

# CS 3750 Machine Learning

## Lecture 20

### Support vector machines

Milos Hauskrecht  
[milos@cs.pitt.edu](mailto:milos@cs.pitt.edu)  
5329 Sennott Square

### Linearly separable classes

There is a **hyperplane** that separates training instances with no error

**Hyperplane:**

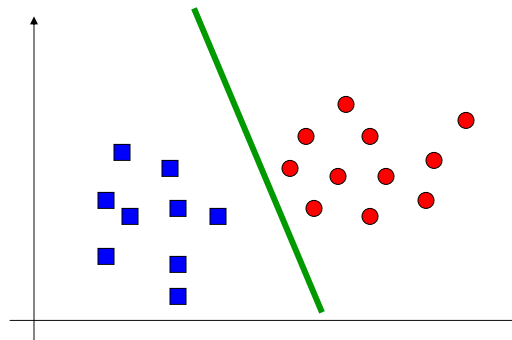
$$\mathbf{w}^T \mathbf{x} + w_0 = 0$$

**Class (+1)**

$$\mathbf{w}^T \mathbf{x} + w_0 > 0$$

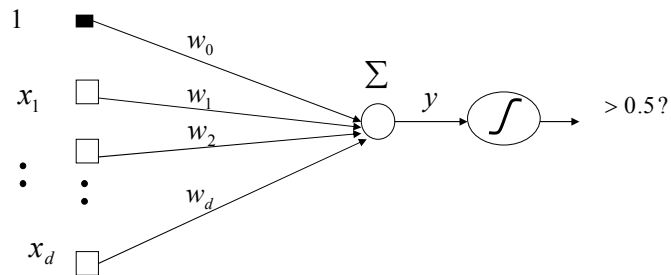
**Class (-1)**

$$\mathbf{w}^T \mathbf{x} + w_0 < 0$$



## Algorithms for linearly separable set

- **Separating hyperplane**  $\mathbf{w}^T \mathbf{x} + w_0 = 0$



- We can use **gradient methods** or Newton Rhapsion for sigmoidal switching functions and learn the weights
- Recall that we learn the linear decision boundary

CS 3750 Advanced Machine Learning

## Algorithms for linearly separable sets

- **Linear program solution:**
  - Find weights that satisfy the following constraints:

$$\mathbf{w}^T \mathbf{x}_i + w_0 \geq 0 \quad \text{For all } i, \text{ such that } y_i = +1$$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq 0 \quad \text{For all } i, \text{ such that } y_i = -1$$

**Property:** if there is a hyperplane separating the examples, the linear program finds the solution

### Other methods:

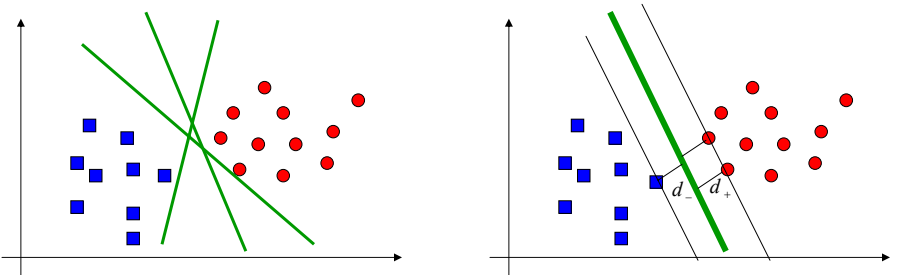
**Fisher linear discriminant**

**Perceptron algorithm**

CS 3750 Advanced Machine Learning

## Optimal separating hyperplane

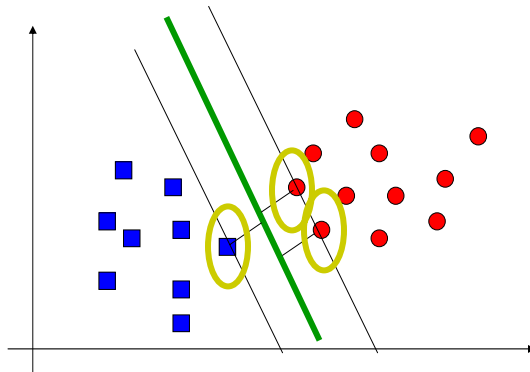
- There are multiple hyperplanes that separate the data points
  - Which one to choose?
- **Maximum margin** choice: the maximum distance of  $d_+ + d_-$ 
  - where  $d_+$  is the shortest distance of a positive example from the hyperplane (similarly  $d_-$  for negative examples)



