Machine Learning

The Non-Linear SVM

Dariush Hosseini

dariush.hosseini@ucl.ac.uk
Department of Computer Science
University College London

Lecture Overview

Lecture Overview

By the end of this lecture you should:

1. Understand how we can extend the **Linear SVM** using **Kernels** to create the **Non-Linear SVM**
2. Know how we can extend the **binary** support vector classifier to enable us to tackle **multi-class** learning problems
3. Know that SVMs can be used for **regression** as well as classification

Lecture Overview

Linear SVM: Optimisation Problem

- Recall the original linear SVM problem:

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

- And the associated **dual problem**:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n}\alpha^{(i)} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\mathbf{x}^{(i)}\cdot\mathbf{x}^{(j)} \tag{2}$$

$$\text{subject to:} \qquad \sum_{i=1}^{n}\alpha^{(i)}y^{(i)} = 0$$

$$0 \leqslant \alpha^{(i)} \leqslant C$$

Linear SVM: Optimisation Solution & Prediction

- Recall that we could express our solution to the problem as:

$$\mathbf{w}^* = \sum_{i \in \mathcal{SV}} \alpha^{(i)*} y^{(i)} \mathbf{x}^{(i)}$$

$$b^* = \frac{1}{|\widetilde{\mathcal{SV}}|} \sum_{i \in \widetilde{\mathcal{SV}}} \left(y^{(i)} - \sum_{j \in \widetilde{\mathcal{SV}}} \alpha^{(j)*} y^{(j)} \mathbf{x}^{(j)} \cdot \mathbf{x}^{(i)} \right)$$

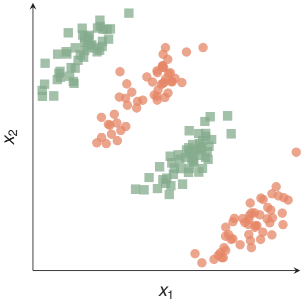
  Where $\mathcal{SV}$ is the set of support vectors, and $\widetilde{\mathcal{SV}}$ is the set of support vectors for which $0 < \alpha^{(i)} < C$

- And predict the class of a novel test point, **z**, as:

$$\begin{aligned} f(\mathbf{z}) &= \operatorname{sgn}(\mathbf{w}^* \cdot \mathbf{z} + b^*) \\ &= \operatorname{sgn}\left(\sum_{i \in \mathcal{SV}} \alpha^{(i)*} y^{(i)} \mathbf{x}^{(i)} \cdot \mathbf{z} + b^* \right) \end{aligned}$$

The Failure Case

- Recall that this works well for both **separable** and **noisy** settings...if the boundary is **linear**
- But not well for **non-linear boundaries**:



- Can we use **kernel methods** to enhance the algorithm?

Kernel Methods: Recap

- **Kernel Trick**:
  - If the dependency of input attributes within our algorithm can be expressed solely in terms of **inner products** between the input vectors...
  - ...Then we can replace all such inner products with an appropriate **kernel function** output
- **Mercer's Theorem**:
  - Gives us criteria for the **validity** of kernels which we may use in the kernel trick such that they will implicitly express a valid feature mapping
- **Representer Theorem**:
  - Gives necessary and sufficient conditions that the form of an optimisation problem must take such that it will admit the kernel trick

Representer Theorem: Recap

- For a regularised loss function, L, defined such that:

$$\mathsf{L}(\mathbf{w}) = \sum_{i=1}^{n} \widetilde{\mathsf{L}}(y^{(i)}, \mathbf{w} \cdot \phi(\mathbf{x}^{(i)})) + \Omega(\mathbf{w})$$

- If and only if $\Omega(\mathbf{w})$ is a non-decreasing function of $\|\mathbf{w}\|_2$ then, if $\mathbf{w}^*$ minimises L, it admits the following representation:

$$\mathbf{w}^* = \sum_{i=1}^{n} \widetilde{\alpha}_i \phi(\mathbf{x}^{(i)}) \qquad \text{where: } \widetilde{\alpha}_i \in \mathbb{R}$$

