Artificial Intelligence
CS 534

Week 7

Research paper: timeline and content

Title	Should have it by now
Abstract: 1-2 paragraphs	The last thing: Dec 3
Introduction: 1-2 paragraphs	Nov 14
Background: 2-4 paragraphs	Nov 17
Methods: 1.5-2 pages	Nov 24
Results: 1.5-2 pages	Dec 1
Conclusion: 2-3 paragraphs	The last thing: Dec 2
Bibliography	The last thing: Dec 2

- Total size: no less than 5 full pages, including figures/tables and references

Research paper. Introduction & Background

- What is the general area of research addressed by the paper? Why is it important?
- What is the specific problem addressed by the paper? Describe it.
- What are the main challenges that one will face when addressing this problem?
- What are the immediate benefits, once the problem is solved?

Research paper. Background

- Describe state-of-art methods that address your problem.
- What are their advantages and disadvantages?
- Describe the computational approach that you will use. How is it going address the disadvantages of the previous methods?
- Paper organization

Research paper. Methods. Results

Methods

- Detailed description of the method used
- Description of the datasets
- Description of the assessment protocol
- Be rigorous and specific

Results

- Simple statements of the obtained results
- Results of comparative assessment

Research paper. Conclusion

- Short summary of the obtained results
- Advantages and disadvantages of your method
- Method's accuracy and coverage, based on the obtained results
- Interpretation of the obtained results
- Future research

Figures and Tables

- An average paper should have 2-3 figures and 1-2 tables
- Make sure that the Figure legends are clearly visible
- Table caption: top of the table, figure caption: bottom of the figure.
- Caption example:

Table 1: Migration of chickens to the South Pole. The percentage of chicken population (P_{ch}) that is known to migrate to the South Pole is estimated by the Chicken-Smith law and compared to the experimentally obtained data (V_{ch})

Figures and Tables

- An average paper should have 2-3 figures and 1-2 tables
- Make sure that the Figure legends are clearly visible
- Table caption: top of the table, figure caption: bottom of the figure.
- Caption example:

Table 1: Migration of chickens to the South Pole. The percentage of chicken population (P_{ch}) that is known to migrate to the South Pole is estimated by the Chicken-Smith law and compared to the experimentally obtained data (V_{ch})

Figures and Tables

- An average paper should have 2-3 figures and 1-2 tables
- Make sure that the Figure legends are clearly visible
- Table caption: top of the table, figure caption: bottom of the figure.
- Caption example:

Table 1: Migration of chickens to the South Pole. The percentage of chicken population (P_{ch}) that is known to migrate to the South Pole is estimated by the Chicken-Smith law and compared to the experimentally obtained data (V_{ch})

Paper style

- Word document (or PDF, if using LaTeX)

- IEEE Manuscript style:

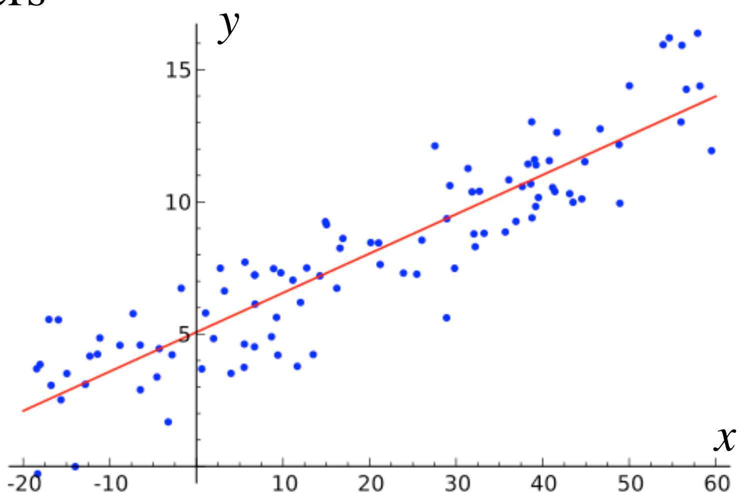
https://www.ieee.org/conferences_events/conferences/publishing/templates.html

- References:
  - Endnote (recommended)
  - Reference manager
  - BibTex

Regression basics

- Linear regression: data can be modeled as a linear function
  - $h_w(x) = w_0 + w_1 x$, where w_i are parameters

  Sometimes, different notation is used
  - $h_\theta(x) = \theta_0 + \theta_1 x$



Cost function

Q: How do we know that we perform well for the classification or regression problem?

A: Cost function!

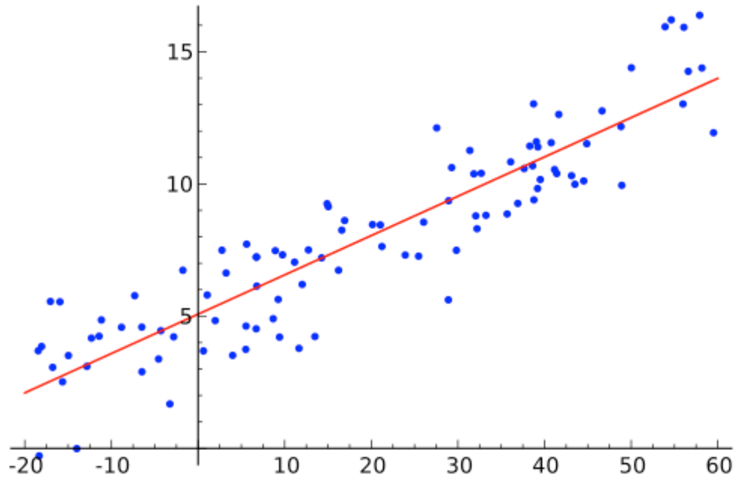
<u>Specifically</u>: Choose parameters w_i such that $h_w(x)$ is close to y for the training set values:

$$\min_{w_0, w_1} \frac{1}{2m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)} \right)^2$$

Therefore, our cost function is

defined as:

$$J(w_0, w_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_w(x^{(i)}) - y^{(i)} \right)^2$$

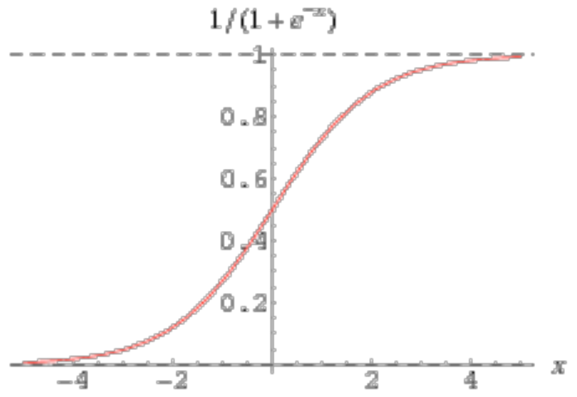


need to minimize it!

Basics of logistic regression

- Intuition: We want $0 \leq h_w(x) \leq 1$

- <u>Solution</u>: $h_w(x)=g(w^T x)$, where $g(z) = \dfrac{1}{1+e^{-z}}$ sigmoid (logistic) function



- We need to fit the parameters from w

- Interpretation: $h_w(x)=$ *estimated probability that* $y=1$, *on input* x

- <u>Formally</u>: $h_w(x)= P(y=1 \mid x;w)$, therefore $P(y=0 \mid x;w) = 1 - P(y=1 \mid x;w)$

Cost function

- To choose the parameters, let's go to the linear regression cost function and modify it:

$$J(w) = J(w_0, w_1, \ldots, w_n) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{1}{2}\left(h_w(x^{(i)}) - y^{(i)}\right)^2}_{\text{Replace by } cost(h_w(x^{(i)}), y)}$$

- Problem: If we leave the same formula (theoretically possible), our cost function becomes non-convex

- Gradient descent will most like get stacked at a local minimum

- Solution: Finding a cost function that is convex

$$cost(h_w(x), y) = \begin{cases} -\log(h_w(x)), \text{ if } y = 1 \\ -\log(1 - h_w(x)), \text{ if } y = 0 \end{cases}$$

Finalized cost function

$$J(w) = J(w_0, w_1, \dots, w_n) = \frac{1}{m} \sum_{i=1}^{m} cost(h_w(x), y)$$

$$= \frac{1}{m} \left[\sum_{i=1}^{m} -y \log(h_w(x)) - (1-y) \log(1 - h_w(x)) \right]$$

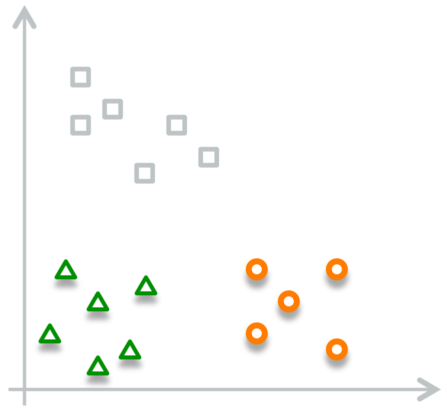
- Fitting parameters $w_0, w_1, \dots, w_n$: use Gradient Descent (algorithm looks identical to linear regression, but it is different in $h_w(x)$)

- To make a prediction given new x :

$h_w(x) = g(w^T x)$, where $g(z) = \frac{1}{1+e^{-Z}}$

Multiclass classification

- Emails: Spam detection vs. Foldering (GMAIL Folders: Main/Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
  Idea: convert multi-class to several binary classifications

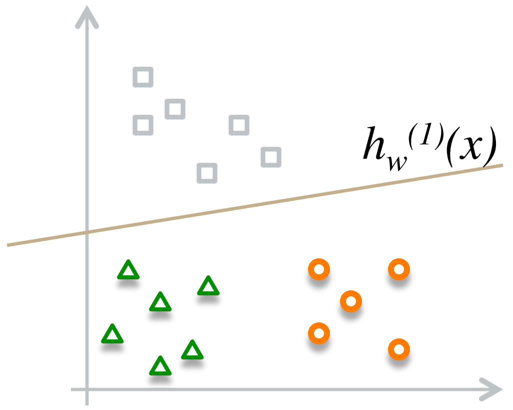


Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/ Social/Advertisement)

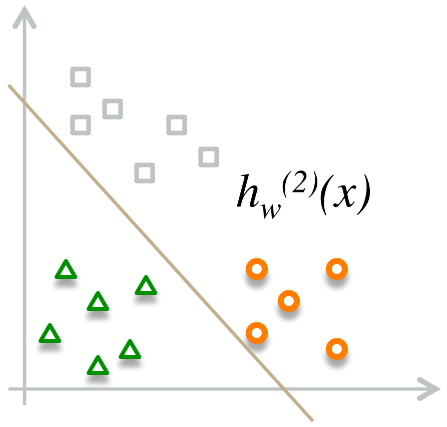
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile

- One-vs-all classification (one-vs-rest)
  Idea: convert multi-class to several binary classifications



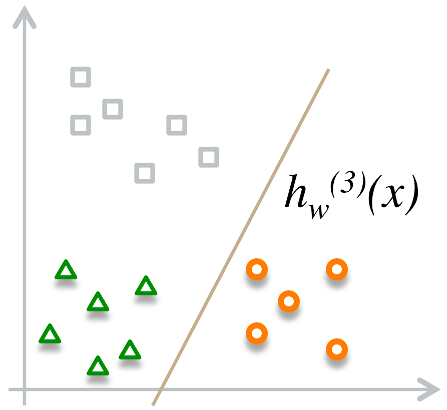
Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/ Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
  Idea: convert multi-class to several binary classifications



