CS 2750: Machine Learning

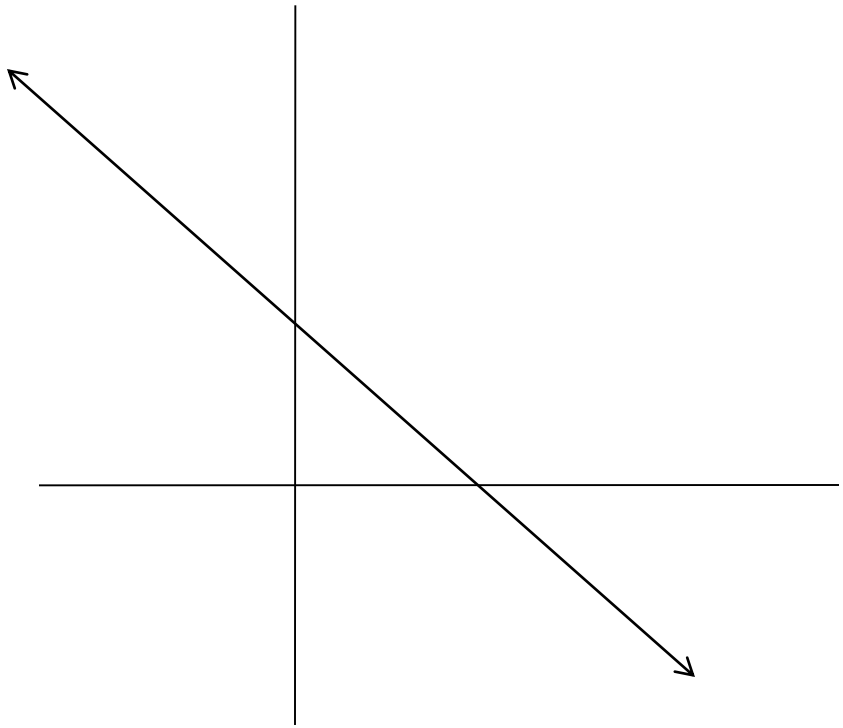
Support Vector Machines

Prof. Adriana Kovashka
University of Pittsburgh
February 16, 2017

Plan for today

- Linear Support Vector Machines
- Non-linear SVMs and the “kernel trick”
- Extensions (briefly)
  - Soft-margin SVMs
  - Multi-class SVMs
  - Hinge loss
  - SVM vs logistic regression
  - SVMs with latent variables
- How to solve the SVM problem (next class)

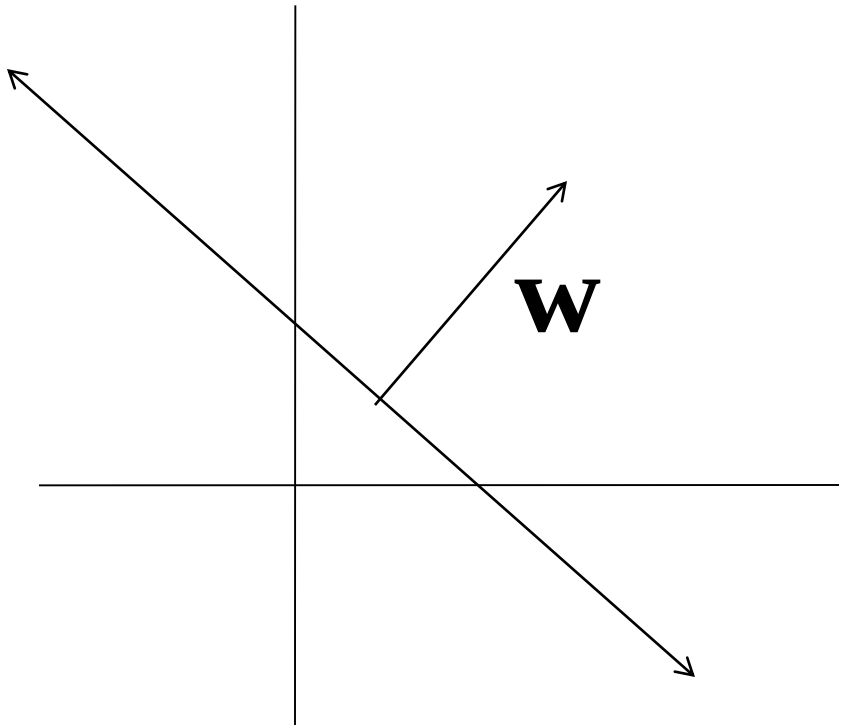
Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

Lines in R^2



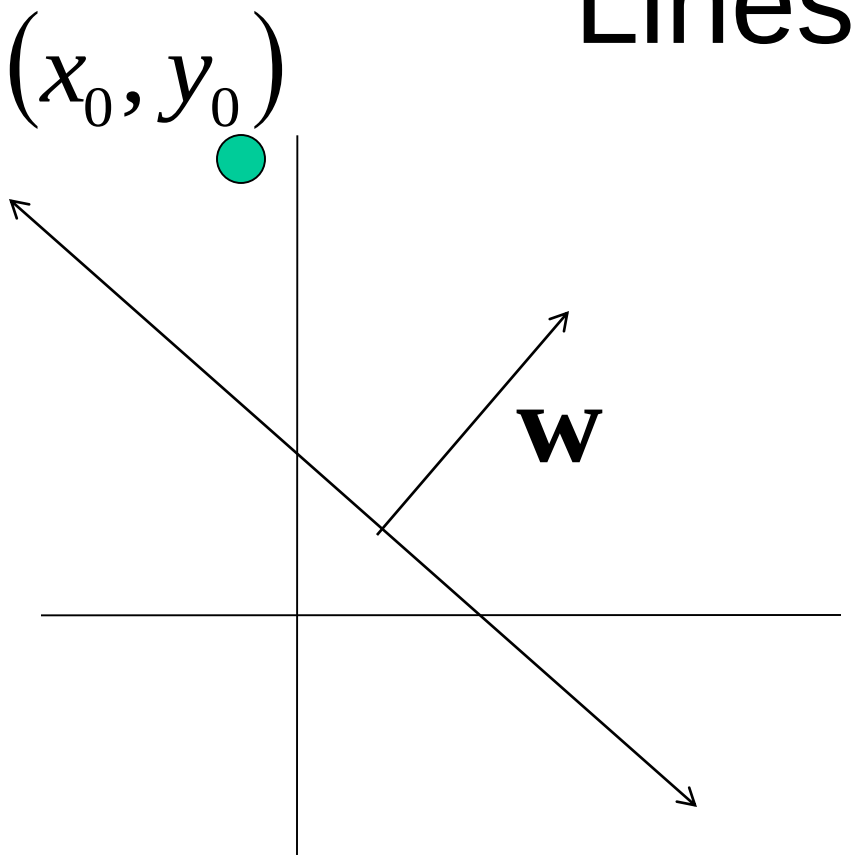
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Lines in R^2