- And therefore for a novel test point, **z**:

$$f(\mathbf{z}) = \mathbf{w} \cdot \phi(\mathbf{z}) = \sum_{i=1}^{n} \widetilde{\alpha}_i \kappa(\mathbf{x}^{(i)}, \mathbf{z})$$

Representer Theorem: SVM

- We note that the **SVM learning problem** given by expression (1), satisfies the requirement of the **Representer Theorem**

- Thus we can state that:

$$\mathbf{w}^* = \sum_{i=1}^{n} \widetilde{\alpha}_i \mathbf{x}^{(i)}$$

- Substituting this back into the original learning problem will lead to a dual form, from which we see that:

$$\widetilde{\alpha}_i = \alpha^{(i)} y^{(i)}$$

- And so we are led back to the **dual solution** of expression (2) by a different route

Representer Theorem: SVM

- So we can apply the kernel trick to the SVM:

$$\phi(\mathbf{x}) \longleftarrow \mathbf{x}$$
$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \longleftarrow \mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)}$$

  Here:

$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)}) \cdot \phi(\mathbf{x}^{(j)})$$

Non-Linear SVM: Optimisation Problem

- The Non-Linear SVM problem becomes:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \max(0, y^{(i)}(\mathbf{w} \cdot \phi(\mathbf{x}^{(i)}) + b)) \tag{3}$$

- With the associated **dual problem**:

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha^{(i)} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha^{(i)}\alpha^{(j)}y^{(i)}y^{(j)}\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \tag{4}$$

$$\text{subject to:} \qquad \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} = 0$$

$$0 \leqslant \alpha^{(i)} \leqslant C$$

Non-Linear SVM: Optimisation Solution & Prediction

- And the solution to the non-linear problem is:

$$\mathbf{w}^* = \sum_{i \in \mathcal{SV}} \alpha^{(i)*} y^{(i)} \phi(\mathbf{x}^{(i)})$$

$$b^* = \frac{1}{|\widetilde{\mathcal{SV}}|} \sum_{i \in \widetilde{\mathcal{SV}}} \left(y^{(i)} - \sum_{j \in \widetilde{\mathcal{SV}}} \alpha^{(j)*} y^{(j)} \kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \right)$$

- And we can predict the class of a novel test point, **z**, as:

$$\begin{aligned} f(\mathbf{z}) &= \text{sgn}(\mathbf{w}^* \cdot \phi(\mathbf{z}) + b^*) \\ &= \text{sgn}\left(\sum_{i \in \mathcal{SV}} \alpha^{(i)*} y^{(i)} \kappa(\mathbf{x}^{(i)}, \mathbf{z}) + b^* \right) \end{aligned}$$

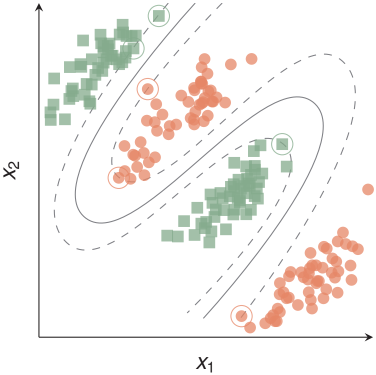
Failure Case: RBF Solution

- Let's return to the failure case, but now we'll attempt to learn a **boundary** in the **feature space** defined by the **RBF kernel**:

$$\kappa(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\gamma\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2\right)$$

- Recall that the RBF kernel is associated with an ∞-order polynomial feature map, so it has the capacity to learn **complex boundaries**

Failure Case: RBF Solution

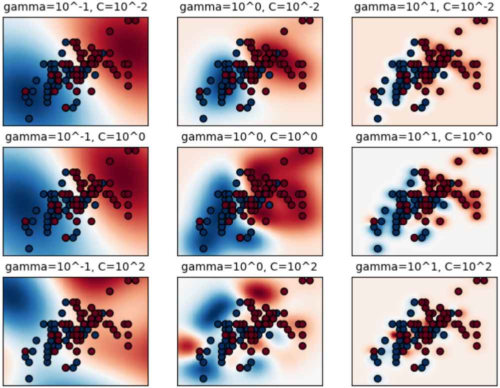


- Soft-margin classifier with RBF Kernel ($\gamma = 0.02$)