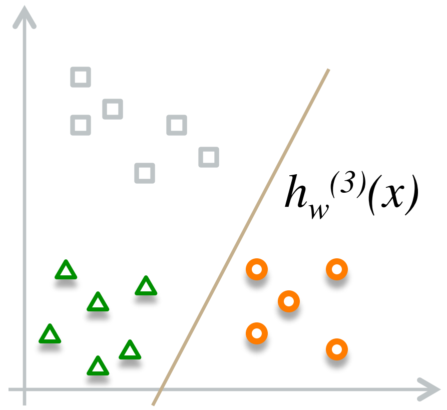
Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/ Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
  Idea: convert multi-class to several binary classifications



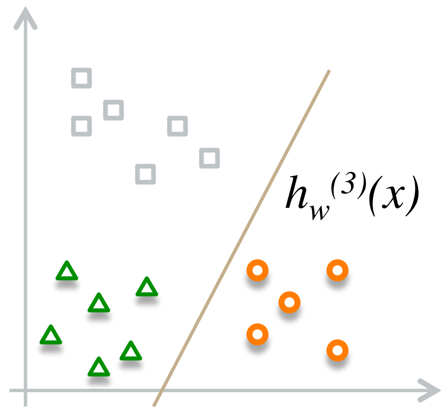
Multiclass classification

- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/ Social/Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
  Idea: convert multi-class to several binary classifications
- As a result we have 3 binary classifiers, each one is trained to recognize one class



Multiclass classification

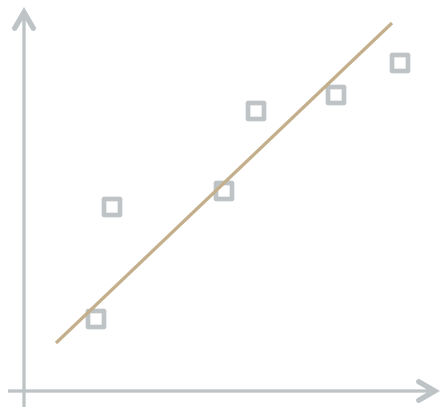
- Emails: Spa detection vs. Foldering (GMAIL Folders: Main/Social/ Advertisement)
- Disease: Healthy/Sick vs Healthy/Cold/Flu/West Nile
- One-vs-all classification (one-vs-rest)
  Idea: convert multi-class to several binary classifications
- As a result we have 3 binary classifiers, each one is trained to recognize one class



- For a new element: Apply all and select the most optimistic one:
  max $(h_w^{(i)}(x))$

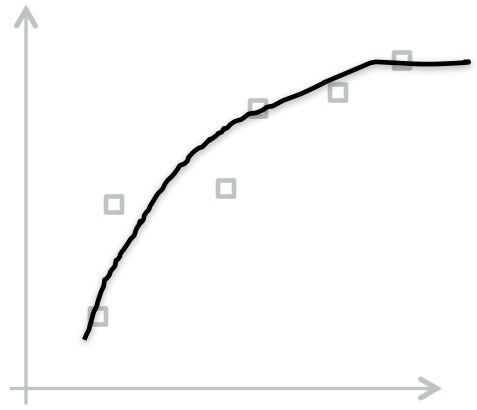
Overfitting and underfitting

- Consider linear regression
- Fitting a linear function
  - But the data "plateau"
  - Underfitting or "high bias"



Overfitting and underfitting

- Consider linear regression
- Fitting a linear function
  - But the data “plateau”
  - Underfitting or “high bias”
- Fitting a quadratic function
  - Seems to be a good fit
  - “Just right”



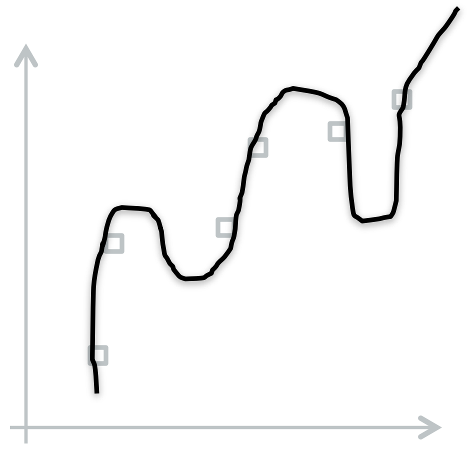
Overfitting and underfitting

- Consider linear regression

- Fitting a linear function
  - But the data “plateau”
  - Underfitting or “high bias”

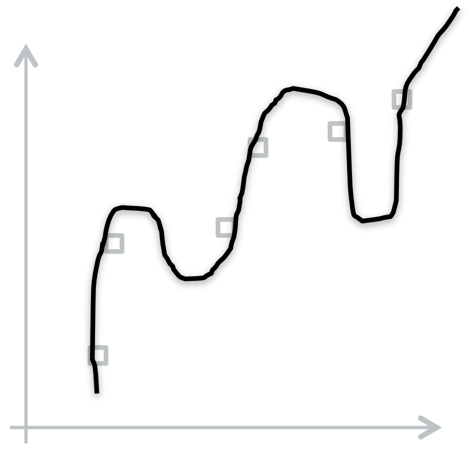
- Fitting a quadratic function
  - Seems to be a good fit
  - “Just right”

- Fitting a 4-degree polynomial
  - While the fit is perfect , the function does not look as a plausible solution
  - Overfitting or “high variance”: not enough data for the hypothesis



Overfitting and underfitting

- Consider linear regression
- Fitting a linear function
  - But the data “plateau”
  - Underfitting or “high bias”
- Fitting a quadratic function
  - Seems to be a good fit
  - “Just right”
- Fitting a 4-degree polynomial
  - While the fit is perfect , the function does not look as a plausible solution
  - Overfitting or “high variance”: not enough data for the hypothesis
- Same thing happens with the classification (for instance with linear regression)



Addressing overfitting

- If we have many features, the learned hypothesis may fit the data very well but fail to generalize (that is predict new examples)

- It is very difficult to address overfitting by visualization (especially when we have any features)

Addressing overfitting

- If we have many features, the learned hypothesis may fit the data very well but fail to generalize (that is predict new examples)

- It is very difficult to address overfitting by visualization (especially when we have any features)

- Solutions:
  - Reduce number of features
    - Manually select which features to keep
    - Use a model selection algorithm (will cover later in the course)

Addressing overfitting

- If we have many features, the learned hypothesis may fit the data very well but fail to generalize (that is predict new examples)
- It is very difficult to address overfitting by visualization (especially when we have any features)
- Solutions:
  - Reduce number of features
    - Manually select which features to keep
    - Use a model selection algorithm (will cover later in the course)
  - Regularization
    - Keep all the features but reduce the magnitude/values of parameters w_i
    - Works well when we have many features each of which contributes a little bit to the value of y

Intuition: Cost function

- Consider 2 hypothesis for our linear regression:
  - $w_0+w_1x+w_2x^2$: Just right
  - $w_0+w_1x+w_2x^2+w_3x^3+w_4x^4$: Overfitting

Intuition: Cost function

- Consider 2 hypothesis for our linear regression:
  - $w_0+w_1x+w_2x^2$: Just right
  - $w_0+w_1x+w_2x^2+w_3x^3+w_4x^4$: Overfitting

- Lets penalize and make w_3 and w_4 very small :

$$\min_{w_i} \frac{1}{2m}\sum_{i=1}^{m}\left(h_w(x^{(i)})-y^{(i)}\right)^2+1000w_3+1000w_4$$

- With this cost function we do need to make w_3 and w_4 very small. Thus, our 4-degree polynomial function will behave more like a quadratic one

Intuition: Cost function

- Consider 2 hypothesis for our linear regression:
  - $w_0+w_1x+w_2x^2$: Just right
  - $w_0+w_1x+w_2x^2+w_3x^3+w_4x^4$: Overfitting

- Lets penalize and make w_3 and w_4 very small :

$$\min_{w_i} \frac{1}{2m}\sum_{i=1}^{m}\left(h_w(x^{(i)})-y^{(i)}\right)^2+1000w_3+1000w_4$$

- With this cost function we do need to make w_3 and w_4 very small. Thus, our 4-degree polynomial function will behave more like a quadratic one

- Overall idea: small values for the parameters, which leads to simpler hypotheses, which are less prone to overfitting

Modified cost function

- The updated cost function to minimize:

$$J(w_0, w_1) = \frac{1}{2m}\left[\ \sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{i=1}^{n} w_i{}^2\ \right]$$

- Note that we have no *w_0*
  - This is by convention
  - In practice – little or no difference

- Parameter λ controls the trade-off between 2 terms
  - If λ is very large, then we will end up penalizing our parameters very heavily => $w_i \approx 0$ and $h_w(x) = w_0$, which is underfitting
  - If λ is very small, then no effect on the parameters => overfitting

Logistic regression is similar

$$J(w) = -\frac{1}{m}\left[\sum_{i=1}^{m} y\log(h_w(x)) + (1-y)\log(1-h_w(x))\right] + \frac{\lambda}{2m}\sum_{i=1}^{n} w_i^{\,2}$$