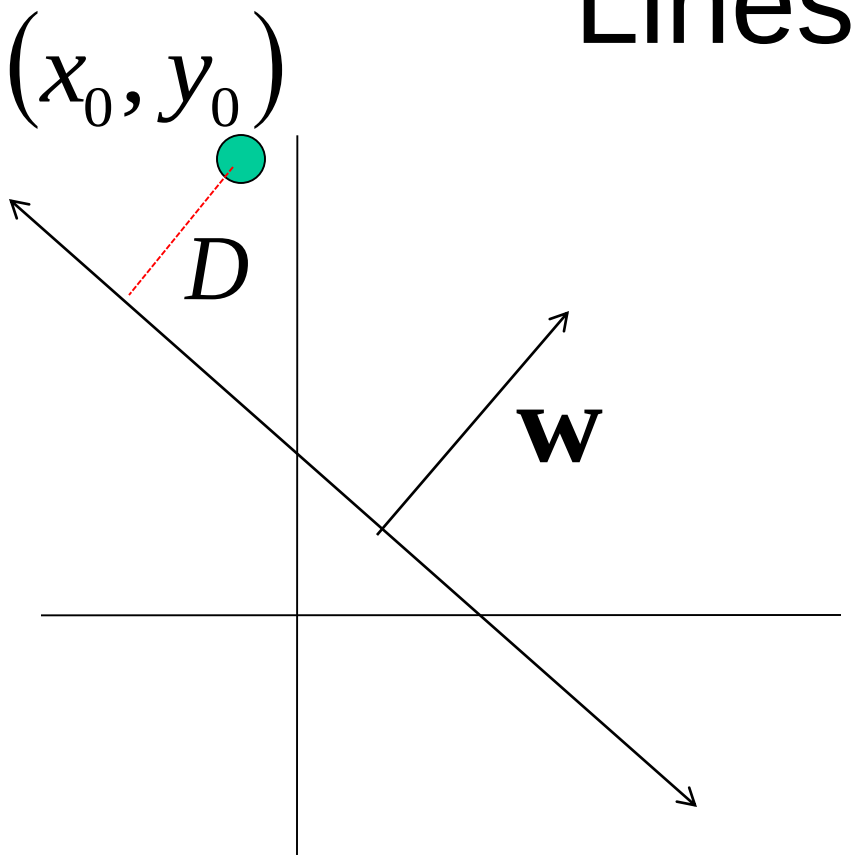
Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

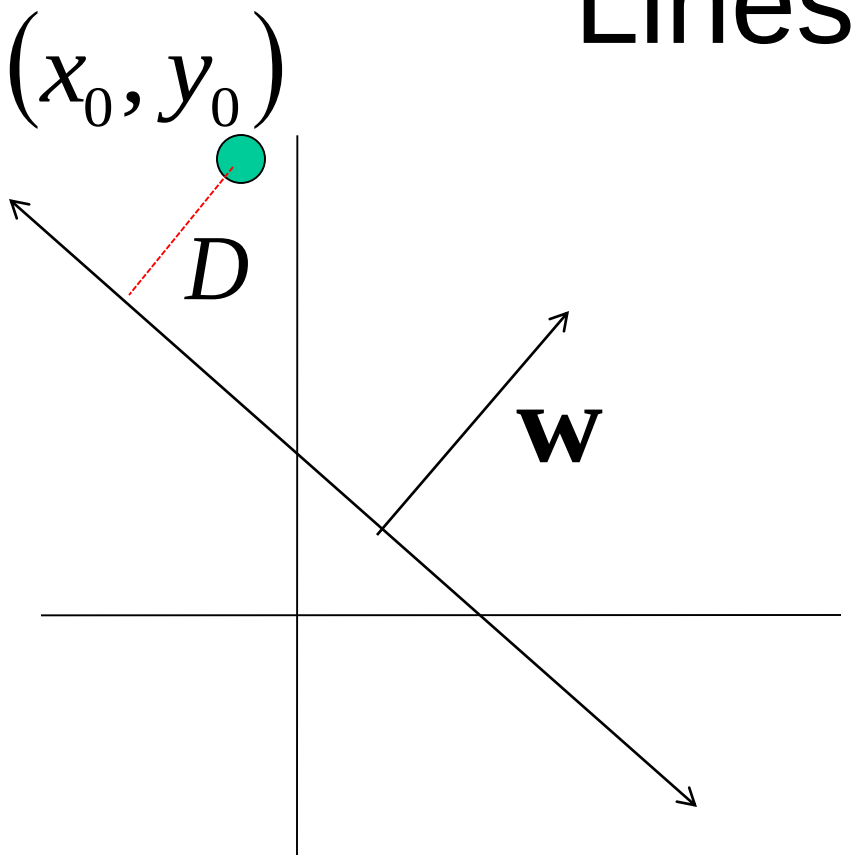
$$\updownarrow$$

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}}$$

distance from point to line

Lines in R^2



Let $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$ax + cy + b = 0$$

$$\updownarrow$$

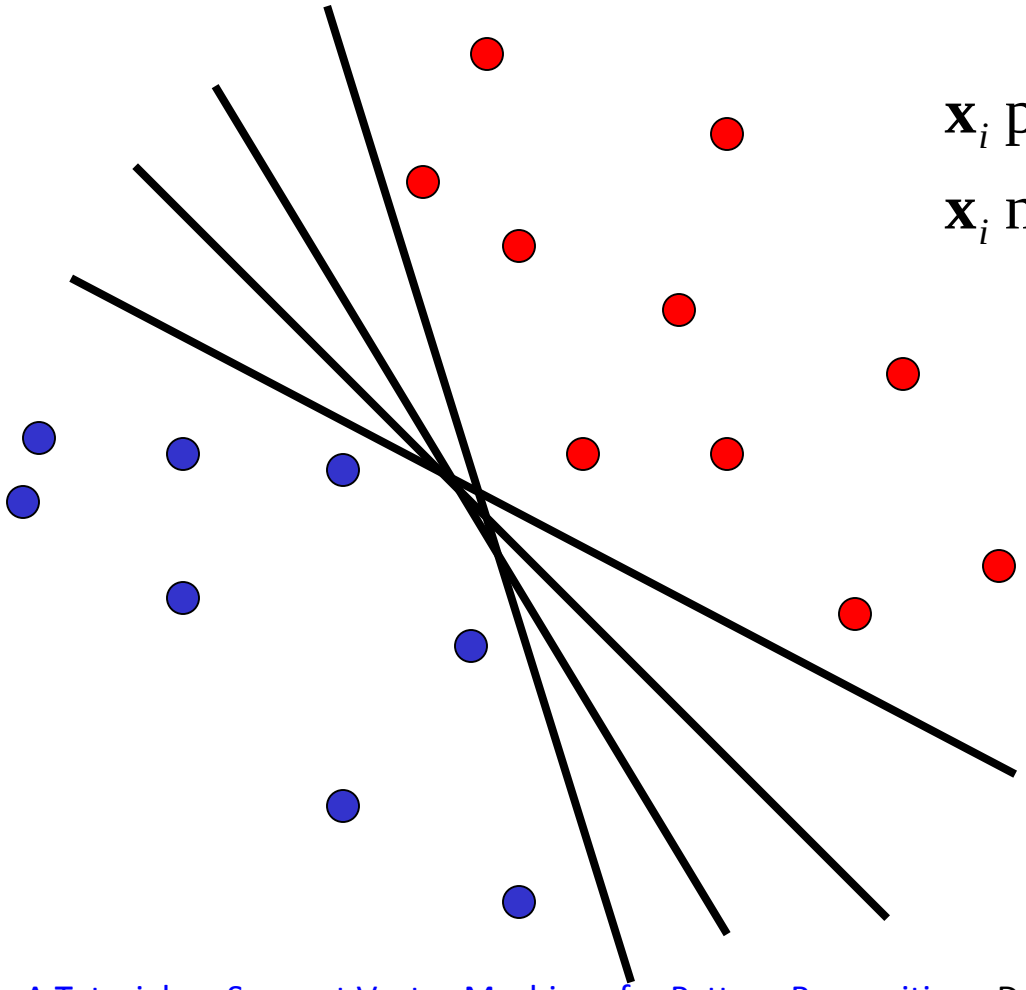
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

