training points

*it is safe to take the mean value of b resulting from all support vectors

"Soft margin" formulation

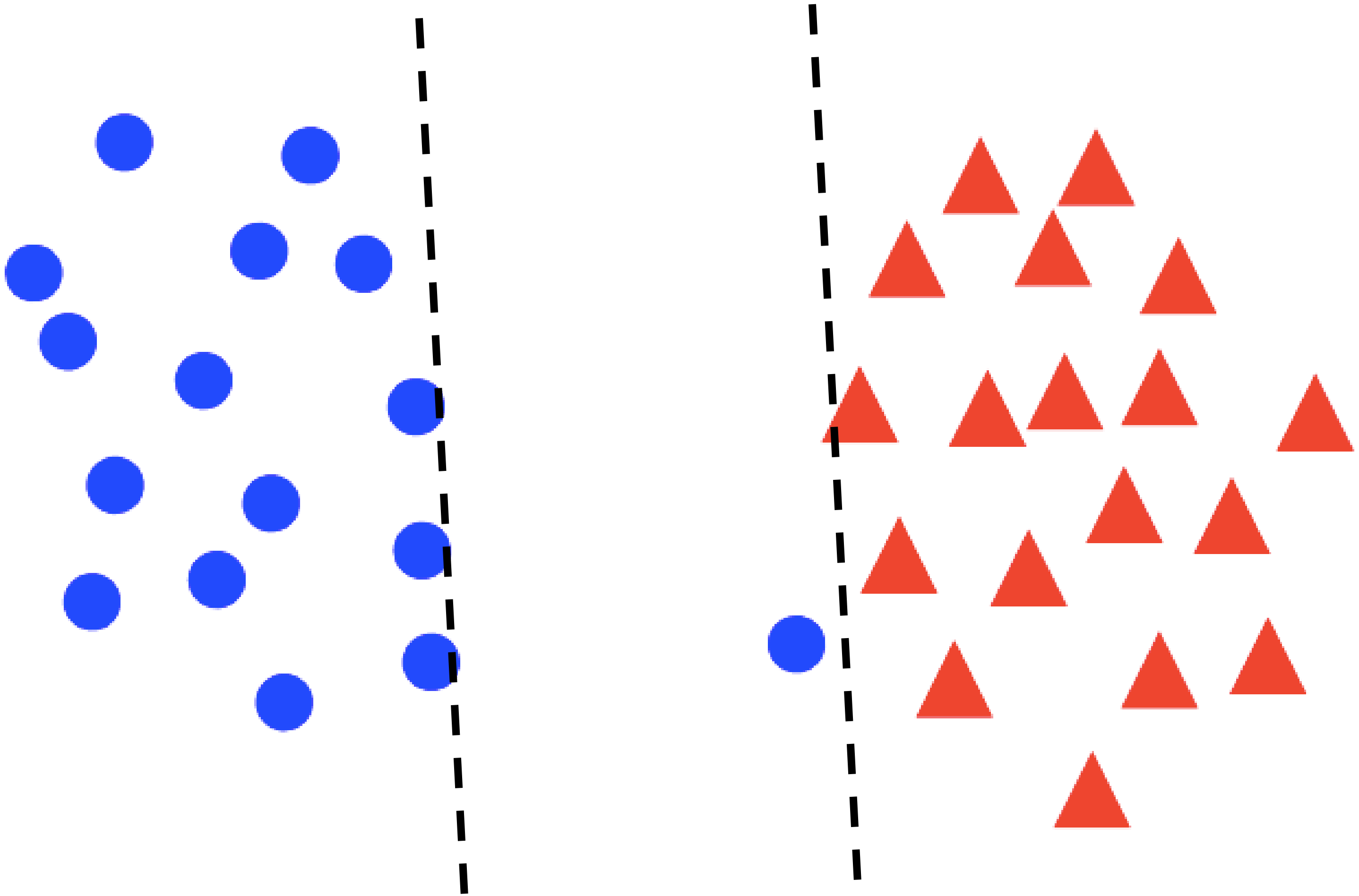
- What about non-separable data?
- And even for separable data, we may prefer a larger margin with a few constraints violated



Source

"Soft margin" formulation

- What about non-separable data?
- And even for separable data, we may prefer a larger margin with a few constraints violated



Source

What if the Data is Not Linearly Separable?