- Parameter lambda regulates how overcomplexified is the decision boundary

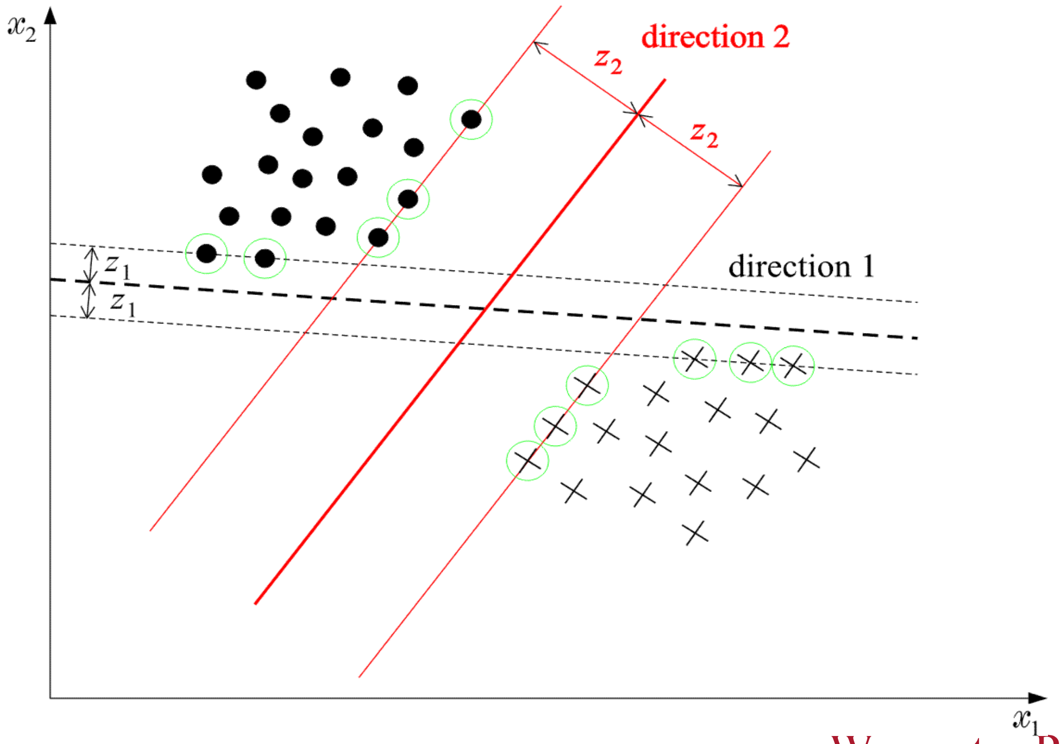
- GD is slightly different

Moving on: Support Vector Machines

Support Vector Machines

– The goal: Given two linearly separable classes, design the classifier

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$

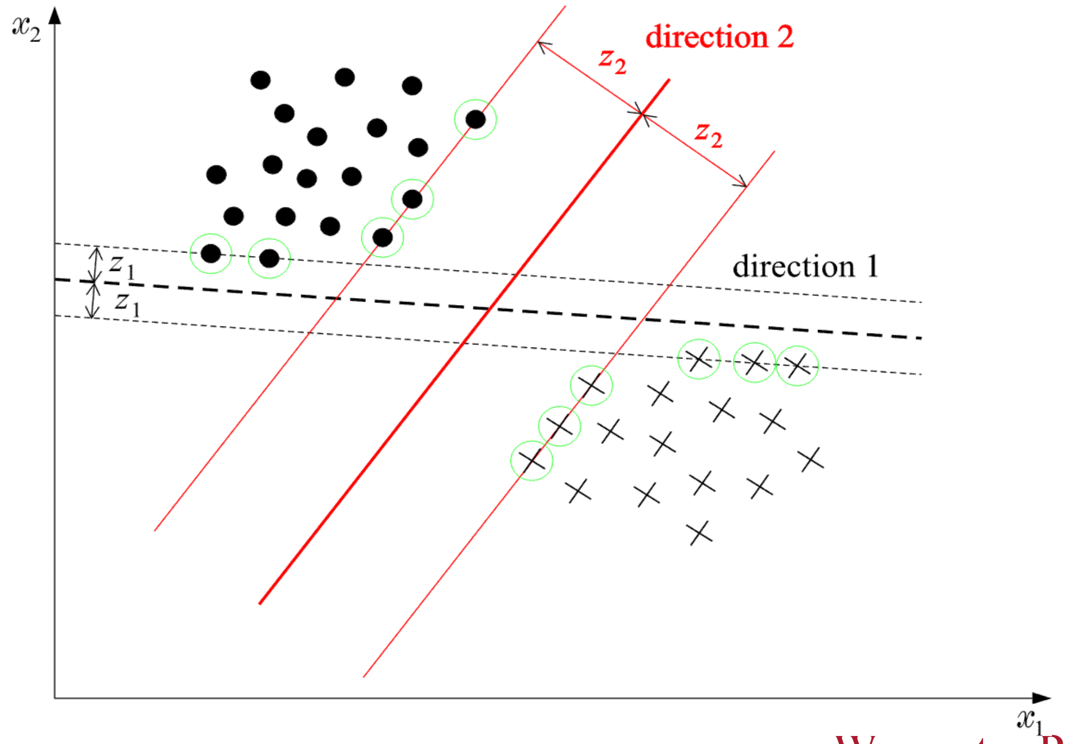


Support Vector Machines

– The goal: Given two linearly separable classes, design the classifier

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0$$

that leaves the maximum margin from both classes.



Margin intuition

- Margin: Each hyperplane is characterized by:
  - Its direction in space, i.e., $\boldsymbol{w}$
  - Its position in space, i.e., w_0
  - For EACH direction, $\boldsymbol{w}$, choose the hyperplane that leaves the SAME distance from the nearest points from each class. The margin is twice this distance.

Problem formulation

– The distance of a point $\hat{\boldsymbol{x}}$ from a hyperplane is given by:

$$z_{\hat{x}} = \frac{g(\hat{\boldsymbol{x}})}{||\boldsymbol{w}||}$$

– Scale $\boldsymbol{w}, \boldsymbol{w}_0$, so that at the nearest points, from each class, the discriminant function is ±1:

$$|g(\boldsymbol{x})| = 1 \quad \{g(\boldsymbol{x}) = +1 \text{ for } \omega_1 \text{ and } g(\boldsymbol{x}) = -1 \text{ for } \omega_2\}$$

– Thus the margin is given by:

$$\frac{1}{||\boldsymbol{w}||} + \frac{1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||}$$

– Also, the following is valid

$$\boldsymbol{w}^T\boldsymbol{x} + w_0 \geq 1 \quad \forall \boldsymbol{x} \in \boldsymbol{w}_1$$
$$\boldsymbol{w}^T\boldsymbol{x} + w_0 \leq -1 \quad \forall \boldsymbol{x} \in \boldsymbol{w}_2$$

SVM linear classifier

– Given

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

– Minimize

$$J(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2$$

– Subject to

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1, \ \ i = 1,2,\ldots,N$$

$$y_i = 1, \ \text{for } \boldsymbol{x}_i \in \omega_i,$$

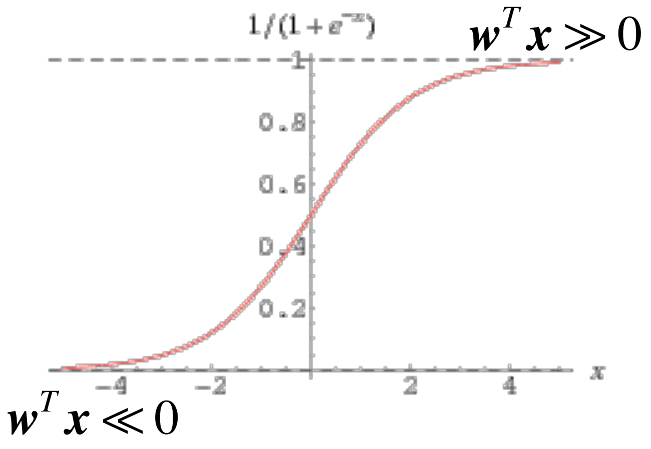
$$y_i = -1, \ \text{for } \boldsymbol{x}_i \in \omega_2$$

– The above is justified, since by minimizing $\frac{2}{||\boldsymbol{w}||}$

the margin $||\boldsymbol{w}||$ is maximized.

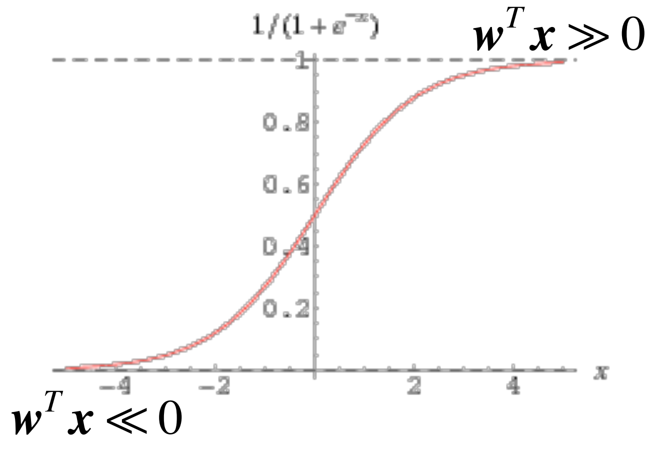
A different look at the same problem

1. Let's define the Optimization Objective
   - Recall logistic regression: $$h_w(x) = \frac{1}{1+e^{-w^T x}}$$
   - If y=1, we want $h_w(x) \sim 1$, that is: $\boldsymbol{w}^T \boldsymbol{x} \gg 0$
   - If y=0, we want $h_w(x) \sim 0$, that is: $\boldsymbol{w}^T \boldsymbol{x} \ll 0$



A different look at the same problem

1. Let's define the Optimization Objective
   - Recall logistic regression: $h_w(x) = \frac{1}{1+e^{-w^T x}}$
   - If y=1, we want $h_w(x) \sim 1$, that is: $\boldsymbol{w}^T \boldsymbol{x} \gg 0$
   - If y=0, we want $h_w(x) \sim 0$, that is: $\boldsymbol{w}^T \boldsymbol{x} \ll 0$

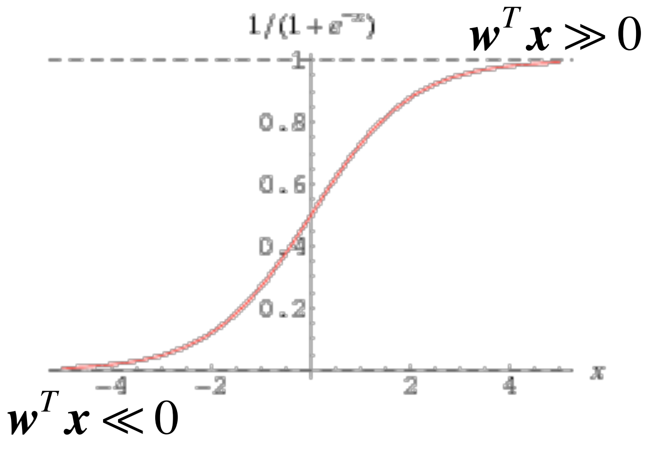


   - Recall cost of example:

$$cost(h_w(x), y) = -y\log(h_w(x)) - (1-y)\log(1-h_w(x))$$

A different look at the same problem

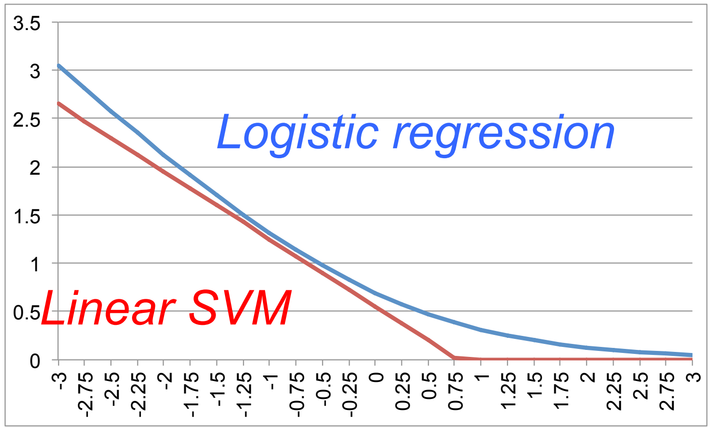
1. Let's define the Optimization Objective
   - Recall logistic regression: $h_w(x) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$
   - If y=1, we want $h_w(x) \sim 1$, that is: $\boldsymbol{w}^T\boldsymbol{x} \gg 0$
   - If y=0, we want $h_w(x) \sim 0$, that is: $\boldsymbol{w}^T\boldsymbol{x} \ll 0$



   - Recall cost of example:

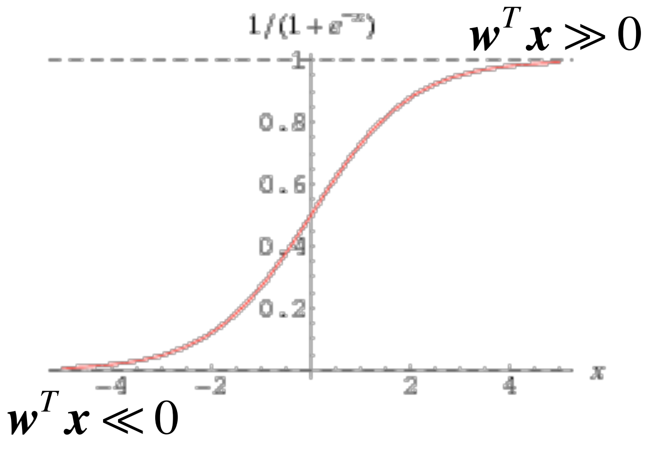
$$cost(h_w(x), y) = -y\log(h_w(x)) - (1-y)\log(1-h_w(x))$$