distance from point to line

Linear classifiers

- Find linear function to separate positive and negative examples



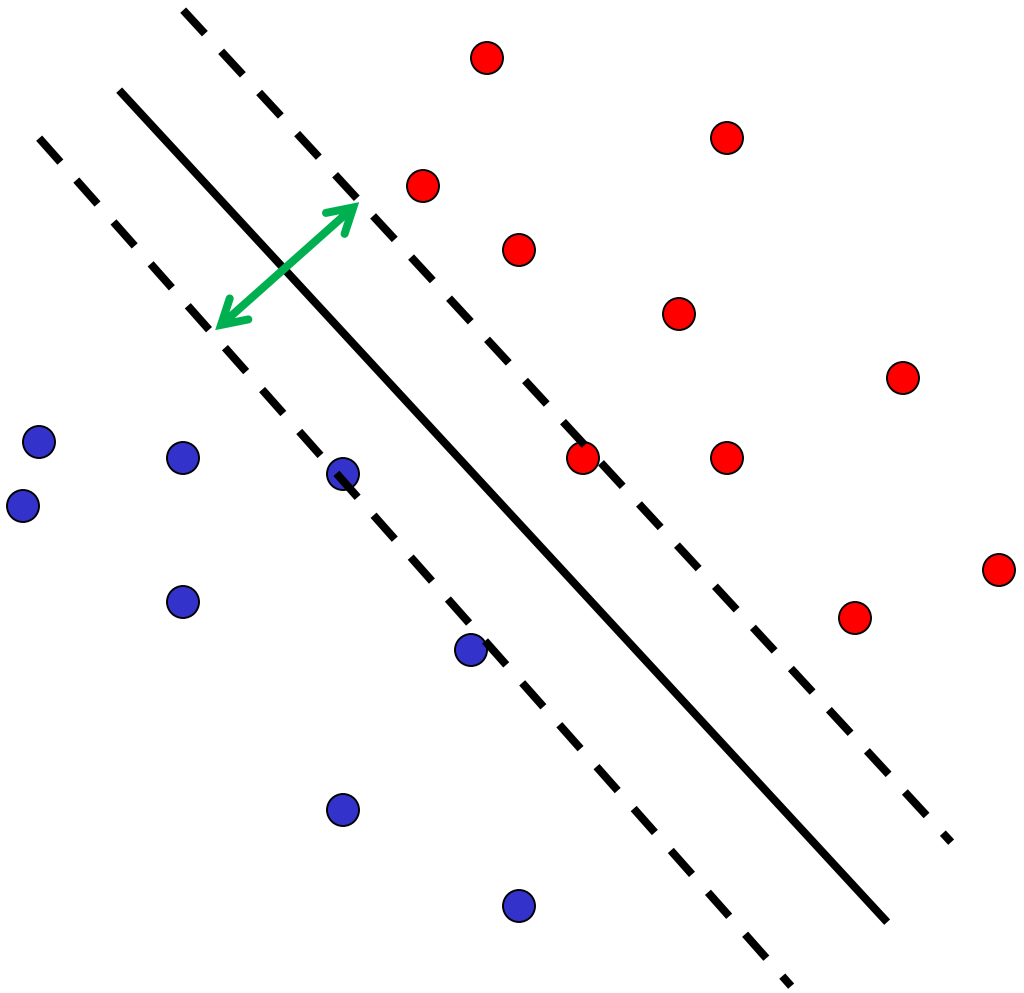
$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$

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

Which line is best?

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

Support vector machines

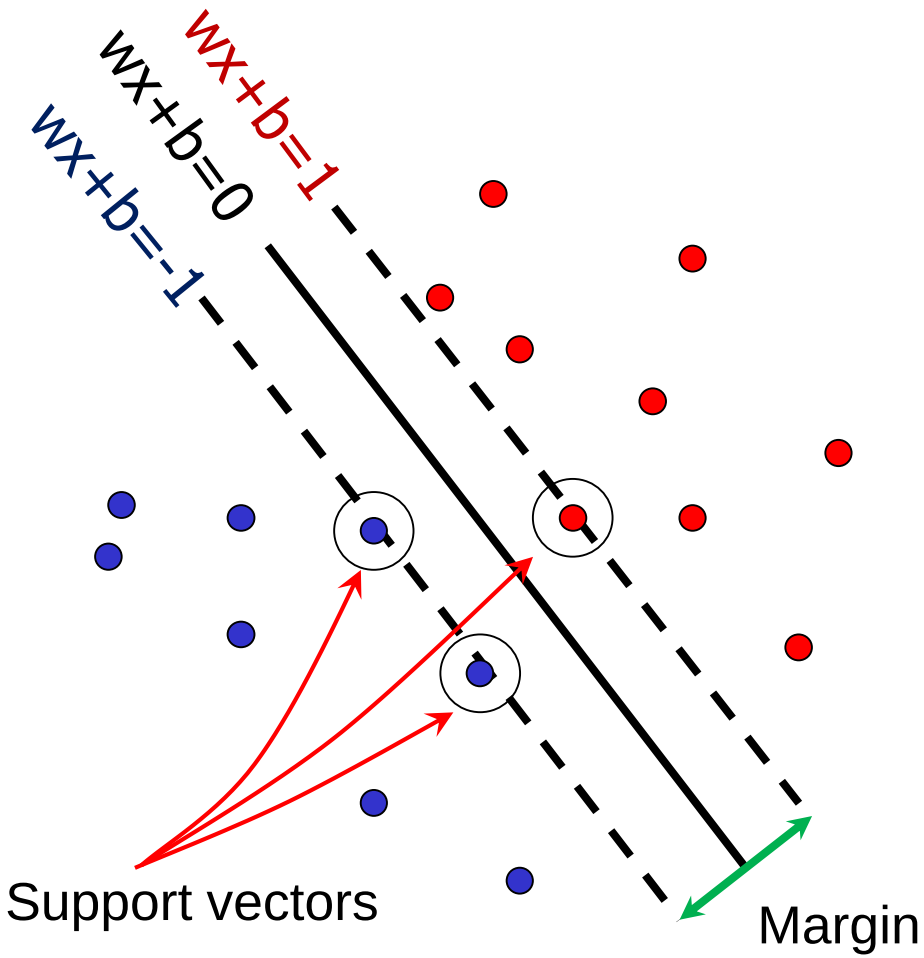


- Discriminative classifier based on *optimal separating line (for 2d case)*
- Maximize the *margin* between the positive and negative training examples

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

Support vector machines

- Want line that maximizes the margin.