Hyperparameters

- How do we set the **hyperparameters** C and γ?
- **Cross-validation** is usually employed
- Increasing C leads to:
  - Less tolerance of errors
  - More complex boundaries
- Increasing γ leads to:
  - Sharply peaked similarity measure
  - So each support vector becomes only locally influential
  - And we obtain more complex boundaries

Hyperparameters



Points to Remember

- **Linearity**
  - SVMs are a linear, maximal margin technique
- **Kernel Trick**
  - The use of kernels makes SVMs extremely flexible
- **Sparsity**
  - Once trained we need only retain the support vectors
- **Convexity**
  - The hinge loss and margin maximisation make the optimisation convex

Lecture Overview

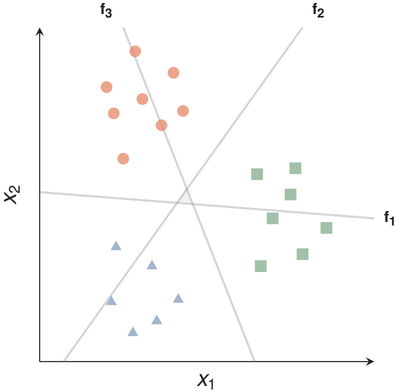
SVMs with Multiple Classes?

- The original SVM is designed for **binary** classification
- We can extend the idea of a separating hyperplane to the case where we are attempting to classify **multiple classes**
- Two of the most popular approaches are:
  - **One-Versus-One** (OvO)
  - **One-Versus-All** (OvA)

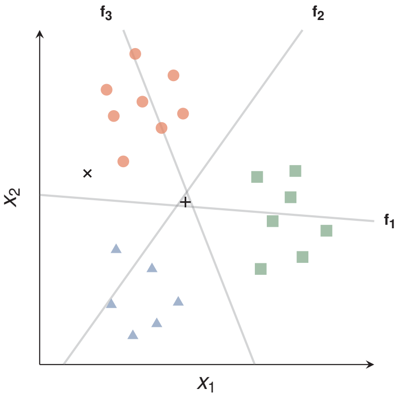
One-versus-One

- The OvO approach for $K > 2$ classes constructs $\frac{K(K-1)}{2}$ SVMs, each of which compares a pair of classes
- A new instance is then classified using each classifier
- A tally is kept and the final clasification is obtained by **majority vote**