   - If y =1 $\quad cost = -\log\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$



A different look at the same problem

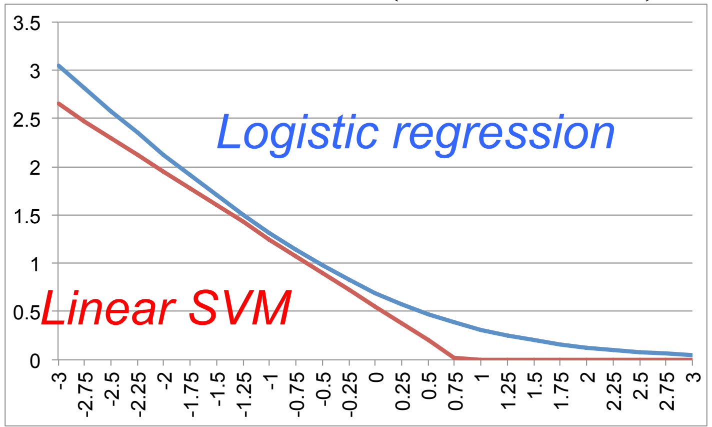
1. Let's define the Optimization Objective
   - Recall logistic regression: $h_w(x) = \dfrac{1}{1+e^{-w^T x}}$
   - If y=1, we want $h_w(x) \sim 1$, that is: $w^T x \gg 0$
   - If y=0, we want $h_w(x) \sim 0$, that is: $w^T x \ll 0$

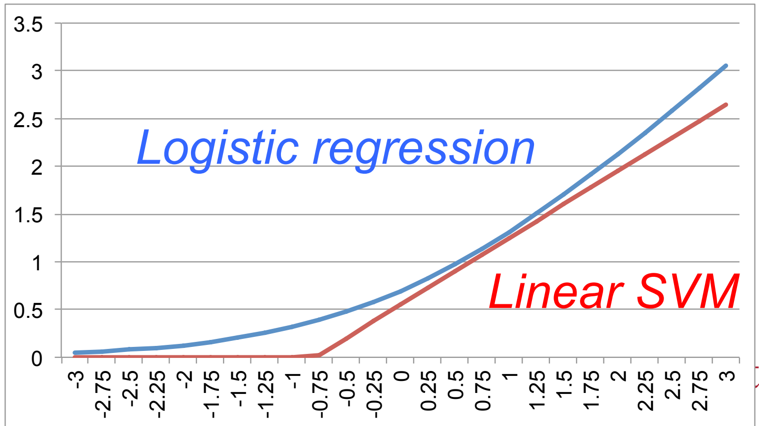


   - Recall cost of example:

$$cost(h_w(x), y) = -y\log(h_w(x)) - (1-y)\log(1-h_w(x))$$

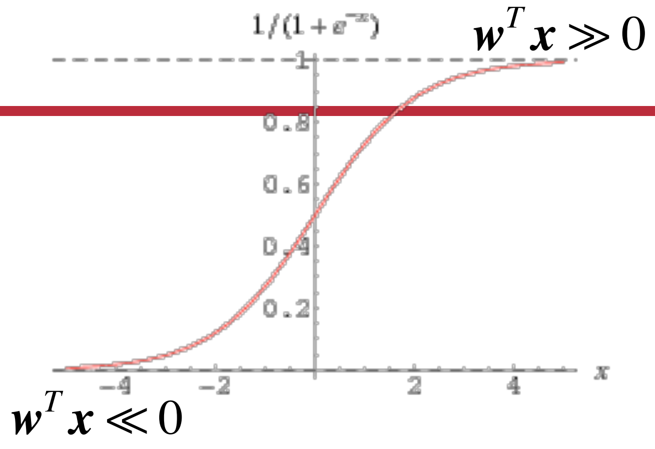
   - If y =1, $cost = -\log \dfrac{1}{1+e^{-w^T x}}$
   - If y =0, $cost = -\log\left(1 - \dfrac{1}{1+e^{-w^T x}}\right)$





A different look at the same problem

1. Let's define the Optimization Objective
   - Recall logistic regression: $h_w(x) = \dfrac{1}{1+e^{-w^T x}}$

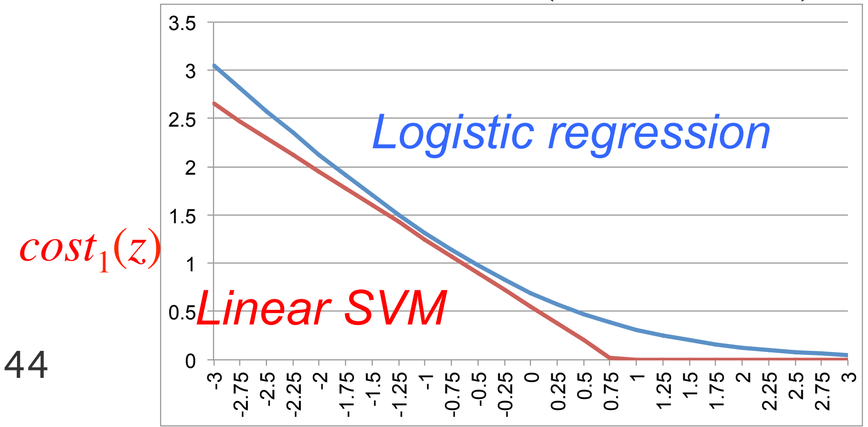


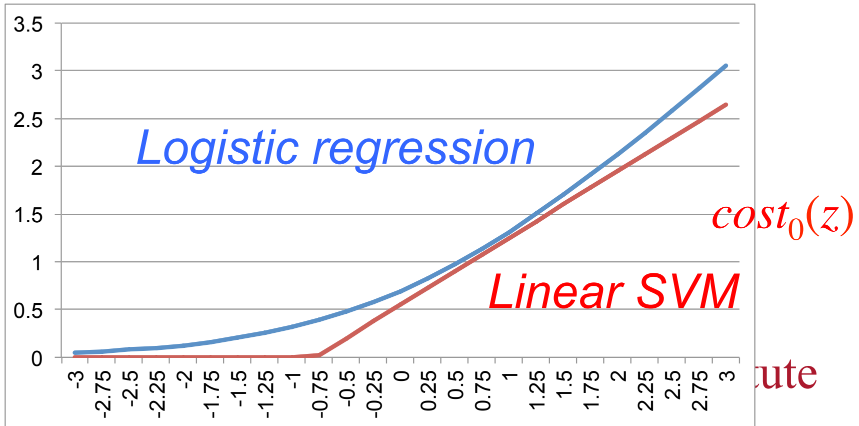
   - If y=1, we want $h_w(x) \sim 1$, that is: $w^T x \gg 0$
   - If y=0, we want $h_w(x) \sim 0$, that is: $w^T x \ll 0$

   - Recall cost of example:

$$cost(h_w(x), y) = -y\log(h_w(x)) - (1-y)\log(1-h_w(x))$$

   - If y =1, $cost = -\log \dfrac{1}{1+e^{-w^T x}}$
   - If y =0, $cost = -\log\left(1 - \dfrac{1}{1+e^{-w^T x}}\right)$





SVM cost function

Logistic regression:

$$\min_{w} \boxed{\frac{1}{m}} \left[\sum_{i=1}^{m} -y^{(i)} \boxed{\log(h_w(x^{(i)}))} + (1-y^{(i)}) \boxed{\log(1-h_w(x^{(i)}))} \right] + \boxed{\frac{\lambda}{2m}} \sum_{i=1}^{n} w_i^2$$

SVM cost function

Logistic regression:

$$\min_{w} \frac{1}{m}\left[\sum_{i=1}^{m} -y^{(i)}\log(h_w(x^{(i)})) + (1-y^{(i)})\log(1-h_w(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{i=1}^{n} w_i^{\,2}$$

SVM:

$$\min_{\boldsymbol{w}} C\left[\sum_{i=1}^{m} y^{(i)} cost_1(\boldsymbol{w}^T \boldsymbol{x}^{(i)}) + (1-y^{(i)}) cost_0(\boldsymbol{w}^T \boldsymbol{x}^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n} w_i^{\,2}$$

SVM cost function

Logistic regression:

$$\min_{w} \boxed{\frac{1}{m}} \left[\sum_{i=1}^{m} -y^{(i)} \boxed{\log(h_w(x^{(i)}))} + (1-y^{(i)}) \boxed{\log(1-h_w(x^{(i)}))} \right] + \boxed{\frac{\lambda}{2m}} \sum_{i=1}^{n} {w_i}^2$$

SVM:

$$\min_{w} C \left[\sum_{i=1}^{m} y^{(i)} cost_1(\boldsymbol{w}^T \boldsymbol{x}^{(i)}) + (1-y^{(i)}) cost_0(\boldsymbol{w}^T \boldsymbol{x}^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} {w_i}^2$$

Note: Moving from $\mathrm{A}+\lambda\mathrm{B}$ form to $c\mathrm{A}+\mathrm{B}$ form

Remarks

- SVMs are often referred to as *Large margin classifiers*
- Based on the cost function*:*
  - If y=1, we want: $\boldsymbol{w}^T\boldsymbol{x} \geq 1$ (not just $\boldsymbol{w}^T\boldsymbol{x} \geq 0$)
  - If y=0, we want: $\boldsymbol{w}^T\boldsymbol{x} \leq -1$ (not just $\boldsymbol{w}^T\boldsymbol{x} \leq 0$)

Boundary condition

Remarks

- SVMs are often referred to as *Large margin classifiers*
- Based on the cost function:
  - If y=1, we want: $\boldsymbol{w}^T\boldsymbol{x} \geq 1$ (not just $\boldsymbol{w}^T\boldsymbol{x} \geq 0$)
  - If y=0, we want: $\boldsymbol{w}^T\boldsymbol{x} \leq -1$ (not just $\boldsymbol{w}^T\boldsymbol{x} \leq 0$)

  Boundary condition

- For our cost function:

$$\min_{\boldsymbol{w}} \underbrace{C\left[\sum_{i=1}^{m} y^{(i)} cost_1(\boldsymbol{w}^T\boldsymbol{x}^{(i)}) + (1-y^{(i)}) cost_0(\boldsymbol{w}^T\boldsymbol{x}^{(i)})\right]}_{= CA} + \underbrace{\frac{1}{2}\sum_{i=1}^{n} w_i^{\,2}}_{= \mathrm{B}}$$