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

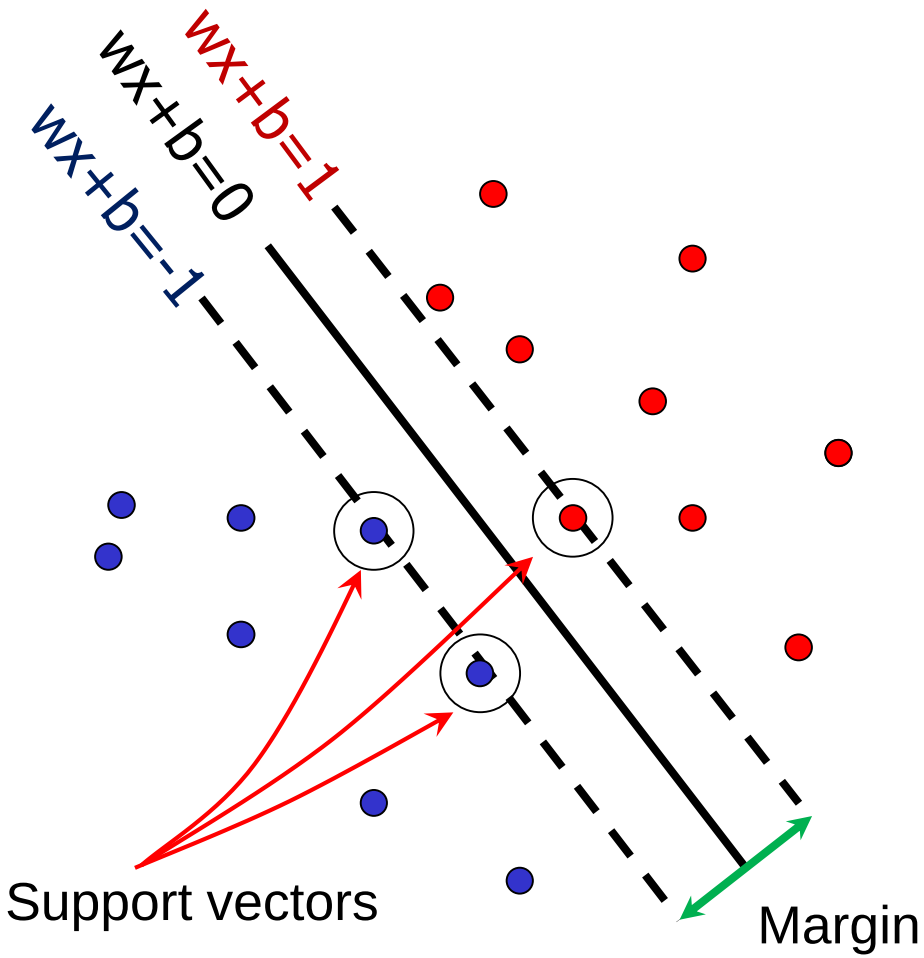
$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

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

- Want line that maximizes the margin.



$\mathbf{x}_i$ 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 line: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

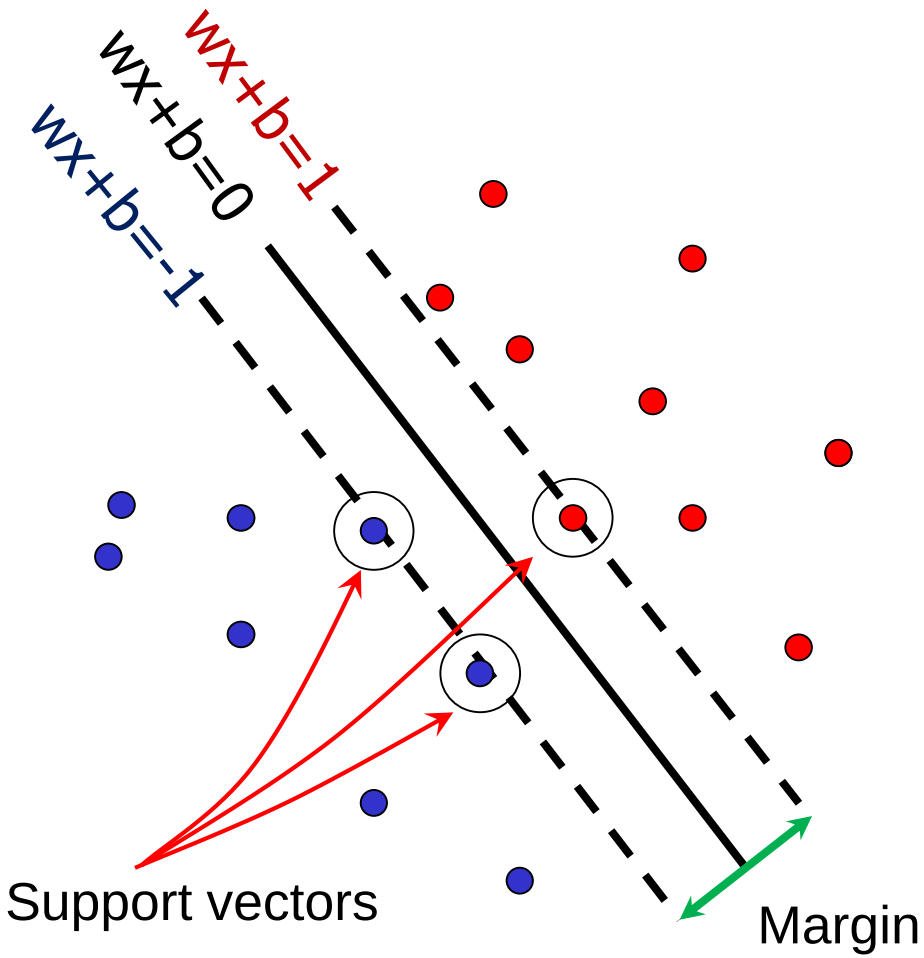
For support vectors:

$$\frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|} = \frac{\pm 1}{\|\mathbf{w}\|} \qquad M = \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$$

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

Support vector machines

- Want line that maximizes the margin.



$\mathbf{x}_i$ 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 line: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

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 line

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

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

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

Quadratic optimization problem:

$$\text{Minimize } \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$\text{Subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

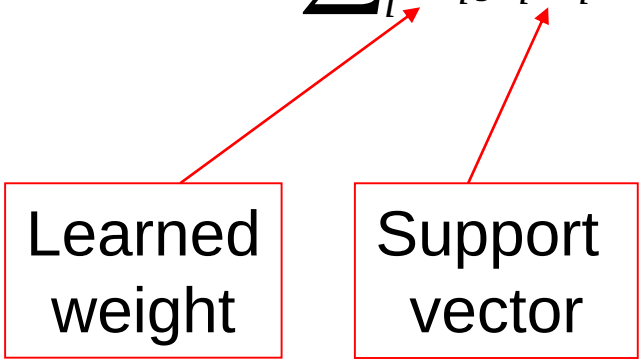
One constraint for each training point.