CS 3750 Advanced Machine Learning

## Maximum margin hyperplane

- For the maximum margin hyperplane only examples on the margin matter (only these affect the distances)
- These are called **support vectors**



CS 3750 Advanced Machine Learning

## Finding maximum margin hyperplanes

- **Assume** that examples in the training set are  $(\mathbf{x}_i, y_i)$  such that  $y_i \in \{+1, -1\}$
- **Assume** that all data satisfy:

$$\mathbf{w}^T \mathbf{x}_i + w_0 \geq 1 \quad \text{for} \quad y_i = +1$$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq -1 \quad \text{for} \quad y_i = -1$$

- The inequalities can be combined as:

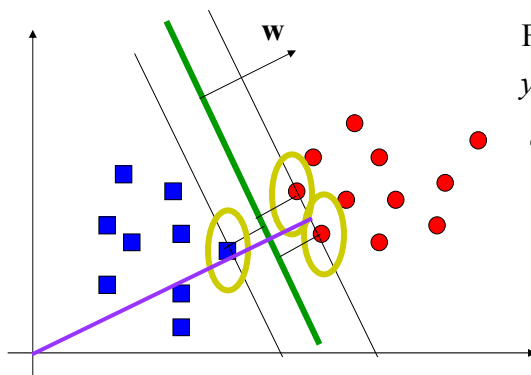
$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 \geq 0 \quad \text{for all } i$$

- Equalities define two hyperplanes:

$$\mathbf{w}^T \mathbf{x}_i + w_0 = 1 \quad \mathbf{w}^T \mathbf{x}_i + w_0 = -1$$

## Finding the maximum margin hyperplane

- **Geometrical margin:**  $\rho_{\mathbf{w}, w_0}(\mathbf{x}, y) = y(\mathbf{w}^T \mathbf{x} + w_0) / \|\mathbf{w}\|$ 
  - measures the distance of a point  $\mathbf{x}$  from the hyperplane
  - $\mathbf{w}$  - normal to the hyperplane  $\|\cdot\|$  - Euclidean norm



For points satisfying:

$$y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 = 0$$

The distance is  $\frac{1}{\|\mathbf{w}\|}$

**Width of the margin:**

$$d_+ + d_- = \frac{2}{\|\mathbf{w}\|}$$

## Maximum margin hyperplane

- We want to maximize  $d_+ + d_- = \frac{2}{\|\mathbf{w}\|}$

- We do it by **minimizing**

$$\|\mathbf{w}\|^2 / 2 = \mathbf{w}^T \mathbf{w} / 2$$

$\mathbf{w}, w_0$  - variables

- But we also need to enforce the constraints on points:

$$[y_i(\mathbf{w}^T \mathbf{x} + w_0) - 1] \geq 0$$

## Maximum margin hyperplane

- **Solution:** Incorporate constraints into the optimization
- **Optimization problem** (Lagrangian)

$$J(\mathbf{w}, w_0, \alpha) = \|\mathbf{w}\|^2 / 2 - \sum_{i=1}^n \alpha_i [y_i(\mathbf{w}^T \mathbf{x} + w_0) - 1]$$

$$\alpha_i \geq 0 \text{ - Lagrange multipliers}$$

- **Minimize** with regard to  $\mathbf{w}, w_0$  (primal variables)
- **Derivatives of L** with regard to  $\alpha$  should vanish at the solution