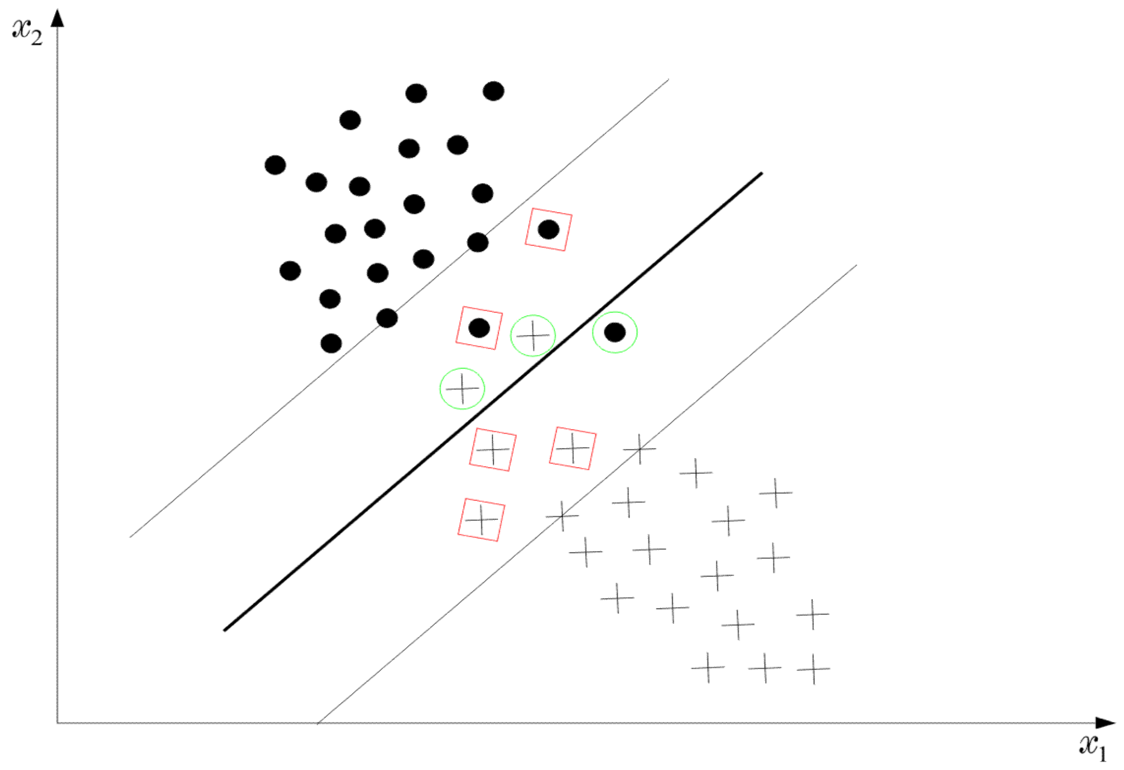
  - If C>>0, we want A=0, which implies: $\min_{\boldsymbol{w}} \frac{1}{2}\sum_{i=1}^{n} w_i^{\,2}$
  - Combined with boundary condition, this optimization will give us the largest minimum distance (margin!) between pos. and neg. examples

Problem with our approach

- If you simply optimize the large margin value (given a large C), the algorithm is sensitive to the outliers!

- How to fix it? Have C not very large, so it can ignore several outliers

Non–Separable classes



Non-Separable classes

In this case, there is no hyperplane such that:

$$\boldsymbol{w}^T \boldsymbol{x} + w_0 (><) 1, \quad \forall \boldsymbol{x}$$

- Recall that the margin is defined as twice the distance between the following two hyperplanes:

$$\boldsymbol{w}^T \boldsymbol{x} + w_0 = 1$$

$$\text{and}$$

$$\boldsymbol{w}^T \boldsymbol{x} + w_0 = -1$$

The training vectors belong to one of three possible categories

1) Vectors outside the band which are correctly classified:

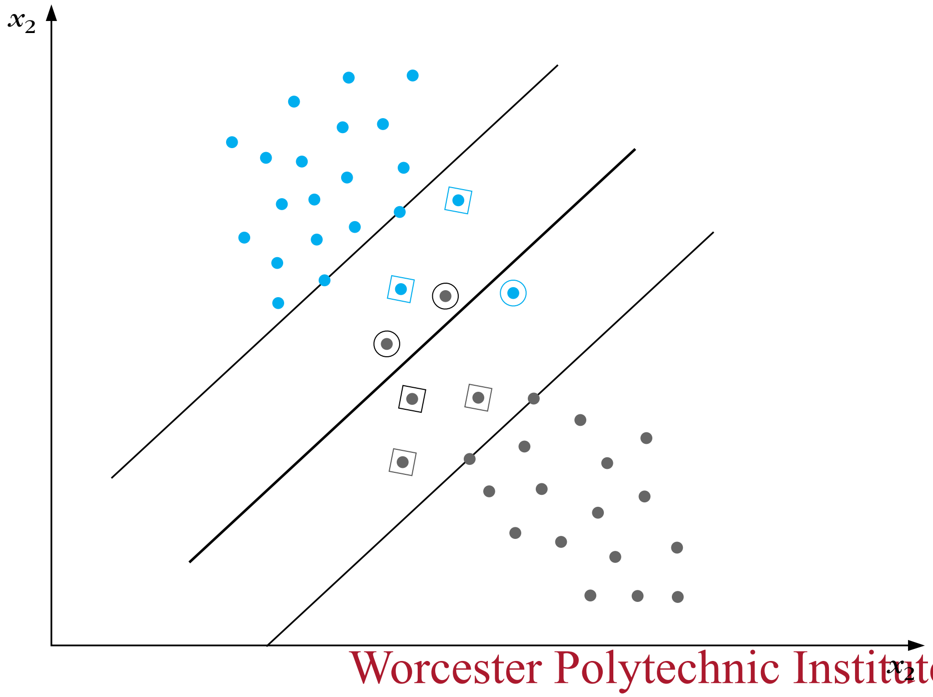
$$y_i(\boldsymbol{w}^T\boldsymbol{x}+w_0)>1$$

2) Vectors inside the band, and correctly classified:

$$0\leq y_i(\boldsymbol{w}^T\boldsymbol{x}+w_0)<1$$

3) Vectors misclassified:

$$y_i(\boldsymbol{w}^T\boldsymbol{x}+w_0)<0$$



Slack variables

All three cases above can be represented as:

$$y_i(\boldsymbol{w}^T\boldsymbol{x} + w_0) \geq 1 - \xi_i$$

1) $\rightarrow \xi_i = 0$
2) $\rightarrow 0 < \xi_i \leq 1$
3) $\rightarrow 1 < \xi_i$

ξ_i are known as slack variables.

Reformulating optimization goals

The goal of the optimization is now two-fold:

- Maximize margin
- Minimize the number of patterns with $\xi_i > 0$.

One way to achieve this goal is via the cost

$$J(\boldsymbol{w}, w_0, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

where *C* is a constant, and

$$I(\xi_i) = \begin{Bmatrix} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{Bmatrix}$$

Note: *I*(.) is not differentiable. In practice, we use an approximation.
A popular choice is:

$$J(\boldsymbol{w}, w_0, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$

- Following a similar procedure as before, we obtain KKT conditions

Multi–class generalization

- Although theoretical generalizations exist, the most popular in practice is to look at the problem as M two-class problems (one-against-all)

- Similar to the logistic regression case, for a new element x, we will select the most optimistic classifier

- Problem 1: Asymmetric case of positive/negative classes

- Problem 2: Uneven number of positive/negative examples, especially when the number of classes is large

Steps to try: Revisitting

1. Get more training data: Fixes high variance
2. Try smaller set of features: Fixes high variance
3. Try getting additional features: Fixes high bias
4. Try adding polynomial features: Fixes high bias
5. Try decreasing λ: Fixes high bias
6. Try increasing λ: Fixes high variance

Logistic regression vs. SVM: What to select and when?

Reminder: n – number of features, m - number of training examples

1. n is large (relative to m): *e.g.*, $n = 10{,}000 - 100{,}000$, $m = 10 - 1000$
   Use: Logistic regression or SVM without kernels

2.

3.

Logistic regression vs. SVM: What to select and when?

Reminder: n – number of features, m - number of training examples

1. n is large (relative to m): *e.g.*, $n = 10{,}000 - 100{,}000,\ m = 10 - 1000$
   Use: Logistic regression or SVM without kernels

2. n is small, m is intermediate: *e.g.*, $n = 1 - 1{,}000,\ m = 10 - 10000$
   Use: Use SVM with Gaussian kernel

3.

Logistic regression vs. SVM: What to select and when?

Reminder: n – number of features, m - number of training examples

1. n is large (relative to m): *e.g.*, $n = 10{,}000 - 100{,}000,\ m = 10 - 1000$
   Use: Logistic regression or SVM without kernels

2. n is small, m is intermediate: *e.g.*, $n = 1 - 1{,}000,\ m = 10 - 10000$
   Use: Use SVM with Gaussian kernel

3. n is small, m is large: *e.g.*, $n = 1 - 1{,}000,\ m = 50{,}000 - 1{,}000{,}000$
   Use: Create/add more features, then use logistic regression or SVM without kernels
   Reason: SVM with Gaussian kernel would struggle to run

Combining Classifiers

- The basic philosophy behind the combination of different classifiers lies in the fact that even the “best” classifier fails in some patterns that other classifiers may classify correctly
- Combining classifiers aims at exploiting this complementary information residing in the various classifiers

–

Combining Classifiers

- The basic philosophy behind the combination of different classifiers lies in the fact that even the “best” classifier fails in some patterns that other classifiers may classify correctly
- Combining classifiers aims at exploiting this complementary information residing in the various classifiers

Idea: one designs different optimal classifiers and then combines the results with a specific rule

– Assume that each of the, say, L designed classifiers provides at its output the posterior probabilities:

$$P(\omega_i \mid \boldsymbol{x}),\ i = 1, 2, \ldots, M$$

Basic Rules

- Product Rule: Assign $\boldsymbol{x}$ to the class ω_i:

$$i = \arg\max_k \prod_{j=1}^{L} P_j\left(\omega_k \mid \boldsymbol{x}\right)$$

where $P_j\left(\omega_k \mid \boldsymbol{x}\right)$ is the respective posterior probability of the j^{th} classifier.

Basic Rules

- Product Rule: Assign $\boldsymbol{x}$ to the class ω_i:

$$i = \arg\max_k \prod_{j=1}^{L} P_j\left(\omega_k \mid \boldsymbol{x}\right)$$

where $P_j\left(\omega_k \mid \boldsymbol{x}\right)$ is the respective posterior probability of the j^{th} classifier

- Sum Rule: Assign $\boldsymbol{x}$ to the class ω_i:

$$i = \arg\max_k \sum_{j=1}^{L} P_j\left(\omega_k \mid \boldsymbol{x}\right)$$

Majority Voting Rule

Majority Voting Rule: Assign $\boldsymbol{x}$ to the class for which there is a consensus or when at least ℓ_c of the classifiers agree on the class label of $\boldsymbol{x}$ where:

$$\ell_c = \begin{cases} \frac{L}{2}+1, & L \text{ is even} \\ \frac{L+1}{2}, & L \text{ is odd} \end{cases}$$

otherwise the decision is rejection, that is no decision is taken.

Majority Voting Rule

Majority Voting Rule: Assign $\boldsymbol{x}$ to the class for which there is a consensus or when at least ℓ_c of the classifiers agree on the class label of $\boldsymbol{x}$ where:

$$\ell_c = \begin{cases} \dfrac{L}{2}+1, \ L \text{ is even} \\ \dfrac{L+1}{2}, \ L \text{ is odd} \end{cases}$$

otherwise the decision is rejection, that is no decision is taken.

Thus, correct decision is made if the majority of the classifiers agree on the correct label, and wrong decision if the majority agrees in the wrong label

Dependent or not Dependent classifiers?