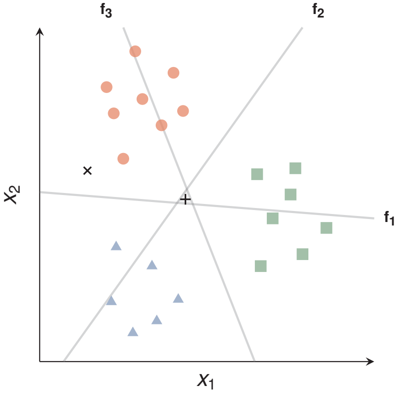
One-versus-One



One-versus-One

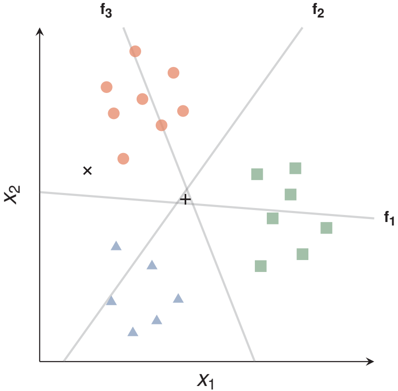


One-versus-One



×	f_1	f_2	f_3
●	✓	✓	
■			
▲			✓

One-versus-One



×	f_1	f_2	f_3
●	✓	✓	
■			
▲			✓

+	f_1	f_2	f_3
●	✓		
■		✓	
▲			✓

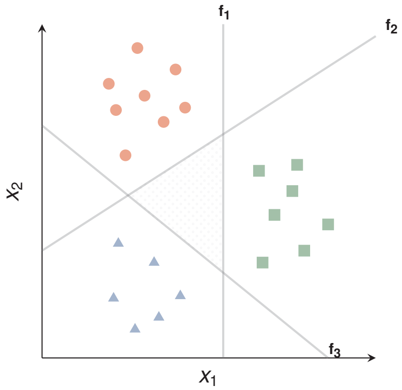
One-versus-One

- **Pros**:
  - Easy to train classifiers (provided they are equally weighted)
- **Cons**:
  - Many classifiers to train
  - Regions of ambiguity

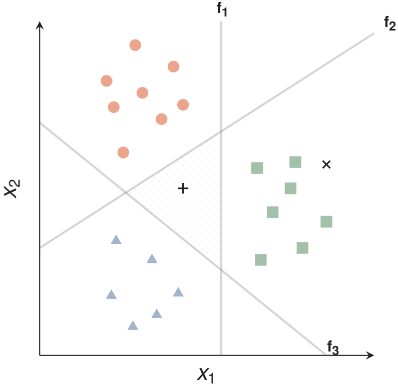
One-versus-All

- The OvA approach constructs K SVMs, each of which compares the instances of one class to instances of *all* other classes
- A new instance is then classified by calculating the discriminant function for each of the K classifiers
- Final classification is obtained by assigning according to a positive classification, or, in the event of ties, according to the **maximal discriminant**

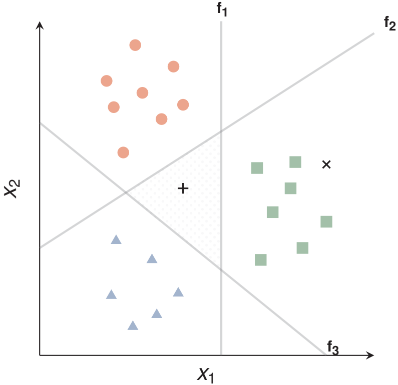
One-versus-All



One-versus-All

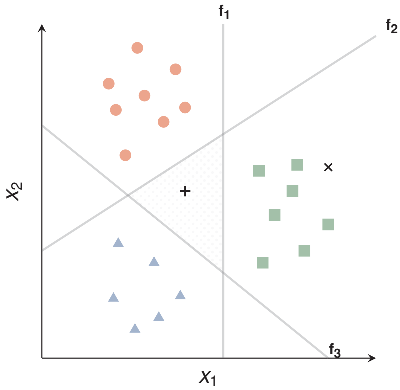


One-versus-All



×	f_1	f_2	f_3
●		-1	
▲			-1
■	$+1$		

One-versus-All



×	f_1	f_2	f_3
●		-1	
▲			-1
■	$+1$		

+	f_1	f_2	f_3
●		-1	
▲			-1
■	-1		

One-versus-All

- **Pros**:
  - Few classifiers to train
- **Cons**:
  - Class imbalance
  - Regions of ambiguity
  - Scaling of discriminant function needs to be tuned (distance to different hyperplanes is not measured on the same scale!)

A Consistent Approach

- The regions of ambiguity occur because OvO and OvA are just **heuristics**
- Each binary classifier does not know that we use its output prediction for a multi-class prediction - this might lead to sub-optimal results
- It is better to specify the complete task initially and seek to tackle the problem whole
- How can we do this?

K Class Discriminant

- We can, for example, specify a K class discriminant, which consists of K linear functions of the form:

$$f_i(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{x} + w_{i0}$$

  Here i is a class index running from 1 to K

- Then we assign a point to a class associated with k if $f_k(\mathbf{x}) > f_j(\mathbf{x})$ for all $j \neq k$

- This results in **decision boundaries** between k and j given by:

$$f_k(\mathbf{x}) = f_j(\mathbf{x})$$
$$\implies (\mathbf{w}_k - \mathbf{w}_j) \cdot \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

- Will this still give rise to regions of ambiguity?