Lagrange multipliers enforce the satisfaction of constraints

$$\begin{aligned} \text{If } [y_i(\mathbf{w}^T \mathbf{x} + w_0) - 1] > 0 &\implies \alpha_i \rightarrow 0 \\ \text{Else } &\implies \alpha_i > 0 \quad \text{Active constraint} \end{aligned}$$

## Maximum margin hyperplane

- **Solution:** Incorporate constraints into the optimization
- **Optimization problem** (Lagrangian)

$$J(\mathbf{w}, w_0, \alpha) = \|\mathbf{w}\|^2 / 2 - \sum_{i=1}^n \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1]$$

$$\alpha_i \geq 0 \quad \text{- Lagrange multipliers}$$

- **Dual formulation (Wolfe dual):**
- **Maximize** with regard to  $\alpha$  (dual variables)
- **Derivatives of L** with regard to  $\mathbf{w}, w_0$  should vanish

---

CS 3750 Advanced Machine Learning

## Max margin hyperplane solution

- Set derivatives to 0 (Kuhn-Tucker conditions)

$$\nabla_{\mathbf{w}} J(\mathbf{w}, w_0, \alpha) = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = \bar{\mathbf{0}}$$

$$\frac{\partial J(\mathbf{w}, w_0, \alpha)}{\partial w_0} = - \sum_{i=1}^n \alpha_i y_i = 0$$

- Now we need to solve for Lagrange parameters only

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

Subject to constraints

$$\alpha_i \geq 0 \quad \text{for all } i, \quad \text{and} \quad \sum_{i=1}^n \alpha_i y_i = 0$$

- **Quadratic optimization problem:** solution  $\hat{\alpha}_i$  for all  $i$

---

CS 3750 Advanced Machine Learning

## Maximum hyperplane solution

- The resulting parameter vector  $\hat{\mathbf{w}}$  can be expressed as:

$$\hat{\mathbf{w}} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i \quad \hat{\alpha}_i \text{ is the solution of the dual problem}$$

- The parameter  $w_0$  is obtained through Karush-Kuhn-Tucker conditions  $\hat{\alpha}_i [y_i (\hat{\mathbf{w}}^T \mathbf{x}_i + w_0) - 1] = 0$

### Solution properties

- $\hat{\alpha}_i = 0$  for all points that are not on the margin
- $\hat{\mathbf{w}}$  is a **linear combination of support vectors only**
- The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 = 0$$

---

CS 3750 Advanced Machine Learning

## Support vector machines

- The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

- The decision:**

$$\hat{y} = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

---

CS 3750 Advanced Machine Learning

## Support vector machines

- **The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

- **The decision:**

$$\hat{y} = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

- **(!!):**

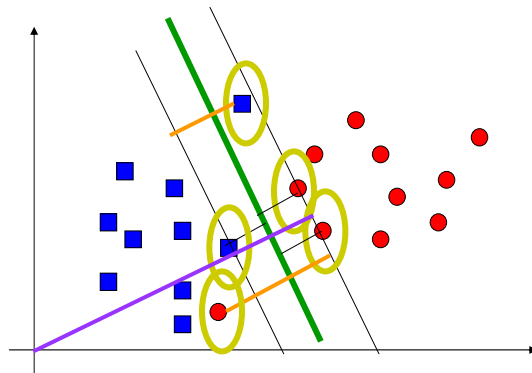
- Decision on a new  $\mathbf{x}$  requires to compute the inner product between the examples  $(\mathbf{x}_i^T \mathbf{x})$
- Similarly, the optimization depends on  $(\mathbf{x}_i^T \mathbf{x}_j)$

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

CS 3750 Advanced Machine Learning

## Extension to a linearly non-separable case

- **Idea:** Allow some flexibility on crossing the separating hyperplane



CS 3750 Advanced Machine Learning

## Extension to the linearly non-separable case

- Relax constraints with variables  $\xi_i \geq 0$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \geq 1 - \xi_i \quad \text{for} \quad y_i = +1$$

$$\mathbf{w}^T \mathbf{x}_i + w_0 \leq -1 + \xi_i \quad \text{for} \quad y_i = -1$$