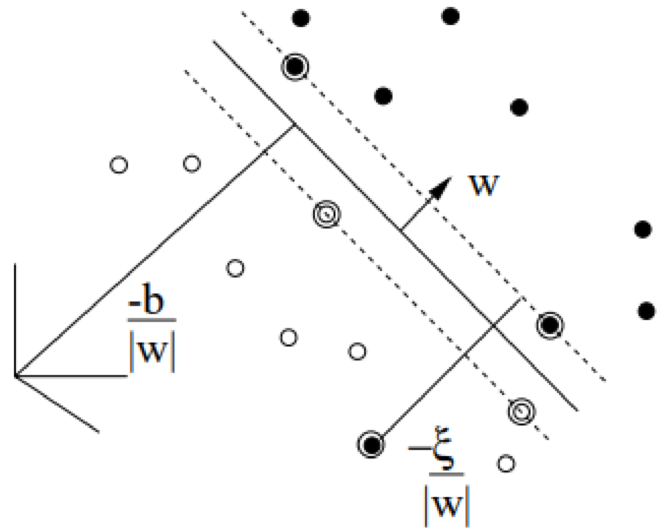
- Separable: $$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

- Idea

  Non-separable: change the constrain $y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i$

  Minimize this reduction $\sum_{i=1}^{n} \xi_i$

  do we need to do this for all?

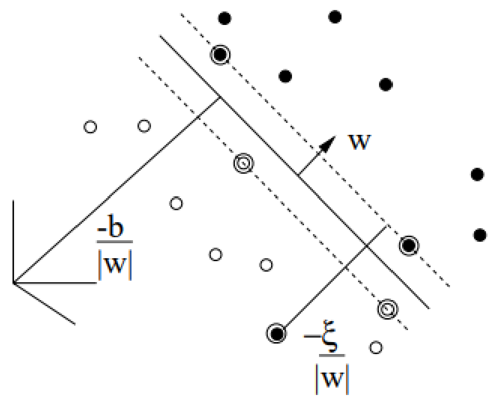


What if the Data is Not Linearly Separable?

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

- Non-separable: $$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$ $$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

- C: tradeoff constant, ξ_i: *slack variable* (positive)
- **Whenever margin is ≥ 1, $\xi_i = 0$**
- Whenever margin is < 1, $$\xi_i = 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)$$



What if the Data is Not Linearly Separable?

Non-separable

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

$$\min_{\mathbf{w},b} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\boxed{\text{Maximize margin}}} + C \underbrace{\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\right)}_{\boxed{\text{Minimize classification mistakes}}}$$