Note sign trick.

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

Finding the maximum margin line

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



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

Finding the maximum margin line

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

MORE DETAILS NEXT TIME

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

- Classification function:

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

$$= \text{sign}\left(\sum_{\text{i}} \alpha_{\text{i}} y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

If f(x) < 0, classify as negative, otherwise classify as positive.

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

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

Inner product

- **The decision boundary for the SVM and its optimization depend on the inner product of two data points (vectors):**

$$\mathbf{x}_i^T \mathbf{x}_j$$

$$f(x) = \text{sign}\ (\mathbf{w} \cdot \mathbf{x} + \mathrm{b})$$
$$= \text{sign}\left(\sum_{\mathrm{i}} \alpha_{\mathrm{i}} y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

- **The inner product is equal**

$$(\mathbf{x}_i^T \mathbf{x}) = \|\mathbf{x}_i\| * \|\mathbf{x}_i\| \cos\theta$$

If the angle in between them is 0 then: $(\mathbf{x}_i^T \mathbf{x}) = \|\mathbf{x}_i\| * \|\mathbf{x}_i\|$

If the angle between them is 90 then: $(\mathbf{x}_i^T \mathbf{x}) = 0$

The inner product measures how similar the two vectors are

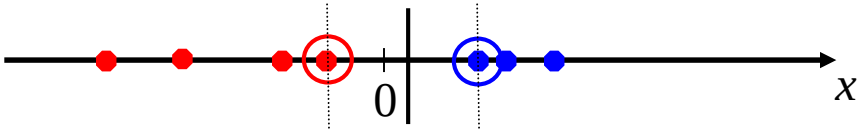
Adapted from Milos Hauskrecht

Plan for today

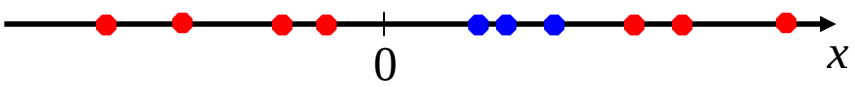
- Linear Support Vector Machines
- Non-linear SVMs and the “kernel trick”
- Extensions (briefly)
  - Soft-margin SVMs
  - Multi-class SVMs
  - Hinge loss
  - SVM vs logistic regression
  - SVMs with latent variables
- How to solve the SVM problem (next class)

Nonlinear SVMs

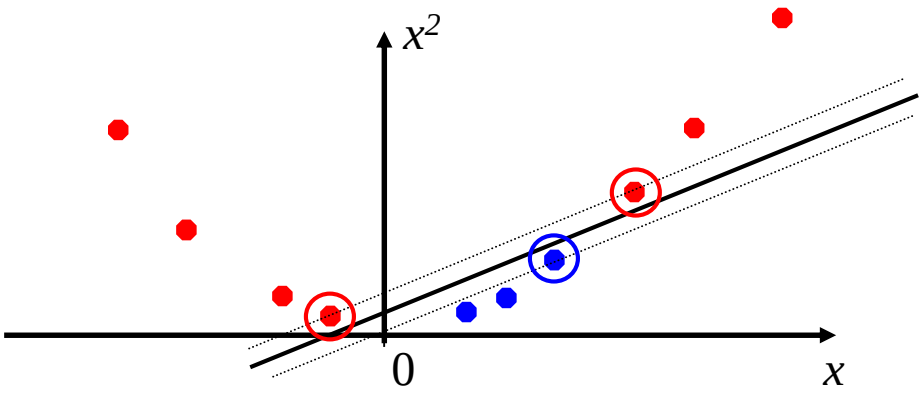
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

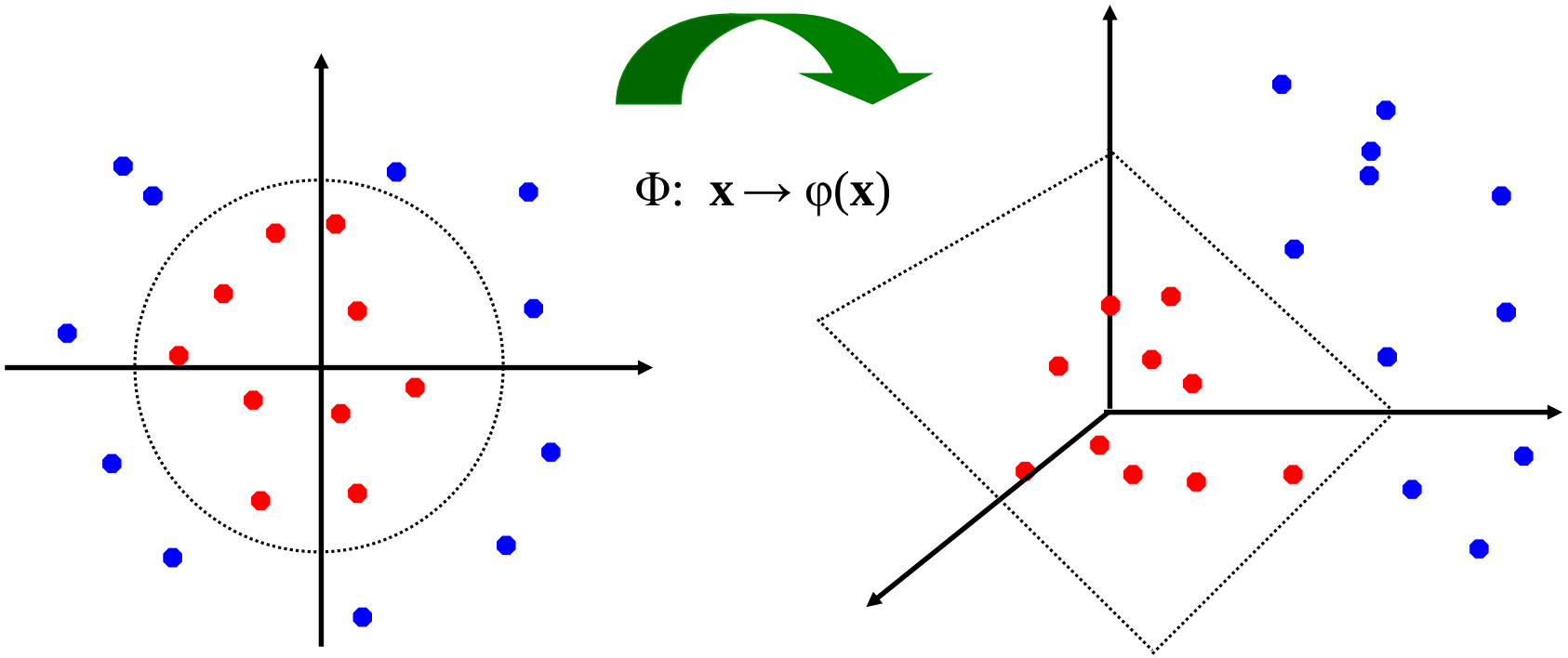


- We can map it to a higher-dimensional space:



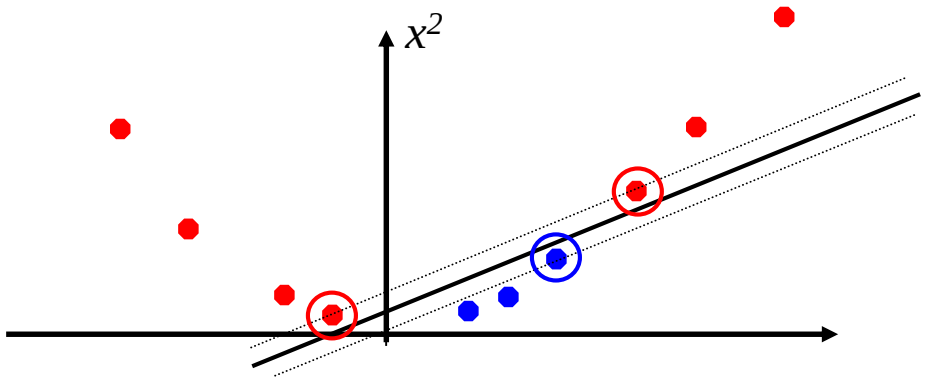
Nonlinear SVMs

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



Nonlinear kernel: Example

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



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

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

The “kernel trick”

- The linear classifier relies on dot product between vectors $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x}_i \rightarrow \boldsymbol{\varphi}(\mathbf{x}_i)$, the dot product becomes: $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$
- A *kernel function* is similarity function that corresponds to an inner product in some expanded feature space
- *The kernel trick*: instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that: $K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$

Examples of kernel functions

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomials of degree up to *d*:

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

- Gaussian RBF:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

- Histogram intersection:

$$K(x_i, x_j) = \sum_k \min(x_i(k), x_j(k))$$

The benefit of the “kernel trick”

- Example: Polynomial kernel for 2-dim features

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{z}) &= \left(1 + \mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (1 + x_1 z_1 + x_2 z_2)^2 \\
&= 1 + 2x_1 z_1 + 2x_2 z_2 + x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\
&= (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2)(1, \sqrt{2}z_1, \sqrt{2}z_2, z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\
&= \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{z}). \qquad (7.42)
\end{aligned}
$$

- … lives in 6 dimensions

Is this function a kernel?

Problem:

- Checking if a given $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ fulfills the conditions for a kernel is *difficult*:
- We need to prove or disprove

$$\sum_{i,j=1}^{n} t_i k(x_i, x_j) t_j \geq 0.$$

for *any set* $x_1, \ldots, x_n \in \mathcal{X}$ *and any* $t \in \mathbb{R}^n$ *for any* $n \in \mathbb{N}$.

Workaround:

- It is easy to *construct* functions k that are positive definite kernels.

Constructing kernels

1) We can *construct kernels from scratch*:

- For any $\varphi : \mathcal{X} \to \mathbb{R}^m$, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathbb{R}^m}$ is a kernel.
- If $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *distance function*, i.e.
  - $d(x, x') \geq 0$ for all $x, x' \in \mathcal{X}$,
  - $d(x, x') = 0$ only for $x = x'$,
  - $d(x, x') = d(x', x)$ for all $x, x' \in \mathcal{X}$,
  - $d(x, x') \leq d(x, x'') + d(x'', x')$ for all $x, x', x'' \in \mathcal{X}$,

  then $k(x, x') := \exp(-d(x, x'))$ is a kernel.

2) We can *construct kernels from other kernels*:

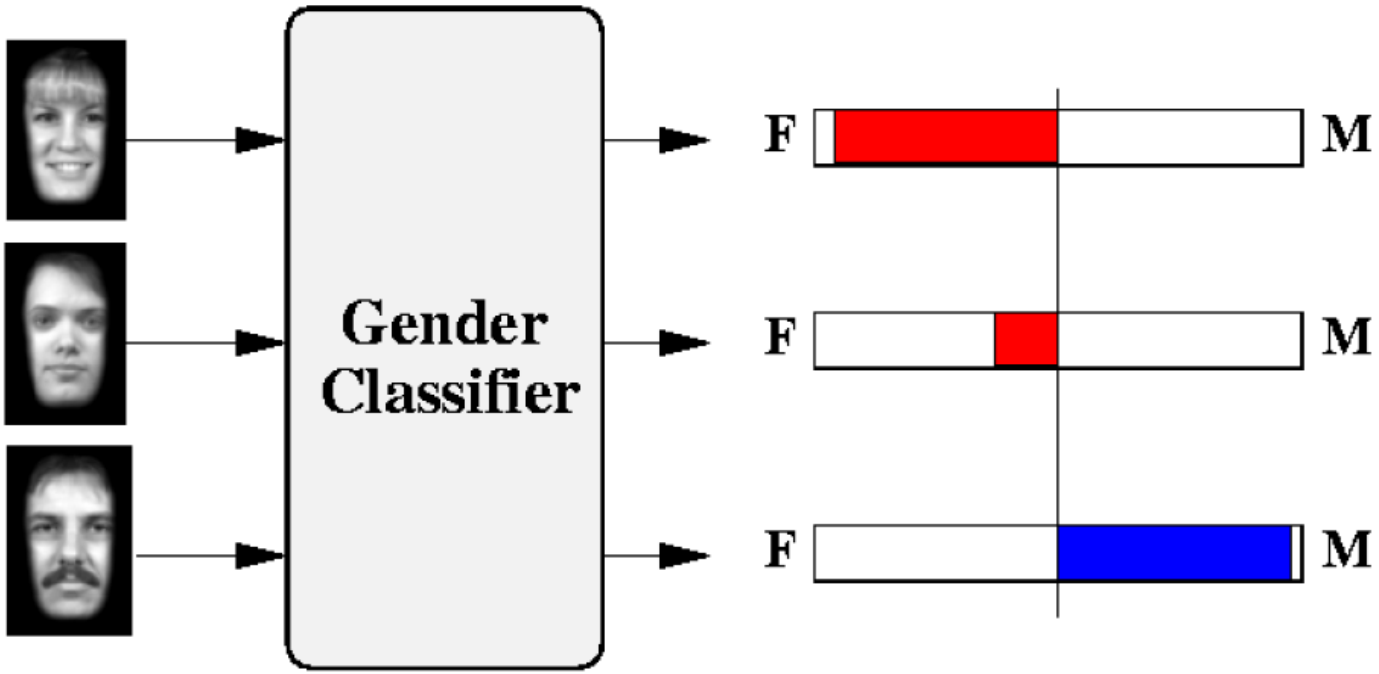
- if k is a kernel and $\alpha > 0$, then αk and $k + \alpha$ are kernels.
- if k_1, k_2 are kernels, then $k_1 + k_2$ and $k_1 \cdot k_2$ are kernels.

Using SVMs

1. Define your representation for each example.
2. Select a kernel function.
3. Compute pairwise kernel values between labeled examples.
4. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.
5. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.