- Although there are not general theoretical results, experimental evidence has shown that the more independent in their decision the classifiers are, the higher the expectation should be for obtaining improved results after combination

- However, there is no guarantee that combining classifiers results in better performance compared to the "best" one among the classifiers

Towards Independence: A number of Scenarios

Train the individual classifiers using different training data points

To this end, we can choose among a number of possibilities:

- Bootstrapping: This is a popular technique to combine unstable classifiers such as decision trees (Bagging belongs to this category of combination)

- Stacking: Train the combiner with data points that have been excluded from the set used to train the individual classifiers

- Use different subspaces to train individual classifiers: According to the method, each individual classifier operates in a different feature subspace. That is, use different features for each classifier

Remarks

- The majority voting and the summation schemes rank among the most popular combination schemes
- Training individual classifiers in different subspaces seems to lead to substantially better improvements compared to classifiers operating in the same subspace
- Besides the above three rules, other alternatives are also possible, such as to use the median value of the outputs of individual classifiers

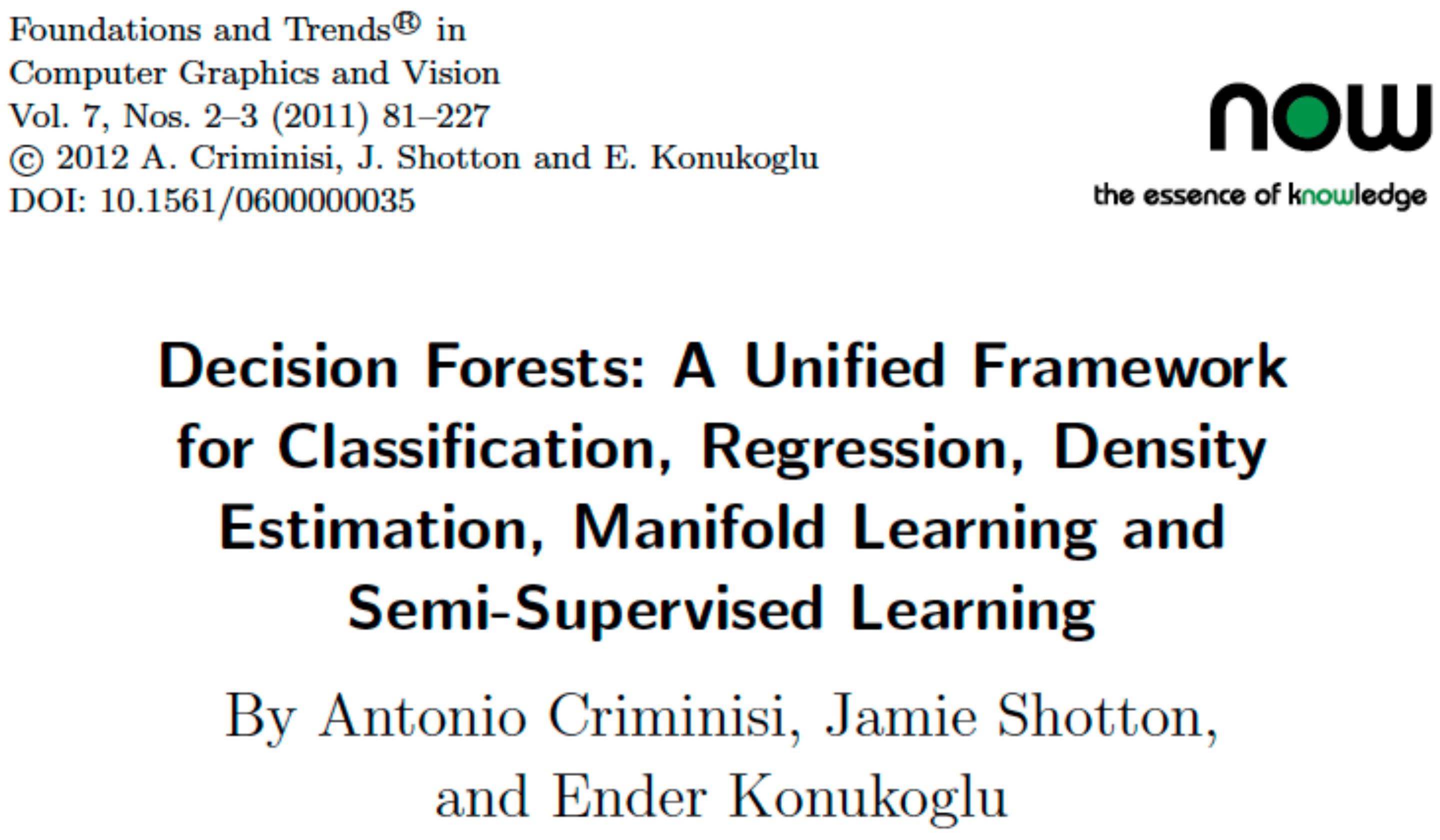
The Boosting Approach

The origins: Is it possible a weak learning algorithm (one that performs slightly better than a random guessing) to be boosted into a strong algorithm? (Villiant 1984)

- The procedure to achieve it:
  - Adopt a weak classifier known as the base classifier.
  - Employing the *base* classifier, design a series of classifiers, in a hierarchical fashion, each time employing a different weighting of the training samples. Emphasis in the weighting is given on the hardest samples, *i.e.*, the ones that keep "failing".
  - Combine the hierarchically designed classifiers by a weighted average procedure

Decision Tree —>Random Forrest

Slides are adapted based on:



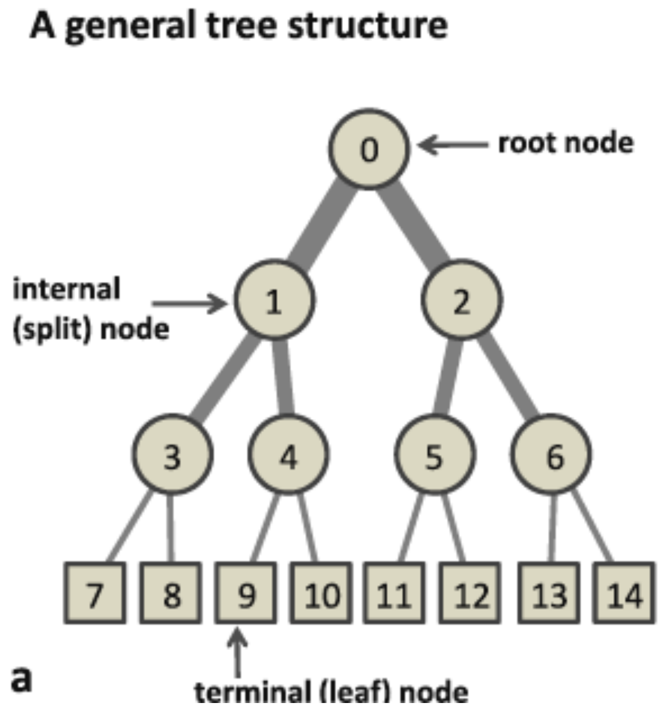
Foundations and Trends® in
Computer Graphics and Vision
Vol. 7, Nos. 2–3 (2011) 81–227
© 2012 A. Criminisi, J. Shotton and E. Konukoglu
DOI: 10.1561/0600000035

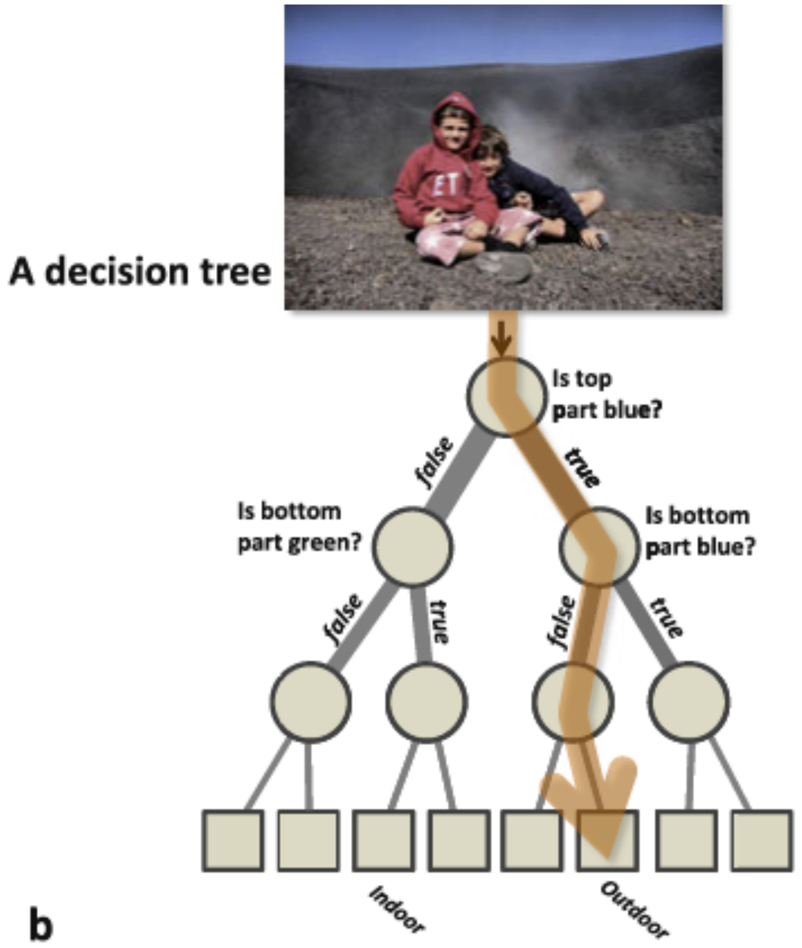
now
the essence of knowledge

Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning

By Antonio Criminisi, Jamie Shotton, and Ender Konukoglu

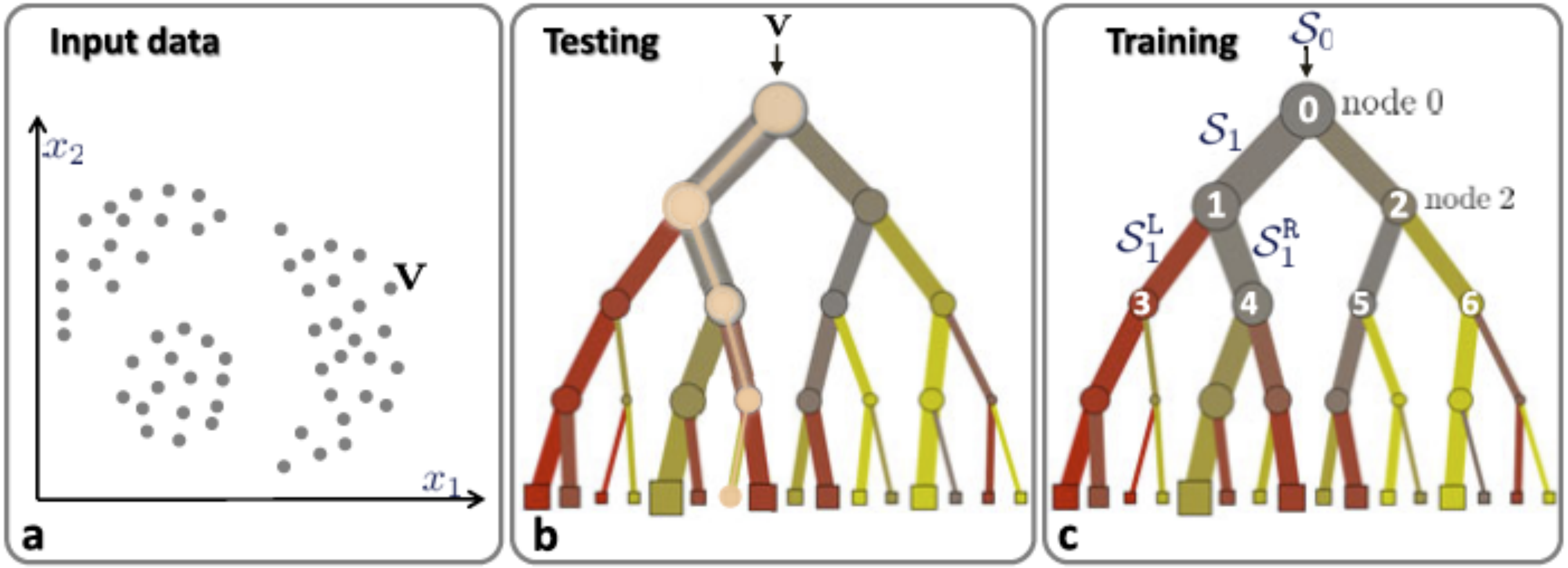
Decision tree





- No loops. Internal nodes circles. Terminal nodes with squares
- Each internal node stores a split (or test) function to be applied to the incoming data
- Each leaf stores the final answer (predictor)
- <u>Example</u>: figure out whether a photo represents an indoor or outdoor scene