- Error occurs if  $\xi_i \geq 1$ ,  $\sum_{i=1}^n \xi_i$  is the upper bound on the number of errors
- Introduce a penalty for the errors

$$\text{minimize} \quad \|\mathbf{w}\|^2 / 2 + C \sum_{i=1}^n \xi_i$$

Subject to constraints

$C$  – set by a user, larger  $C$  leads to a larger penalty for an error

## Extension to linearly non-separable case

- Lagrange multiplier form (primal problem)

$$J(\mathbf{w}, w_0, \alpha) = \|\mathbf{w}\|^2 / 2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + w_0) - 1 + \xi_i] - \sum_{i=1}^n \mu_i \xi_i$$

- Dual form after  $\mathbf{w}, w_0$  are expressed ( $\xi_i$  s cancel out)

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

Subject to:  $0 \leq \alpha_i \leq C$  for all  $i$ , and  $\sum_{i=1}^n \alpha_i y_i = 0$

**Solution:**  $\hat{\mathbf{w}} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i$

The difference from the separable case:  $0 \leq \alpha_i \leq C$

The parameter  $w_0$  is obtained through KKT conditions

## Support vector machines

- **The decision boundary:**

$$\hat{\mathbf{w}}^T \mathbf{x} + w_0 = \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0$$

- **The decision:**

$$\hat{y} = \text{sign} \left[ \sum_{i \in SV} \hat{\alpha}_i y_i (\mathbf{x}_i^T \mathbf{x}) + w_0 \right]$$

- **(!!):**
- Decision on a new  $\mathbf{x}$  requires to compute the inner product between the examples  $(\mathbf{x}_i^T \mathbf{x})$
- Similarly, the optimization depends on  $(\mathbf{x}_i^T \mathbf{x}_j)$

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

CS 3750 Advanced Machine Learning

## Nonlinear case

- The linear case requires to compute  $(\mathbf{x}_i^T \mathbf{x})$
- The non-linear case can be handled by using a set of features. Essentially we map input vectors to (larger) feature vectors

$$\mathbf{x} \rightarrow \boldsymbol{\phi}(\mathbf{x})$$

- It is possible to use SVM formalism on feature vectors

$$\boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}')$$

- **Kernel function**

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}')$$

- **Crucial idea:** If we choose the kernel function wisely we can compute linear separation in the feature space implicitly such that we keep working in the original input space !!!!

CS 3750 Advanced Machine Learning

## Kernel function example

- Assume  $\mathbf{x} = [x_1, x_2]^T$  and a feature mapping that maps the input into a quadratic feature set

$$\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x}) = [x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$$

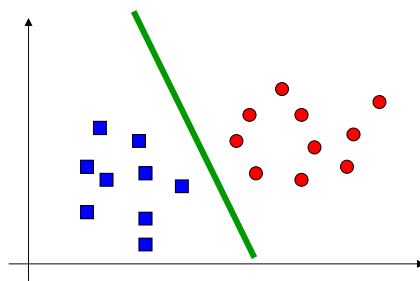
- Kernel function for the feature space:

$$\begin{aligned} K(\mathbf{x}', \mathbf{x}) &= \boldsymbol{\varphi}(\mathbf{x}')^T \boldsymbol{\varphi}(\mathbf{x}) \\ &= x_1^2 x_1'^2 + x_2^2 x_2'^2 + 2x_1 x_2 x_1' x_2' + 2x_1 x_1' + 2x_2 x_2' + 1 \\ &= (x_1 x_1' + x_2 x_2' + 1)^2 \\ &= (1 + (\mathbf{x}^T \mathbf{x}'))^2 \end{aligned}$$

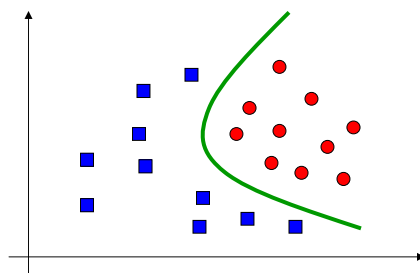
- The computation of the linear separation in the higher dimensional space is performed implicitly in the original input space