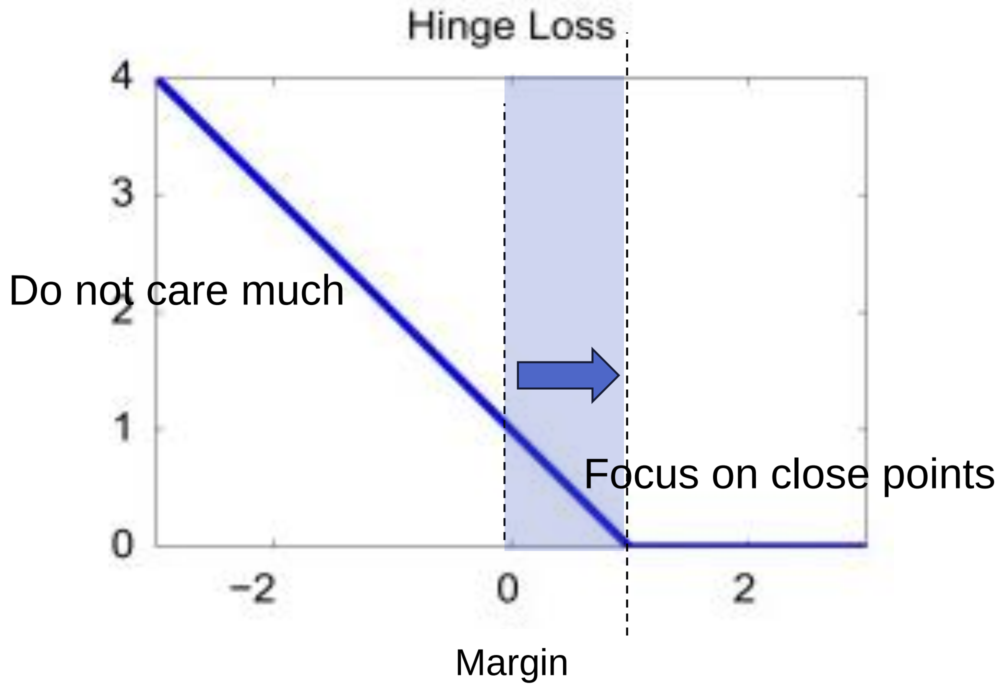
more mistakes

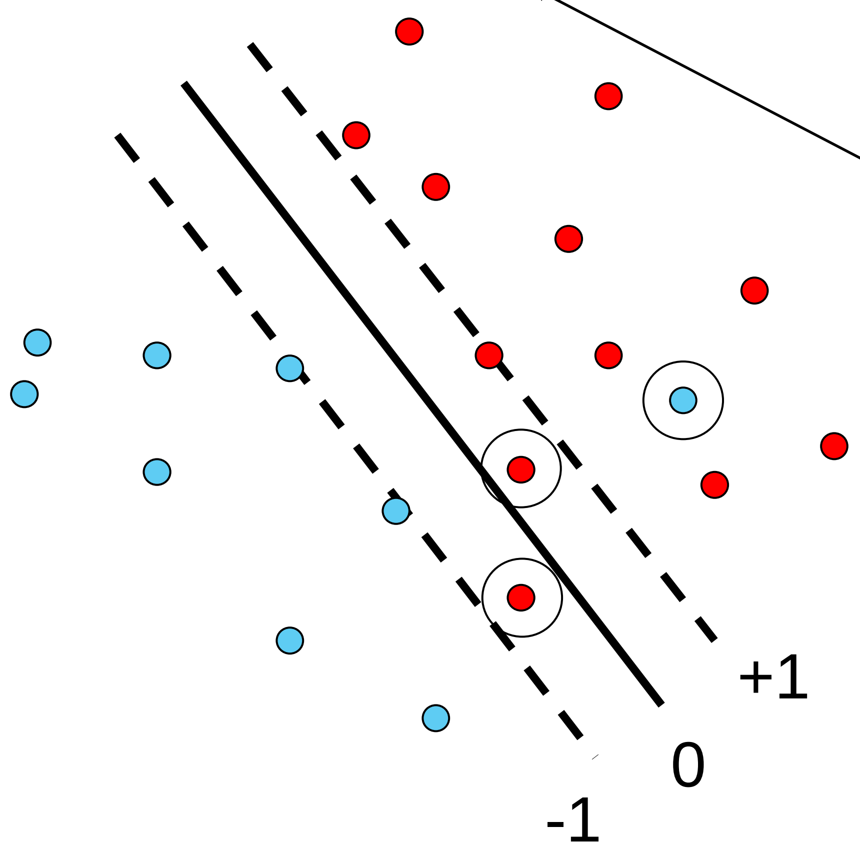
What if the Data is Not Linearly Separable?

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

$\max(0, 1-y_i(\mathbf{w}\cdot\mathbf{x}_i+b))$

The hinge loss function is designed to penalize misclassified points and encourage the SVM to maximize the margin between the classes.