Basic notation



(a) Input data are represented as a collection of points in the *d*-dimensional space defined by their feature responses (2D in this example)

(b) During testing, a split (internal) node applies a test to the input data *v* and sends it to the appropriate child. The process is repeated until a leaf (terminal) node is reached (beige path)

(c) Training a decision tree involves sending the entire training set S_0 into the tree and optimizing the parameters of the split nodes so as to optimize a chosen energy function

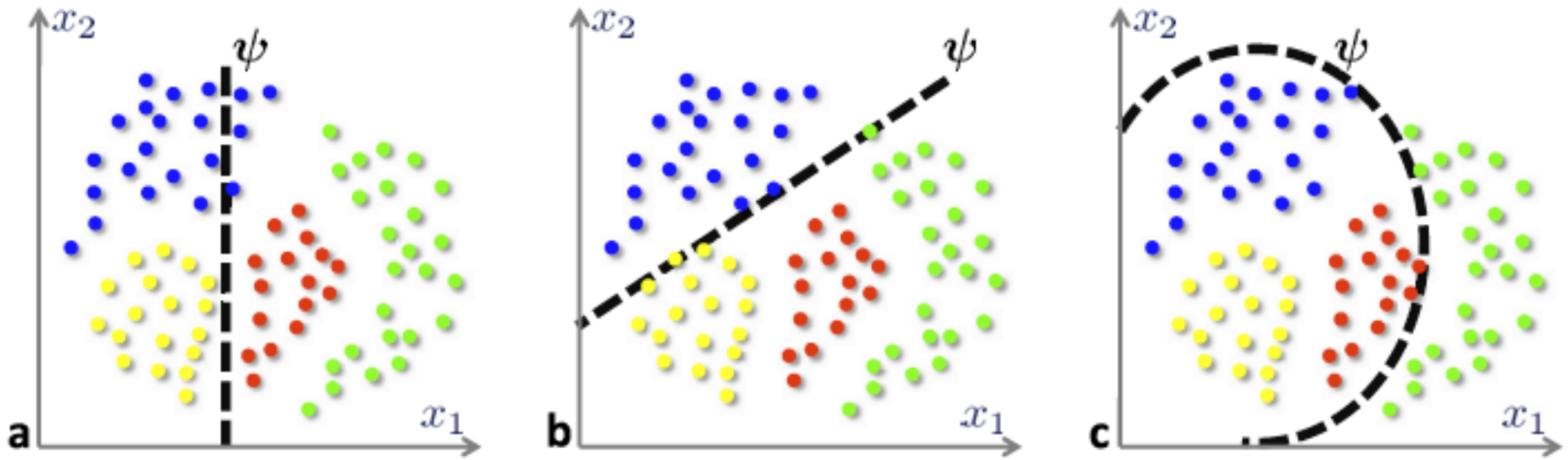
Learning tree structure

- The structure of the tree depends on how and when we decide to stop growing various branches of the tree

- Diverse stopping criteria can be applied. For example it is common to stop the tree when a maximum number of levels D has been reached. Tree growing may also be stopped when a node contains too few training points

- At the end of the training phase we obtain:
  - (i) the (greedily) optimum weak learners (split functions) associated with each node
  - (ii) a learned tree structure,
  - (iii) a different set of training points at each leaf

Weak learner



Example: colors of each data point (*circles*) indicate different classes

(a) Axis-aligned hyperplane weak learner

(b) General oriented hyperplane

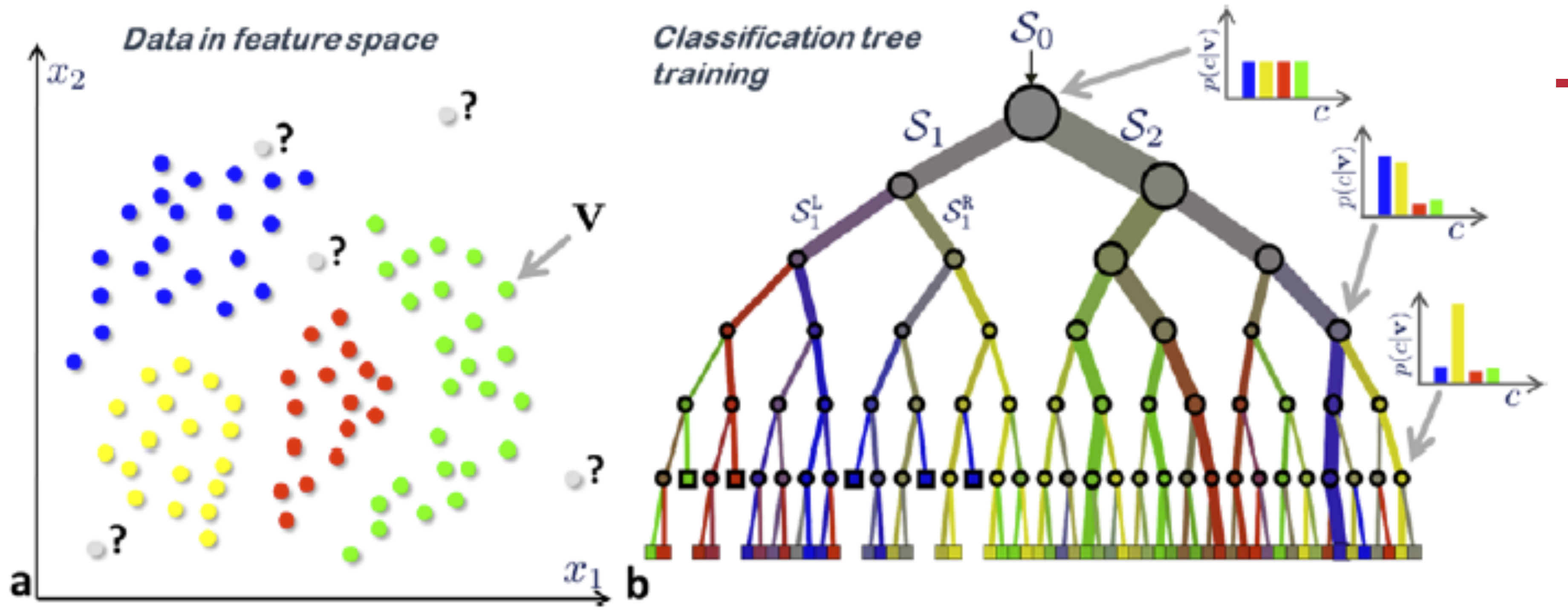
(c) Quadratic surface

In general, $\psi(\mathbf{v})$ has a dimensionality ≤ 2, irrespective of number of features

Random Forests: Basics and historical notes

- Random forests are an ensemble learning method for classification / regression
- Idea: generalization of decision trees
- Algorithm for inducing a random forest was developed by Leo Breiman and Adele Cutler in 2001
- "Random Forests" is their trademark
- The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995
- Algorithm combines Breiman's "bagging" idea and the random selection of features, introduced by Ho, Amit and Geman for constructing decision trees with controlled variation.
- In addition, the idea of randomized node optimization, where the decision at each node is selected by a randomized procedure introduced by Dietterich
- More recently several major advances in this area have come from Microsoft Research

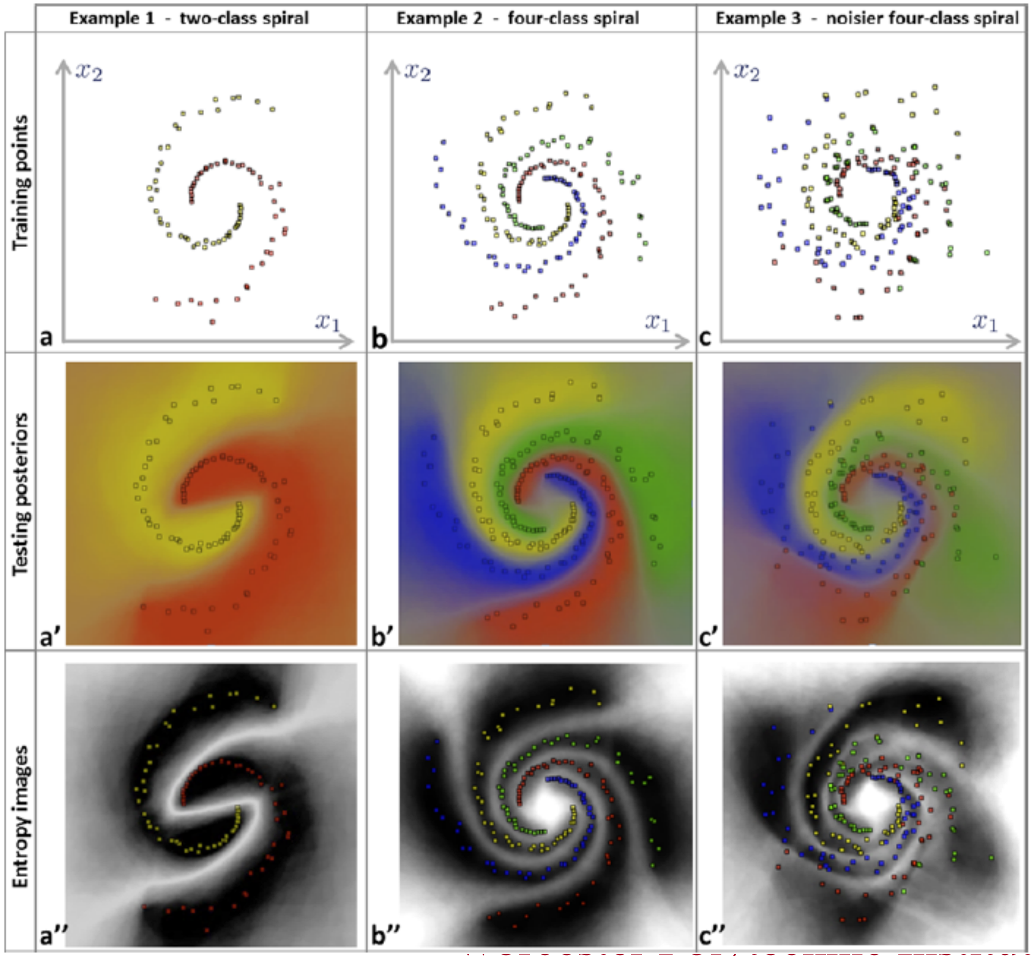
Classification training data and tree training