Non-Ambiguity of the K Class Discriminant

- No, because each of the decision regions defined by the discriminant functions is **convex**:

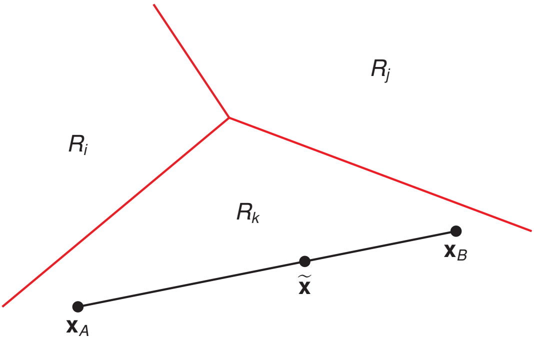
- Consider two points $\mathbf{x}_A$ and $\mathbf{x}_B$ which both lie in the decision region associated with k, R_k

- Then any point, $\widetilde{\mathbf{x}}$, that lies on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ can be expressed as:

$$\widetilde{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda)\mathbf{x}_B$$

  Here $0 \leqslant \lambda \leqslant 1$

Non-Ambiguity of the K Class Discriminant



UCL

Non-Ambiguity of the K Class Discriminant

- Using our decision functions we can write:

$$
\begin{aligned}
f_k(\widetilde{\mathbf{x}}) &= \mathbf{w}_k \cdot \widetilde{\mathbf{x}} + w_{k0} \\
&= \mathbf{w}_k \cdot (\lambda \mathbf{x}_A + (1-\lambda)\mathbf{x}_B) + (\lambda w_{k0} + (1-\lambda) w_{k0}) \\
&= \lambda f_k(\mathbf{x}_A) + (1-\lambda) f_k(\mathbf{x}_B)
\end{aligned}
$$

- Because $\mathbf{x}_A$ and $\mathbf{x}_B$ lie in R_k, then $f_k(\mathbf{x}_A) > f_j(\mathbf{x}_A)$ and $f_k(\mathbf{x}_B) > f_j(\mathbf{x}_B)$ for all $j \neq k$

- Therefore $f_k(\widetilde{\mathbf{x}}) > f_j(\widetilde{\mathbf{x}})$ for all $j \neq k$, and so $\widetilde{\mathbf{x}}$ also lies in R_k, and R_k is convex

- And this rules out any ambiguous regions, which would necessarily be non-convex

Aside: Softmax Regression

- How do we motivate such a K class discriminant?
- **Softmax Regression** provides one way
- This is a **probabilistic approach**, which is the multinomial generalisation of **logistic regression**
- We begin by noting that for **multinomial classification** and **misclassification loss** we can, w.l.o.g, prove the following **Bayes Optimal classifier**, f^*:

$$f^*(\mathbf{x}) = k \qquad \text{if:} \qquad p_{\mathcal{Y}}(y = k|\mathbf{x}) > p_{\mathcal{Y}}(y = j|\mathbf{x}) \qquad \forall j \neq k$$

Aside: Softmax Regression

- Now, we make a set of modelling assumptions: We take one of the classes, say K, as the reference class, and then make the following assumption:

Aside: Softmax Regression

- Now, we make a set of modelling assumptions: We take one of the classes, say K, as the reference class, and then make the following assumption:

$$\log\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) = (\mathbf{w}_i - \mathbf{w}_K)\cdot\mathbf{x} + (w_{i0} - w_{K0})$$

Aside: Softmax Regression

- Now, we make a set of modelling assumptions: We take one of the classes, say K, as the reference class, and then make the following assumption:

$$\log\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) = (\mathbf{w}_i - \mathbf{w}_K)\cdot\mathbf{x} + (w_{i0} - w_{K0})$$

$$\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) = \exp((\mathbf{w}_i - \mathbf{w}_K)\cdot\mathbf{x} + (w_{i0} - w_{K0})) \quad (5)$$