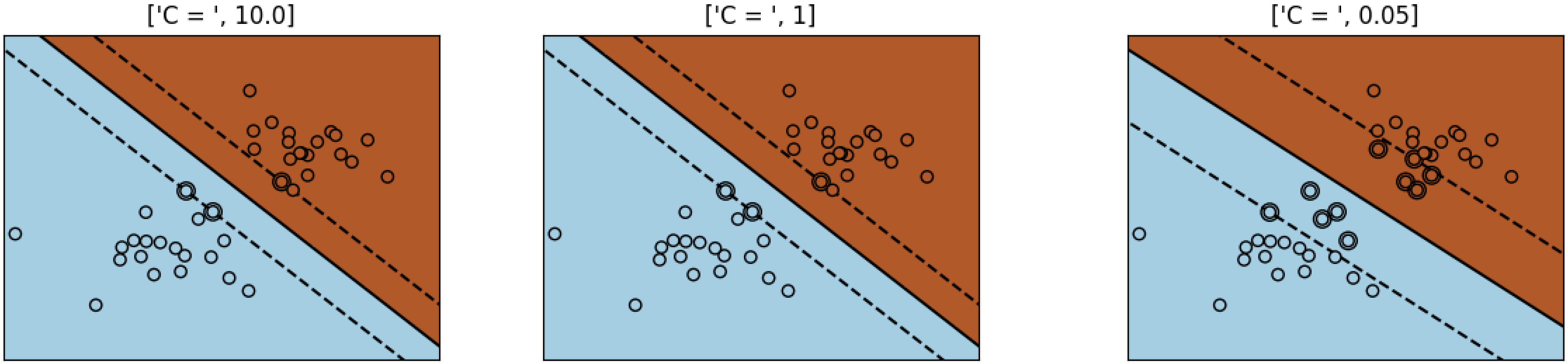




Demo: http://cs.stanford.edu/people/karpathy/svmjs/demo

What if the Data is Not Linearly Separable?

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



“A small value of C includes more/all the observations, allowing the margins to be calculated using all the data in the area.” SVM Margins Example — scikit-learn 1.3.0 documentation

SGD update for SVM

$$l(w, x_i, y_i) = \frac{\lambda}{2n}\|w\|^2 + \max[0, 1 - y_i w^T x_i]$$

$$\nabla l(w, x_i, y_i) = \frac{\lambda}{n} w - \mathbb{I}[y_i w^T x_i < 1] y_i x_i$$

Recall: $\frac{d}{da}\max(0, a) = \mathbb{I}[a > 0]$

SGD update for SVM

$$l(w, x_i, y_i) = \frac{\lambda}{2n}\|w\|^2 + \max[0, 1 - y_i w^T x_i]$$

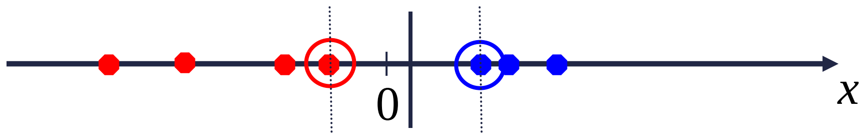
$$\nabla l(w, x_i, y_i) = \frac{\lambda}{n} w - \mathbb{I}[y_i w^T x_i < 1] y_i x_i$$

- SGD update:
  - If $y_i w^T x_i \geq 1$: $w \leftarrow w - \eta \frac{\lambda}{n} w$
  - If $y_i w^T x_i < 1$: $w \leftarrow w + \eta \left(y_i x_i - \frac{\lambda}{n} w \right)$

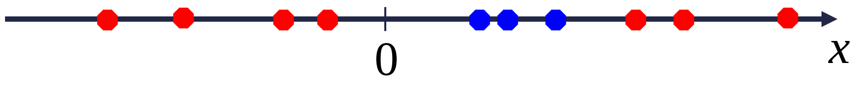
S. Shalev-Schwartz et al., Pegasos: Primal Estimated sub-GrAdient SOlver for SVM, *Mathematical Programming*, 2011

Nonlinear SVMs

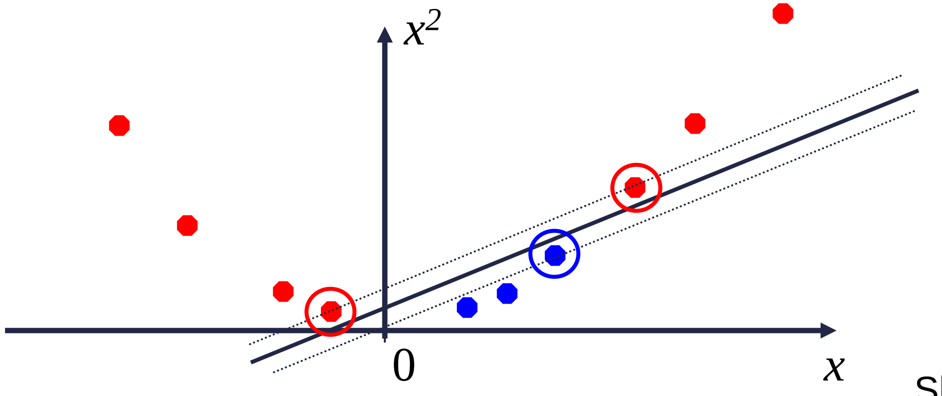
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