CS 3750 Advanced Machine Learning

## Kernel function example



Linear separator  
in the feature space



Non-linear separator  
in the input space

CS 3750 Advanced Machine Learning

## Kernel functions

- **Linear kernel**

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- **Polynomial kernel**

$$K(\mathbf{x}, \mathbf{x}') = [1 + \mathbf{x}^T \mathbf{x}']^k$$

- **Radial basis kernel**

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

## Kernels

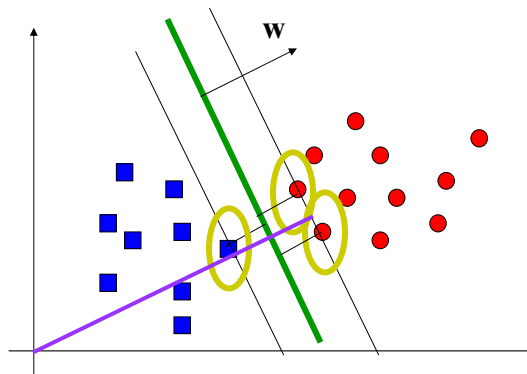
- The dot product  $\mathbf{x}^T \mathbf{x}$  is a **distance measure**
- **Kernels** can be seen as distance measures
  - Or conversely express degree of similarity
- Design criteria - we want kernels to be
  - **valid** – Satisfy Mercer condition of positive semi-definiteness
  - **good** – embody the “true similarity” between objects
  - **appropriate** – generalize well
  - **efficient** – the computation of  $k(\mathbf{x}, \mathbf{x}')$  is feasible

## Kernels

- SVM researchers have proposed kernels for comparison of variety of objects:
  - Strings
  - Trees
  - Graphs
- **Cool thing:**
  - SVM algorithm can be now applied to classify a variety of objects

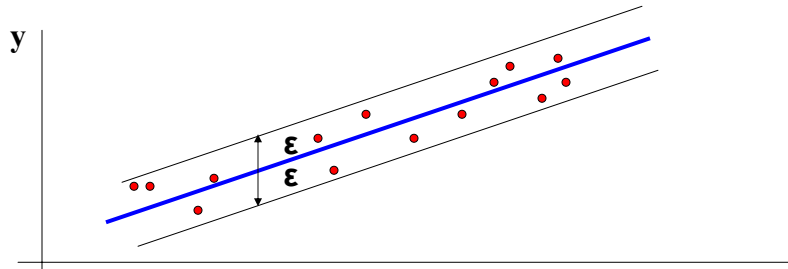
## Support vector machine SVM

- SVM maximize the margin around the separating hyperplane.
- The decision function is fully specified by a subset of the training data, the support vectors.



## Support vector machine for regression

- **Regression** = find a function that fits the data.
- A data point may be wrong due to the noise
- **Idea:** Error from points which are close **should count as a valid noise**
- Line should be influenced by the real data not the noise.

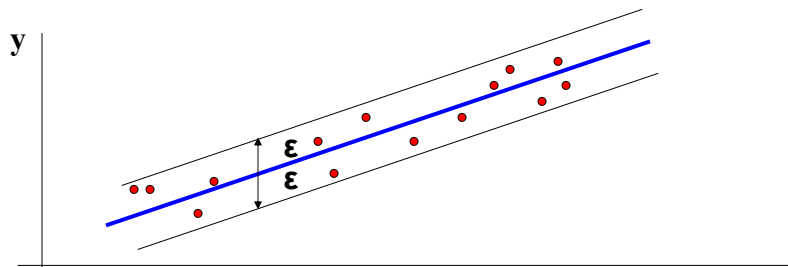


CS 3750 Advanced Machine Learning

## Linear model

- **Training data:**  
 $\{(x_1, y_1), \dots, (x_l, y_l)\}$ ,  $x \in R^n$ ,  $y \in R$
- Our goal is to find a function  $f(x)$  that has at most  $\epsilon$  deviation from the actually obtained target for all the training data.