Aside: Softmax Regression

- Now, we make a set of modelling assumptions: We take one of the classes, say K, as the reference class, and then make the following assumption:

$$\begin{aligned}
\log\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) &= (\mathbf{w}_i-\mathbf{w}_K)\cdot\mathbf{x}+(w_{i0}-w_{K0}) \\
\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) &= \exp((\mathbf{w}_i-\mathbf{w}_K)\cdot\mathbf{x}+(w_{i0}-w_{K0})) \qquad (5)\\
\sum_{i=1}^{K-1}\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) &= \frac{1-p_{\mathrm{y}}(y=K|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}
\end{aligned}$$

Aside: Softmax Regression

- Now, we make a set of modelling assumptions: We take one of the classes, say K, as the reference class, and then make the following assumption:

$$\begin{aligned}
\log\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) &= (\mathbf{w}_i - \mathbf{w}_K)\cdot\mathbf{x} + (w_{i0} - w_{K0}) \\
\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) &= \exp((\mathbf{w}_i - \mathbf{w}_K)\cdot\mathbf{x} + (w_{i0} - w_{K0})) \qquad (5) \\
\sum_{i=1}^{K-1}\left(\frac{p_{\mathrm{y}}(y=i|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})}\right) &= \frac{1 - p_{\mathrm{y}}(y=K|\mathbf{x})}{p_{\mathrm{y}}(y=K|\mathbf{x})} \\
&= \sum_{i=1}^{K-1}\exp\left((\mathbf{w}_i - \mathbf{w}_K)\cdot\mathbf{x} + (w_{i0} - w_{K0})\right)
\end{aligned}$$

Aside: Softmax Regression

- So:

$$p_{\mathcal{Y}}(y = K|\mathbf{x}) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp\left((\mathbf{w}_i - \mathbf{w}_K) \cdot \mathbf{x} + (w_{i0} - w_{K0})\right)}$$

- Substitute in expression (5):

$$p_{\mathcal{Y}}(y = i|\mathbf{x}) = \frac{\exp\left((\mathbf{w}_i - \mathbf{w}_K) \cdot \mathbf{x} + (w_{i0} - w_{K0})\right)}{1 + \sum_{j=1}^{K-1} \exp\left((\mathbf{w}_j - \mathbf{w}_K) \cdot \mathbf{x} + (w_{j0} - w_{K0})\right)}$$

- More compactly, for all i:

$$p_{\mathcal{Y}}(y = i|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_i \cdot \mathbf{x} + w_{i0}\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{w}_j \cdot \mathbf{x} + w_{j0}\right)}$$

Aside: Softmax Regression

- This is the **softmax function**
- We can use it to state that if:

$$p_{\mathcal{Y}}(y = k|\mathbf{x}) > p_{\mathcal{Y}}(y = j|\mathbf{x})$$

- Then:

$$\frac{\exp\left(\mathbf{w}_k \cdot \mathbf{x} + w_{k0}\right)}{\sum_{i=1}^{K} \exp\left(\mathbf{w}_i \cdot \mathbf{x} + w_{i0}\right)} > \frac{\exp\left(\mathbf{w}_j \cdot \mathbf{x} + w_{j0}\right)}{\sum_{i=1}^{K} \exp\left(\mathbf{w}_i \cdot \mathbf{x} + w_{i0}\right)}$$
$$\mathbf{w}_k \cdot \mathbf{x} + w_{k0} > \mathbf{w}_j \cdot \mathbf{x} + w_{j0}$$
$$f_k(\mathbf{x}) > f_j(\mathbf{x})$$

Aside: Softmax Regression

- So, the condition for the Bayes Optimal Classifier allows us to write:

$$f^*(\mathbf{x}) = k \qquad \text{if:} \qquad f_k(\mathbf{x}) > f_j(\mathbf{x}) \qquad \forall j \neq k$$