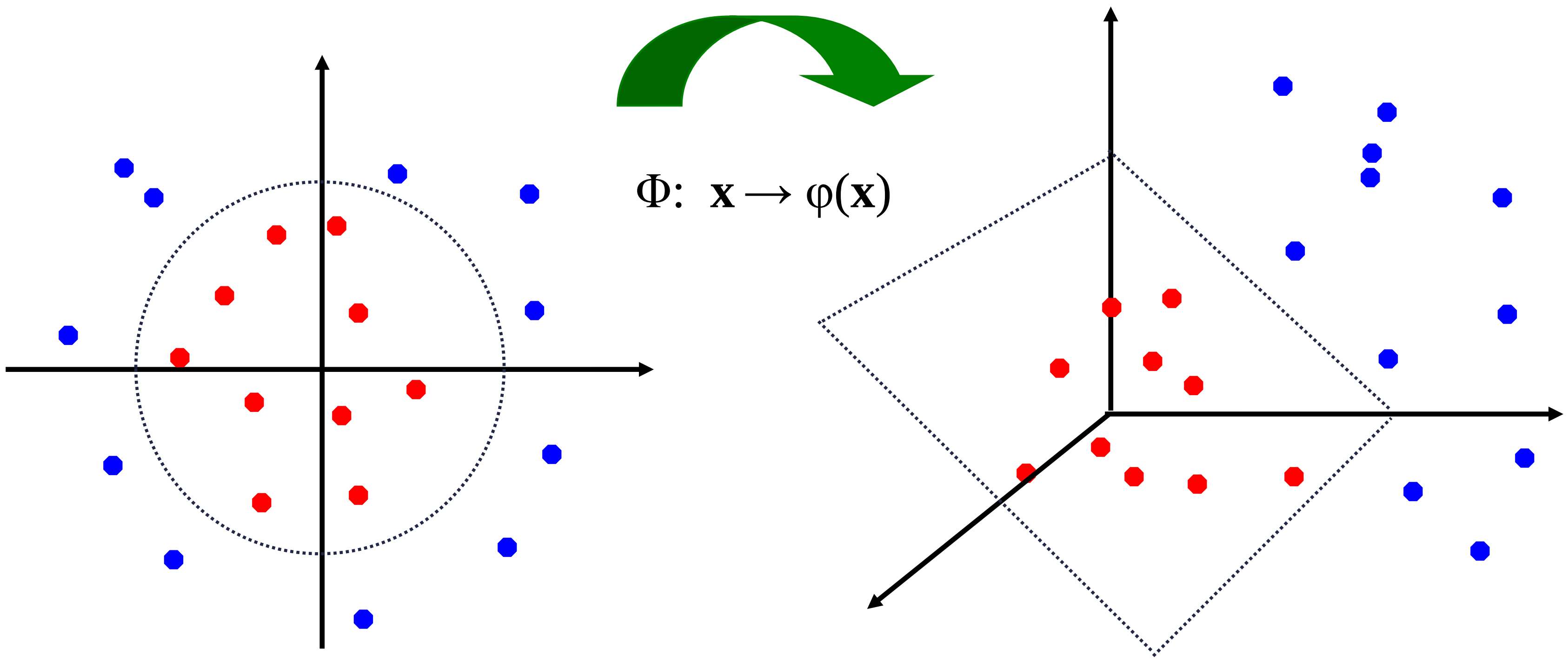
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

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



Slide credit: Andrew Moore

Nonlinear SVMs

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

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

(to be valid, the kernel function must satisfy *Mercer's condition*)

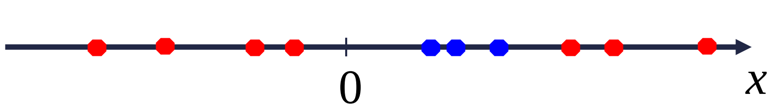
- This gives a nonlinear decision boundary in the original feature space:

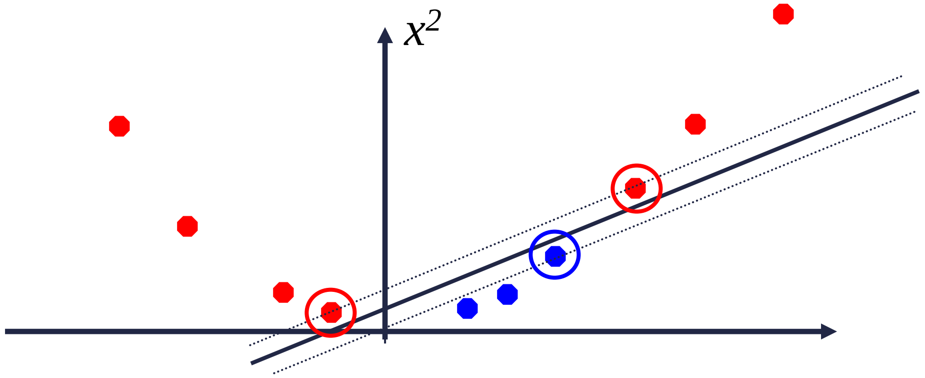
$$\sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

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

Nonlinear Kernel: Example

- Consider the mapping $\varphi(x) = (x, x^2)$



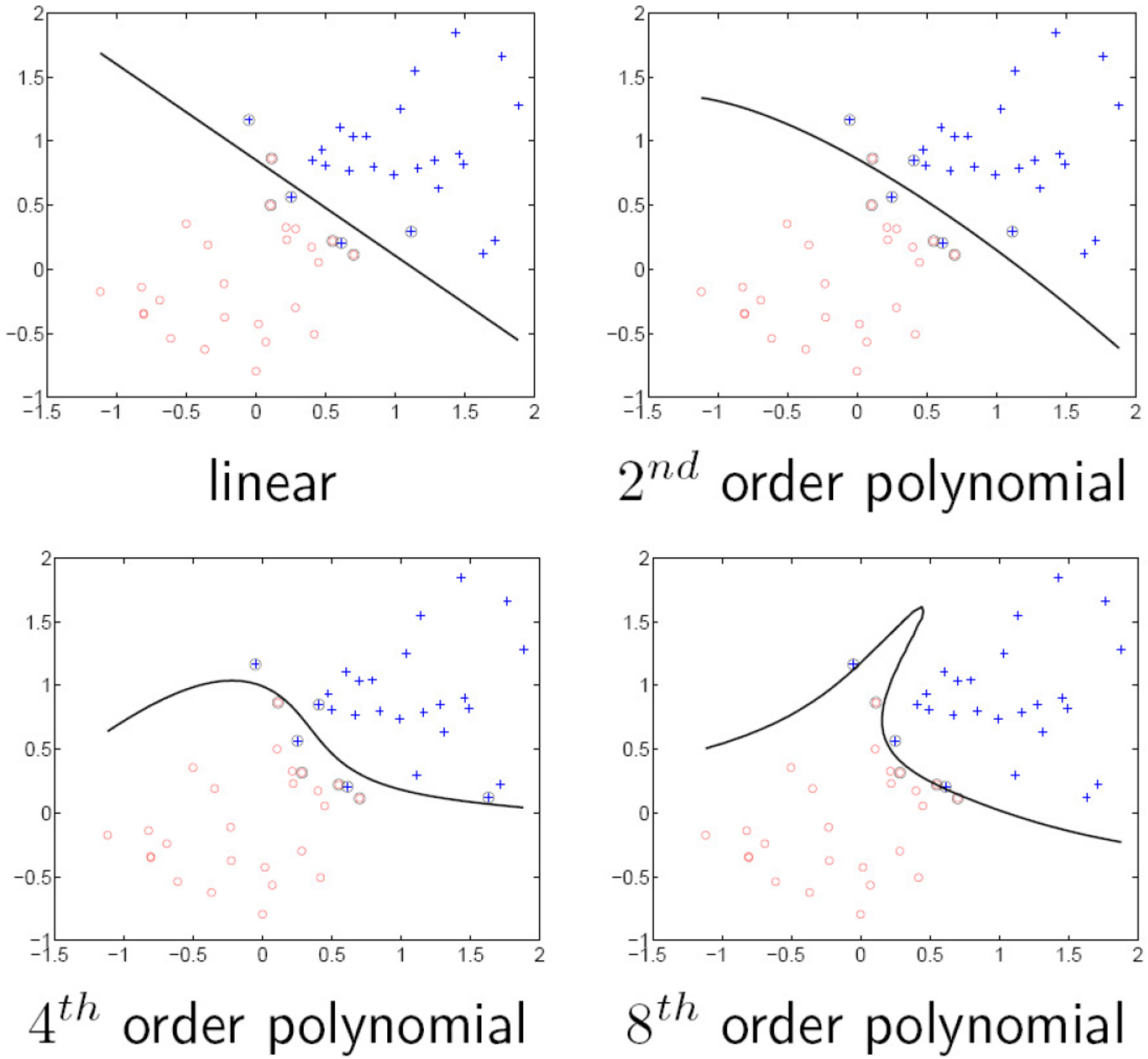


$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

Polynomial Kernel:

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



Higher values of d allow SVM to fit complex and intricate decision boundaries, but they may also lead to overfitting if not appropriately tuned.

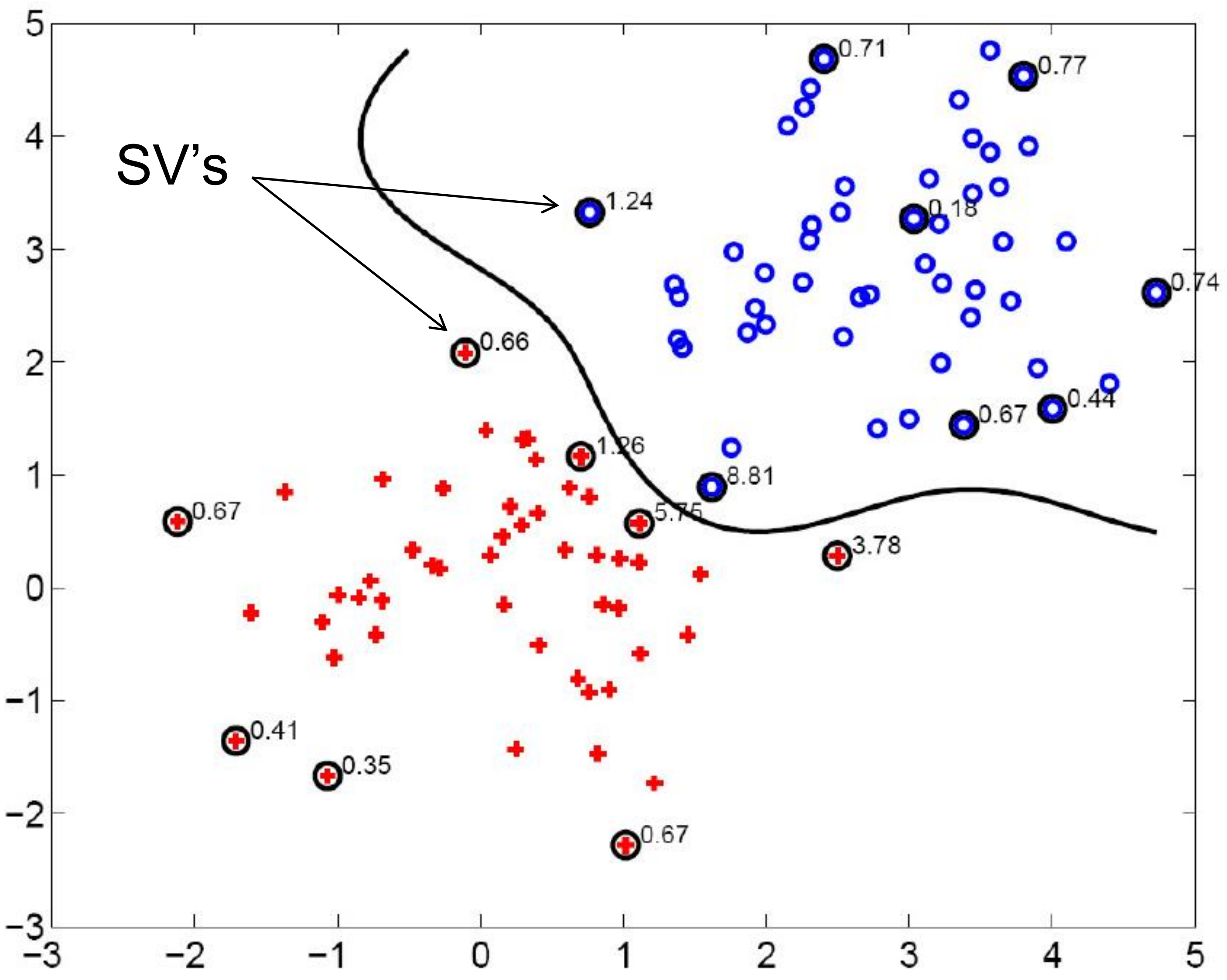
Gaussian Kernel

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

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

- The corresponding mapping $\varphi(\mathbf{x})$ is infinite-dimensional!