Adapted from Kristen Grauman

Example: Learning gender w/ SVMs

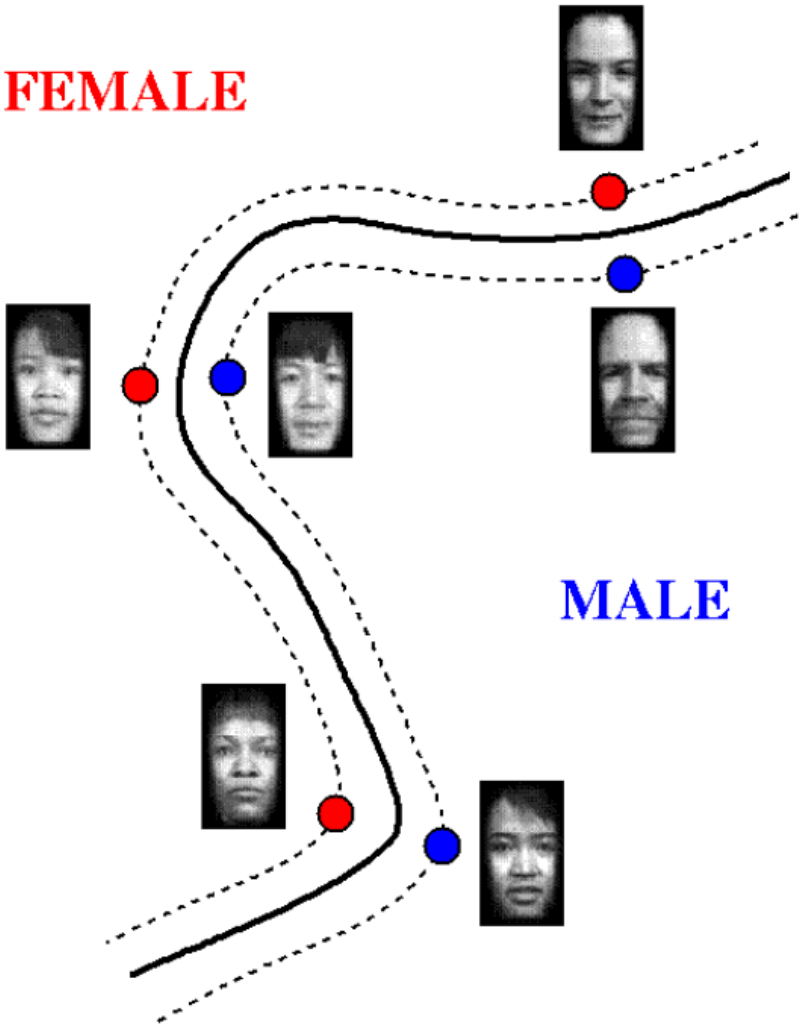


Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002

Moghaddam and Yang, Face & Gesture 2000

Example: Learning gender w/ SVMs

Support faces



Example: Learning gender w/ SVMs

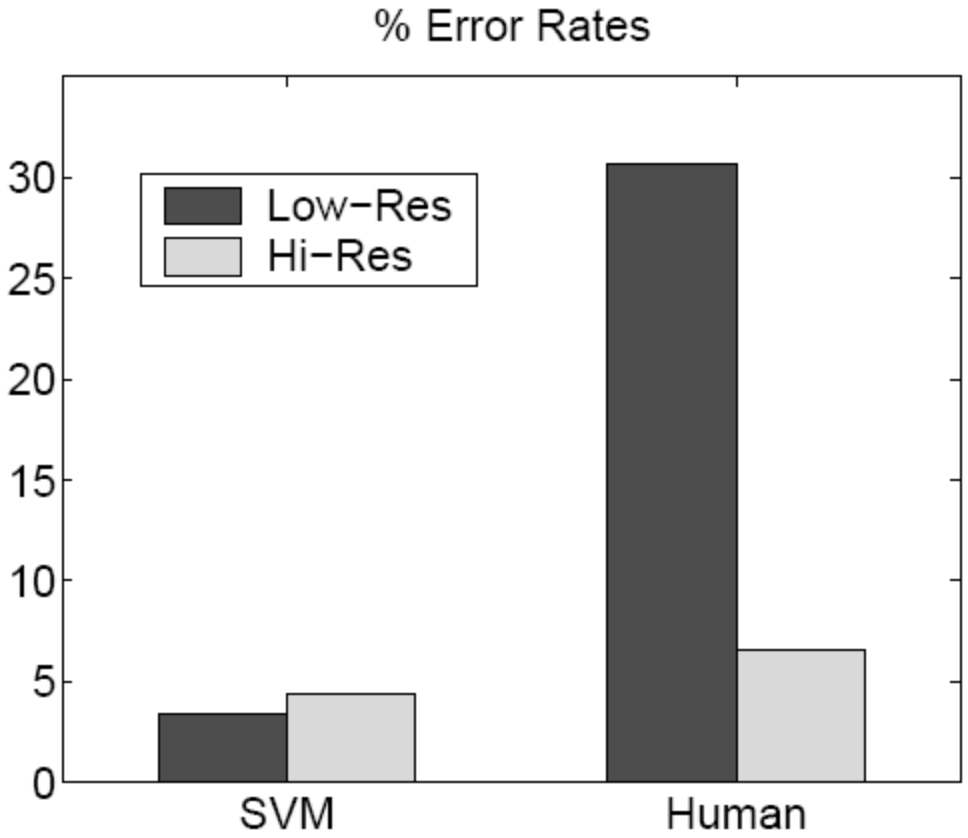


Figure 6. SVM vs. Human performance

- SVMs performed better than humans, at either resolution

Plan for today

- Linear Support Vector Machines
- Non-linear SVMs and the “kernel trick”
- Extensions (briefly)
  - Soft-margin SVMs
  - Multi-class SVMs
  - Hinge loss
  - SVM vs logistic regression
  - SVMs with latent variables
- How to solve the SVM problem (next class)

Hard-margin SVMs

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2$$

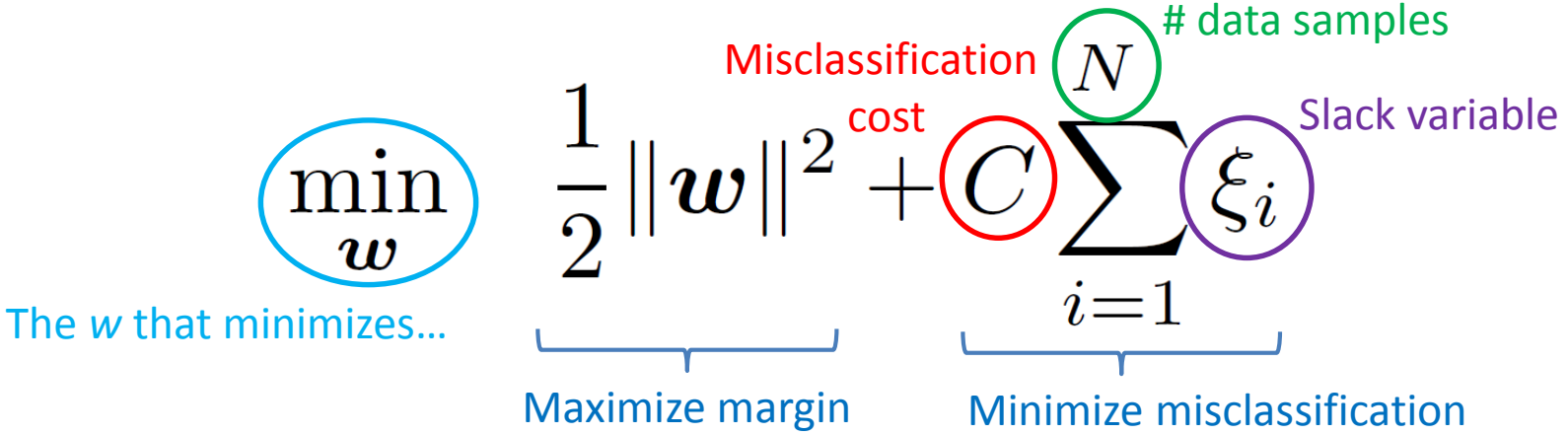
The *w* that minimizes…

Maximize margin

$$\text{subject to} \quad y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1 \quad , \quad \forall i = 1, \ldots, N$$