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$



CS 3750 Advanced Machine Learning

## Linear model

**Linear function:**

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

We want a function that is:

- **flat:** means that one seeks small  $\mathbf{w}$
- all data points are within its  $\varepsilon$  neighborhood

The problem can be formulated as a **convex optimization problem:**

$$\begin{aligned} & \text{minimize} && \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to} && \begin{cases} y_i - \langle \mathbf{w}_i, \mathbf{x}_i \rangle - b \leq \varepsilon \\ \langle \mathbf{w}_i, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon \end{cases} \end{aligned}$$

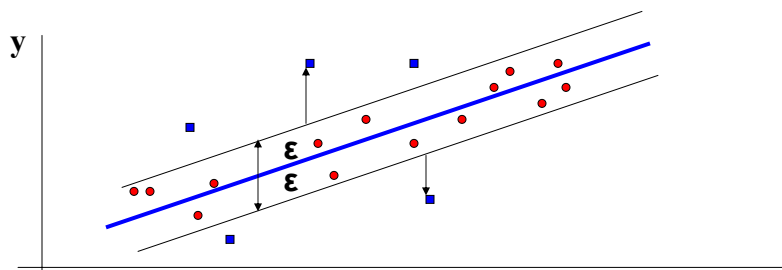
All data points are assumed to be in the  $\varepsilon$  neighborhood

## Linear model

- **Real data:** not all data points always fall into the  $\varepsilon$  neighborhood

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

- **Idea:** penalize points that fall outside the  $\varepsilon$  neighborhood



## Linear model

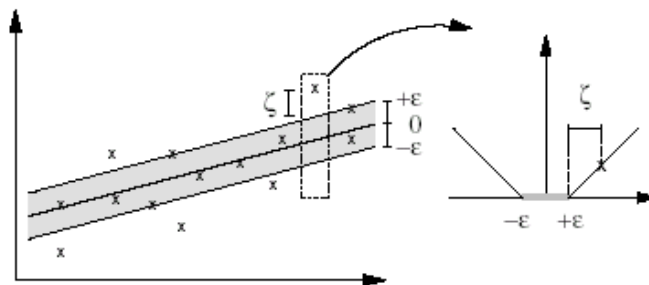
**Linear function:**

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

**Idea:** penalize points that fall outside the  $\varepsilon$  neighborhood

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l (\xi_i + \xi_i^*) \\ \text{subject to} \quad & \begin{cases} y_i - \langle \mathbf{w}_i, \mathbf{x}_i \rangle - b \leq \varepsilon + \xi_i \\ \langle \mathbf{w}_i, \mathbf{x}_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{aligned}$$

## Linear model



$$|\xi|_{\varepsilon} = \begin{cases} 0 & \text{for } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases}$$

**$\varepsilon$ -insensitive loss function**

## Optimization

**Lagrangian that solves the optimization problem**

$$\begin{aligned} L = & \frac{1}{2} \langle w, w \rangle + C \sum_{i=1}^l (\xi_i + \xi_i^*) \\ & - \sum_{i=1}^l a_i (\varepsilon - \xi_i - y_i + \langle w, x_i \rangle + b) - \sum_{i=1}^l a_i^* (\varepsilon + \xi_i^* + y_i - \langle w, x_i \rangle - b) \\ & - \sum_{i=1}^l (\eta_i \xi_i + \eta_i^* \xi_i^*) \end{aligned}$$

**Subject to**  $a_i, a_i^*, \eta_i, \eta_i^* \geq 0$

**Primal variables**  $w, b, \xi_i, \xi_i^*$

---

CS 3750 Advanced Machine Learning

## Optimization

**Derivatives with respect to primal variables**

$$\frac{\partial L}{\partial b} = \sum_{i=1}^l (a_i^* - a_i) = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l (a_i^* - a_i) \mathbf{x}_i = \mathbf{0}$$

$$\frac{\partial L}{\partial \xi_i^{(*)}} = C - a_i^{(*)} - \eta_i^{(*)} = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - a_i - \eta_i = 0$$