  Where $f_i(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{x} + w_{i0}$

- And this is the K Class Linear Discriminant definition

Aside: The Multiclass SVM

- Alternatively, Weston & Watkins ('99) proposed the **multiclass SVM**
- This is a formulation of the SVM that enables a multiple classification problem to be solved in a single optimisation
- This approach also gives rise to a K Class Linear Discriminant

Lecture Overview

Support Vector Regression

- The **SVR** follows similar principles to the SVC
- It is a learning algorithm that emerges from the optimisation of a **generalisation bound** on the **ϵ-insensitive loss**
- This is a **convex**, **sparsity inducing**, regression loss function

Support Vector Regression

- The optimisation problem for the linear SVR is:

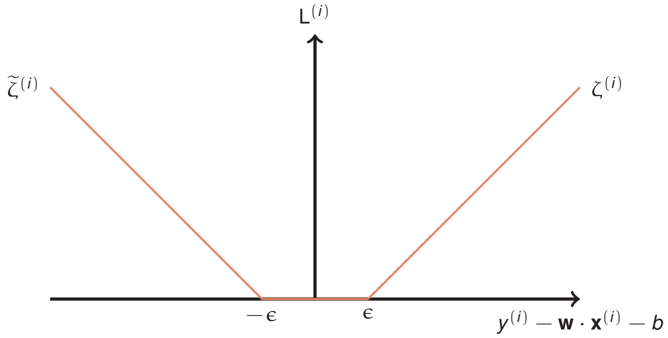
$$\min_{\mathbf{w}, b, \zeta, \widetilde{\zeta}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\zeta^{(i)} + \widetilde{\zeta}^{(i)})$$

$$\text{subject to:} \qquad \begin{aligned} & y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)} - b \leqslant \epsilon + \zeta^{(i)} \\ & \mathbf{w}\cdot\mathbf{x}^{(i)} + b - y^{(i)} \leqslant \epsilon + \widetilde{\zeta}^{(i)} \\ & \zeta^{(i)}, \widetilde{\zeta}^{(i)} \geqslant 0 \end{aligned}$$

  Here ϵ, C are **hyperparameters**
  C is the trade-off parameter which controls the width of **margin** versus the **tolerance for error**

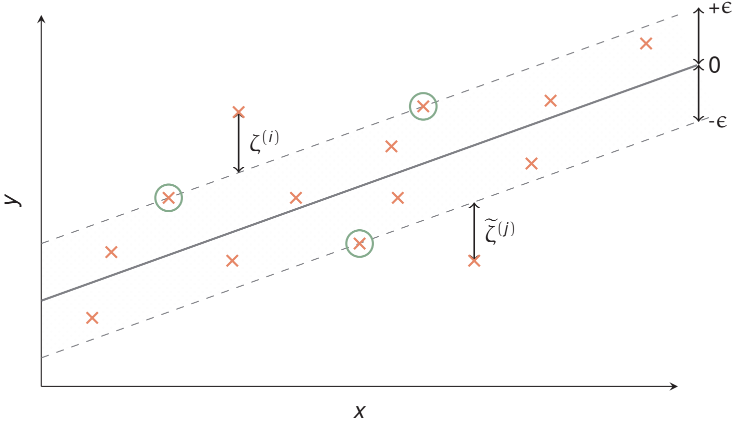
- What is the nature of the errors?

Support Vector Regression



- $L^{(i)} = \max(|y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} - b| + \epsilon, 0)$
- $\zeta^{(i)}, \tilde{\zeta}^{(i)}$ are **slack variables**

Support Vector Regression



- Here, for $\epsilon = 0$, as $C \to \infty$, we tend to **Laplace regression**

Lecture Overview

Summary

1. Adding **kernels** to the **linear SVM** allows us to build highly effective **non-linear SVM** classifiers

2. It is possible to extend the **binary** approach to classification to **multi-class problems** via heuristics such as **OvO** and **OvA**...with some problems

3. The SVM framework can be extended to tackle **regression** problems via the **SVR**. This algorithm shares the **sparsity** inducing properties of the original SVM.