Soft-margin SVMs

(allowing misclassifications)



$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad y_i \boldsymbol{w}^T \boldsymbol{x}_i \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \quad \forall i = 1, \ldots, N$$

BOARD

Multi-class SVMs

- In practice, we obtain a multi-class SVM by combining two-class SVMs
- One vs. others
  - Training: learn an SVM for each class vs. the others
  - Testing: apply each SVM to the test example, and assign it to the class of the SVM that returns the highest decision value
- One vs. one
  - Training: learn an SVM for each pair of classes
  - Testing: each learned SVM "votes" for a class to assign to the test example
- There are also "natively multi-class" formulations
  - Crammer and Singer, JMLR 2001

$$\text{minimize}_{\mathbf{w},b} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C\sum_j \xi_j$$

$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1 - \xi_j, \;\; \forall y' \neq y_j, \;\; \forall j$$

$$\xi_j \geq 0, \;\; \forall j$$

Hinge loss

- Let $h(z) = max(0, 1 - z)$
- We have the objective to minimize ξ_i where:

$$y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i \qquad \xi_i \geq 1 - y_i \mathbf{w}^T \mathbf{x}_i$$
$$\xi_i \geq 0$$
$$\xi_i \geq max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$$

- Then we can define a loss:

$$h(y_i \mathbf{w}^T \mathbf{x}_i) = max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$$

- and *unconstrained* SVM objective:

$$min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i^N h(y_i \mathbf{w}^T \mathbf{x}_i)$$

Multi-class hinge loss

- We want:

$$\mathbf{w}_{y_i}\mathbf{x}_i \geq \mathbf{w}_y\mathbf{x}_i + 1, \forall y \neq y_i$$

- Minimize:

$$1 + max_{y \neq y_i}\mathbf{w}_y\mathbf{x}_i - \mathbf{w}_{y_i}\mathbf{x}_i$$

- Hinge loss:

$$max(0, 1 + max_{y \neq y_i}\mathbf{w}_y\mathbf{x}_i - \mathbf{w}_{y_i}\mathbf{x}_i)$$

SVMs vs logistic regression

- When viewed from the point of view of regularized empirical loss minimization, SVM and logistic regression appear quite similar:

$$\text{SVM:} \qquad \sum_{i=1}^{n} \left(1 - y_i \left[w_0 + \mathbf{x}_i^T \mathbf{w}_1\right]\right)^+ + \quad \|\mathbf{w}_1\|^2/2$$

$$\text{Logistic:} \qquad \sum_{i=1}^{n} \overbrace{-\log \sigma\Big(y_i \left[w_0 + \mathbf{x}_i^T \mathbf{w}_1\right]\Big)}^{-\log P(y_i|\mathbf{x},\mathbf{w})} + \quad \|\mathbf{w}_1\|^2/2$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the logistic function.

(Note that we have transformed the problem maximizing the penalized log-likelihood into minimizing negative penalized log-likelihood.)

Adapted from Tommi Jaakola

SVMs vs logistic regression

- The difference comes from how we penalize "errors":

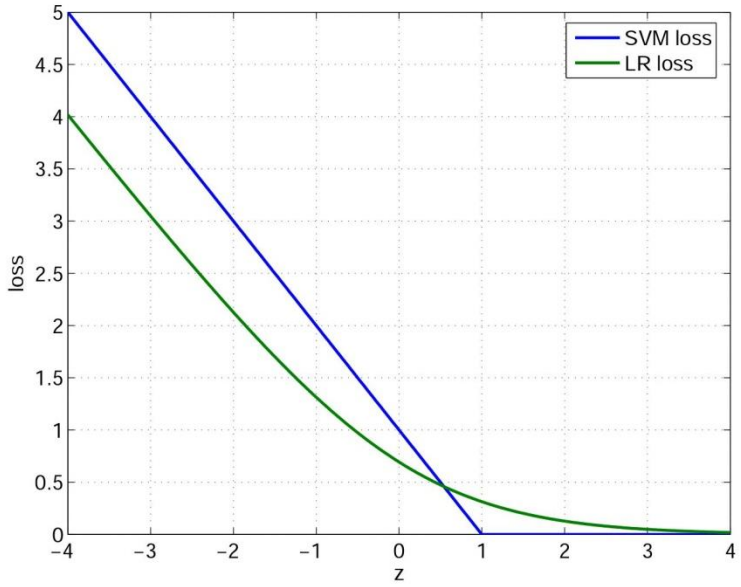
$$\text{Both:} \quad \sum_{i=1}^{n} \text{Loss}\Big(\overbrace{y_i\left[w_0 + \mathbf{x}_i^T \mathbf{w}_1\right]}^{z}\Big) + \quad \|\mathbf{w}_1\|^2/2$$

- SVM:

$$\text{Loss}(z) = (1-z)^+$$

- Regularized logistic reg:

$$\text{Loss}(z) = \log(1+\exp(-z))$$



Adapted from Tommi Jaakola

SVMs with latent variables

Three types of variables:

- $x \in \mathcal{X}$ always observed,
- $y \in \mathcal{Y}$ observed only in training,
- $z \in \mathcal{Z}$ never observed (latent).

Decision function: $f(x) = \text{argmax}_{y \in \mathcal{Y}} \ \max_{z \in \mathcal{Z}} \ \langle w, \phi(x, y, z) \rangle$

[C. Yu, T. Joachims, "Learning Structural SVMs with Latent Variables", ICML, 2009]
similar idea: [Felzenszwalb, McAllester, Ramaman. A Discriminatively Trained, Multiscale, Deformable Part Model, CVPR'08]

Adapted from S. Nowozin and C. Lampert

SVMs: Pros and cons

- Pros
  - Kernel-based framework is very powerful, flexible
  - Often a sparse set of support vectors – compact at test time
  - Work very well in practice, even with very small training sample sizes
  - Solution can be formulated as a quadratic program (next time)
  - Many publicly available SVM packages: e.g. LIBSVM, LIBLINEAR, SVMLight
- Cons
  - Can be tricky to select best kernel function for a problem
  - Computation, memory
    - At training time, must compute kernel values for all example pairs
    - Learning can take a very long time for large-scale problems

Adapted from Lana Lazebnik