Gaussian Kernel



Gaussian Kernel

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

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

- The corresponding mapping $\boldsymbol{\varphi}(\mathbf{x})$ is infinite-dimensional!
- What is the role of parameter $\boldsymbol{\sigma}$?
  - What if $\boldsymbol{\sigma}$ is close to zero?
  - What if $\boldsymbol{\sigma}$ is very large?

A small σ can lead to overfitting (The decision boundary will be highly sensitive to individual data points), while a large σ can result in underfitting. Large σ Gaussian kernel becomes more spread out and smoother. The kernel assigns similar weights to a larger region around each data point. A larger σ results in a more generalized decision boundary, and the SVM model is less likely to overfit the training data. However, too large a value of σ may lead to the decision boundary becoming too smooth, potentially underfitting the data and not capturing the intricacies of the data distribution

Kernels for Bags of Features

- Histogram intersection kernel:

$$I(\mathbf{h}_1, \mathbf{h}_2) = \sum_{i=1}^{N} \min(\mathbf{h}_1(i), \mathbf{h}_2(i))$$

- Hellinger kernel: $K(\mathbf{h}_1, \mathbf{h}_2) = \sum_{i=1}^{N} \sqrt{\mathbf{h}_1(i)\,\mathbf{h}_2(i)}$

- Generalized Gaussian kernel:

$$K(\mathbf{h}_1, \mathbf{h}_2) = \exp\left(-\frac{1}{A} D(\mathbf{h}_1, \mathbf{h}_2)^2\right)$$

  - *D* can be L1, Euclidean, χ^2 distance, etc.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study, IJCV 2007

Summary: SVMs for Image Classification

1. Pick an image representation (in our case, bag of features)
2. Pick a kernel function for that representation
3. Compute the matrix of kernel values between every pair of training examples
4. Feed the kernel matrix into your favorite SVM solver to obtain support vectors and weights
5. At test time: compute kernel values for your test example and each support vector, and combine them with the learned weights to get the value of the decision function

What about Multi-Class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. rest (ovr)
  - Training: learn an SVM for each class vs. the others
  - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value
- One vs. one (ovo)
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM “votes” for a class to assign to the test example

SVMs: Pros and Cons

- Pros
  - Effective in high dimensional spaces.
  - Still effective in cases where number of dimensions is greater than the number of samples.
  - Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
  - Many publicly available SVM packages: http://www.kernel-machines.org/software
  - Kernel-based framework is very powerful, flexible.
  - SVMs work very well in practice, even with very small training sample sizes
- Cons
  - No "direct" multi-class SVM, must combine two-class SVMs.
  - If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
  - SVMs do not directly provide probability estimates.
  - Computation, memory
    - During training time, must compute matrix of kernel values for every pair of examples
    - Learning can take a very long time for large-scale problems

Material from https://scikit-learn.org/stable/modules/svm.html as well.

SVMs for Large-Scale Datasets

- Efficient *linear* solvers
  - LIBLINEAR, PEGASOS
- Explicit approximate embeddings: define an explicit mapping $\boldsymbol{\varphi}(\mathbf{x})$ such that $\boldsymbol{\varphi}(\mathbf{x}) \cdot \boldsymbol{\varphi}(\mathbf{y})$ approximates $K(\mathbf{x}, \mathbf{y})$ and train a linear SVM on top of that embedding
  - Random Fourier features for the Gaussian kernel (Rahimi and Recht, 2007)
  - Embeddings for additive kernels, e.g., histogram intersection (Maji et al., 2013, Vedaldi and Zisserman, 2012)

Summary: Classifiers

- Nearest-neighbor and k-nearest-neighbor classifiers
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - Non-separable case
  - The kernel trick
  - Multi-class SVMs
  - Large-scale SVMs
- Of course, there are many other classifiers out there
  - Neural networks, boosting, decision trees/forests, …