---

CS 3750 Advanced Machine Learning

## Optimization

$$\begin{aligned}
 L &= \frac{1}{2} \langle w, w \rangle + \sum_{i=1}^l C \xi_i + \sum_{i=1}^l C \xi_i^* \\
 &- \sum_{i=1}^l a_i \varepsilon - \sum_{i=1}^l a_i \xi_i - \sum_{i=1}^l a_i y_i - \sum_{i=1}^l a_i \langle \omega, x_i \rangle + \sum_{i=1}^l a_i b \\
 &- \sum_{i=1}^l a_i^* \varepsilon - \sum_{i=1}^l a_i^* \xi_i^* - \sum_{i=1}^l a_i^* y_i + \sum_{i=1}^l a_i^* \langle \omega, x_i \rangle + \sum_{i=1}^l a_i^* b \\
 &- \sum_{i=1}^l \eta_i \xi_i - \sum_{i=1}^l \eta_i^* \xi_i^*
 \end{aligned}$$

---

CS 3750 Advanced Machine Learning

## Optimization

$$\begin{aligned}
 L &= \frac{1}{2} \langle w, w \rangle + \sum_{i=1}^l \xi_i \underbrace{(C - \eta_i - a_i)}_{=0(C-\eta_i^{(*)}-a_i^{(*)}=0)} + \\
 &\sum_{i=1}^l \xi_i^* \underbrace{(C - \eta_i^* - a_i^*)}_{=0(C-\eta_i^{(*)}-a_i^{(*)}=0)} - \sum_{i=1}^l (a_i + a_i^*) \varepsilon - \sum_{i=1}^l (a_i + a_i^*) y_i \\
 &- \sum_{i=1}^l \underbrace{(a_i - a_i^*) \langle \omega, x_i \rangle}_{=\langle w, w \rangle (\omega = \sum_{i=1}^l (a_i + a_i^*) x_i)} + \sum_{i=1}^l \underbrace{(a_i^* - a_i) b}_{=0(\sum_{i=1}^l (a_i^* - a_i) = 0)}
 \end{aligned}$$

---

CS 3750 Advanced Machine Learning

## Optimization

$$L = -\frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^l (a_i + a_i^*)\varepsilon - \sum_{i=1}^l (a_i + a_i^*)y_i$$

Maximize the dual

$$L(a, a^*) = -\frac{1}{2} \sum_{i=1}^l (a_i - a_i^*)(a_j - a_j^*)\langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ - \sum_{i=1}^l (a_i + a_i^*)\varepsilon - \sum_{i=1}^l (a_i + a_i^*)y_i$$

$$\text{subject to } \begin{cases} \sum_{i=1}^l (a_i - a_i^*) = 0 \\ a_i, a_i^* \in [0, C] \end{cases}$$

---

CS 3750 Advanced Machine Learning

## Solution

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^l (a_i^* - a_i)\mathbf{x}_i = \mathbf{0}$$

$$\mathbf{w} = \sum_{i=1}^l (a_i - a_i^*)\mathbf{x}_i$$

We can get:

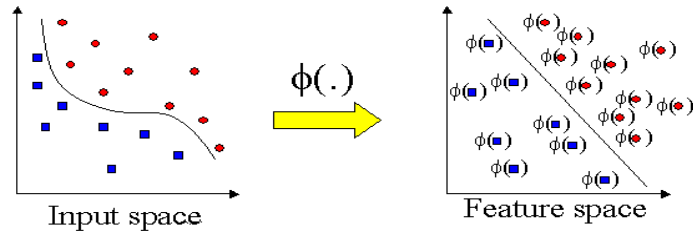
$$f(\mathbf{x}) = \sum_{i=1}^l (a_i - a_i^*)\langle \mathbf{x}_i, \mathbf{x} \rangle + b$$

at the optimal solution the Lagrange multipliers are non-zero only **for points outside the  $\varepsilon$  band.**

---

CS 3750 Advanced Machine Learning

## Nonlinear extension



### Kernel trick

- Replace the inner product with a kernel
- A well chosen kernel leads to efficient computation