- *Gray circles* indicate unlabeled, previously unseen test data
- The edge thickness is proportional to the amount of training data going through it
- Edge colors are a mix of the colors of the four classes, weighted in proportion to the associated class probabilities
- Note the gray-ish color of the root node and the more distinct colors of the leaves

Multiple classification

- One major advantage of decision forests over *e.g.* support vector machines and boosting is that the **same** classification model can handle both binary and multi-class problems
- Shown are both two- and four-class examples, and different levels of noise in the training data

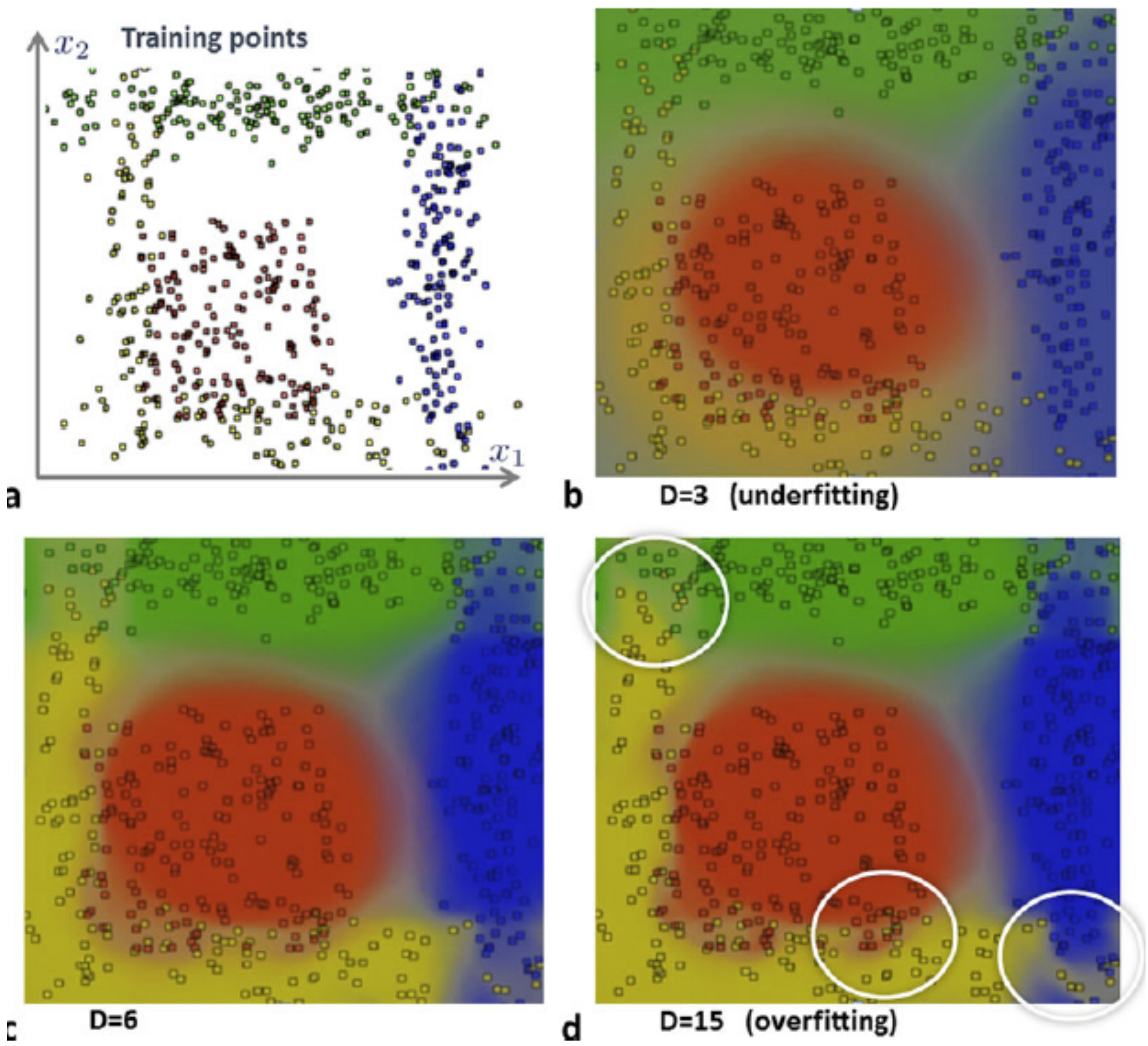


The effect of tree depth

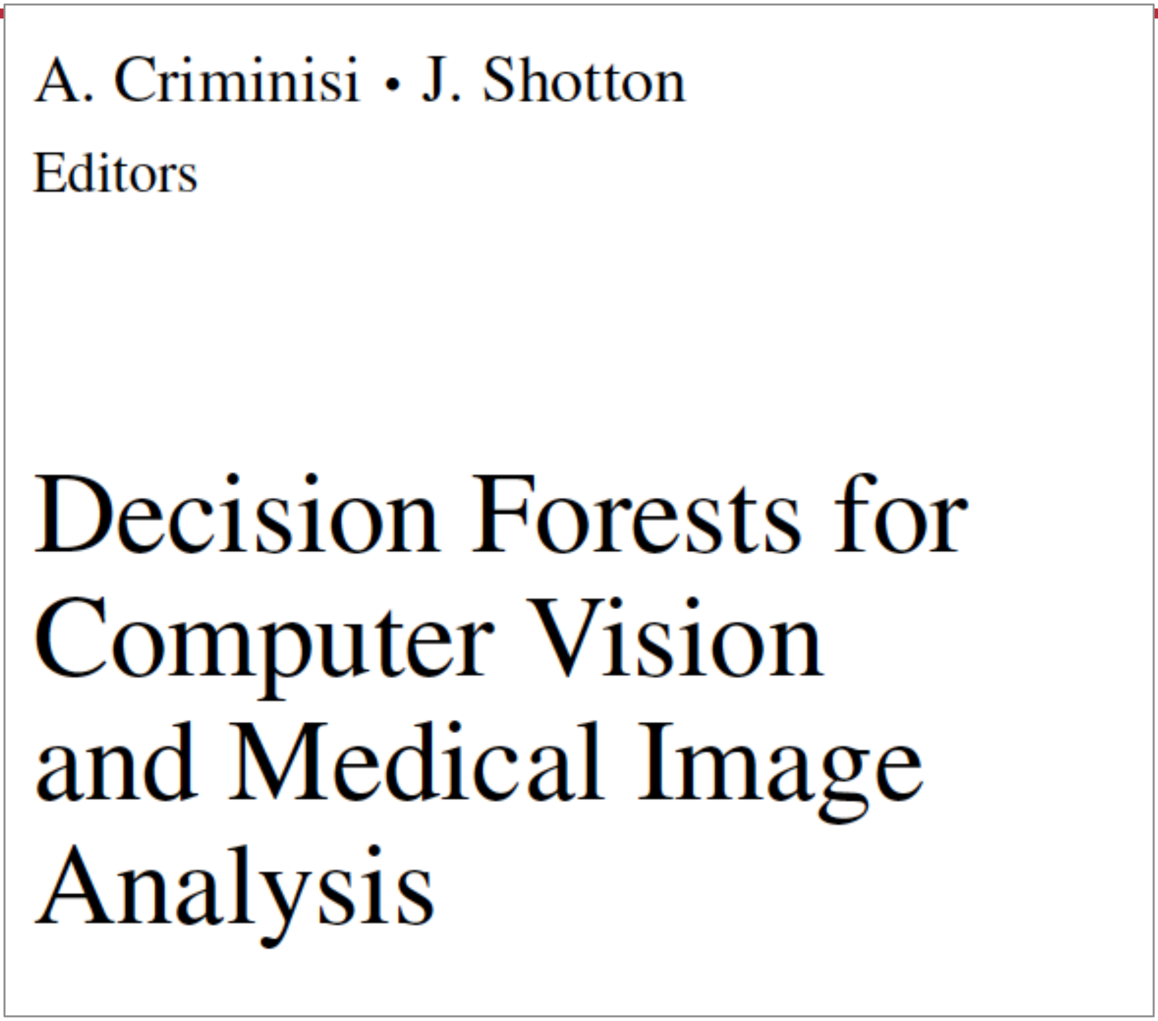
- A four-class problem with both mixing of training labels and large gaps
- The tree depth is a crucial parameter in avoiding under- or overfitting

(**a**) Training points

(**b**, **c**, **d**): Different tree depths D



More information


A. Criminisi • J. Shotton

Editors

Decision Forests for Computer Vision and Medical Image Analysis

Semi-Supervised Learning

Semi-Supervised Learning: Basics

- In addition to unlabeled data, the algorithm is provided with some supervision information
  - but not necessarily for all examples
- Data set $X = (x_i)_{i \in [n]}$ is thus split into two subsets:
  - Labeled set $X_l := (x_1, \ldots, x_l)$ $Y_l := (y_1, \ldots, y_l)$
  - Unlabeled set $X_u := (x_{l+1}, \ldots, x_{l+u})$
- A problem related to SSL was introduced by Vapnik already several decades ago: *transductive learning*
  - In this setting, one is also given a (labeled) training set and an (unlabeled) test set
  - The idea of transduction is to perform predictions **only** for the test inductive learning points
  - This is in contrast to *inductive learning*, where the goal is to output a prediction function which is defined on the entire space
- Many SSL algorithms are transductive learning algorithms

When Can SSL work?

- One of the most common criteria: Smoothness assumption

- Supervised learning: If two points x_1; x_2 are close, then so should be the corresponding outputs y_1; y_2
  - Strictly speaking, this assumption only refers to continuity rather than smoothness; however, the term smoothness is commonly used, possibly because in regression estimation y is often modeled in practice as a smooth function of x

- SSL: If two points x_1; x_2 in a **high-density region** are close, then so should be the corresponding outputs y_1; y_2
  - Note that by transitivity, this assumption implies that if two points are linked by a path of high density (e.g., if they belong to the same cluster), then their outputs are likely to be close
  - If, on the other hand, they are separated by a **low-density region**, then their outputs need not be close

The cluster assumption

- Suppose we knew that the points of each class tended to form a cluster

- Then the unlabeled data could aid in finding the boundary of each cluster more accurately: one could run a clustering algorithm and use the labeled points to assign a class to each cluster

- ***Cluster Assumption:*** If points are in the same cluster, they are likely to be of the same class
  - Note that the cluster assumption does not imply that each class forms a single, compact cluster: it only means that, usually, we do not observe objects of two distinct classes in the same cluster
  - The cluster assumption can easily be seen as a special case of the above-proposed semi-supervised smoothness assumption

Classes of SSL algorithms

1. Self-learning algorithms

2. Generative models
   - Mixture models
   - Data-dependent priors

3. Low-density separation (LDS)

4. Graph-based methods

5. Change of representation methods

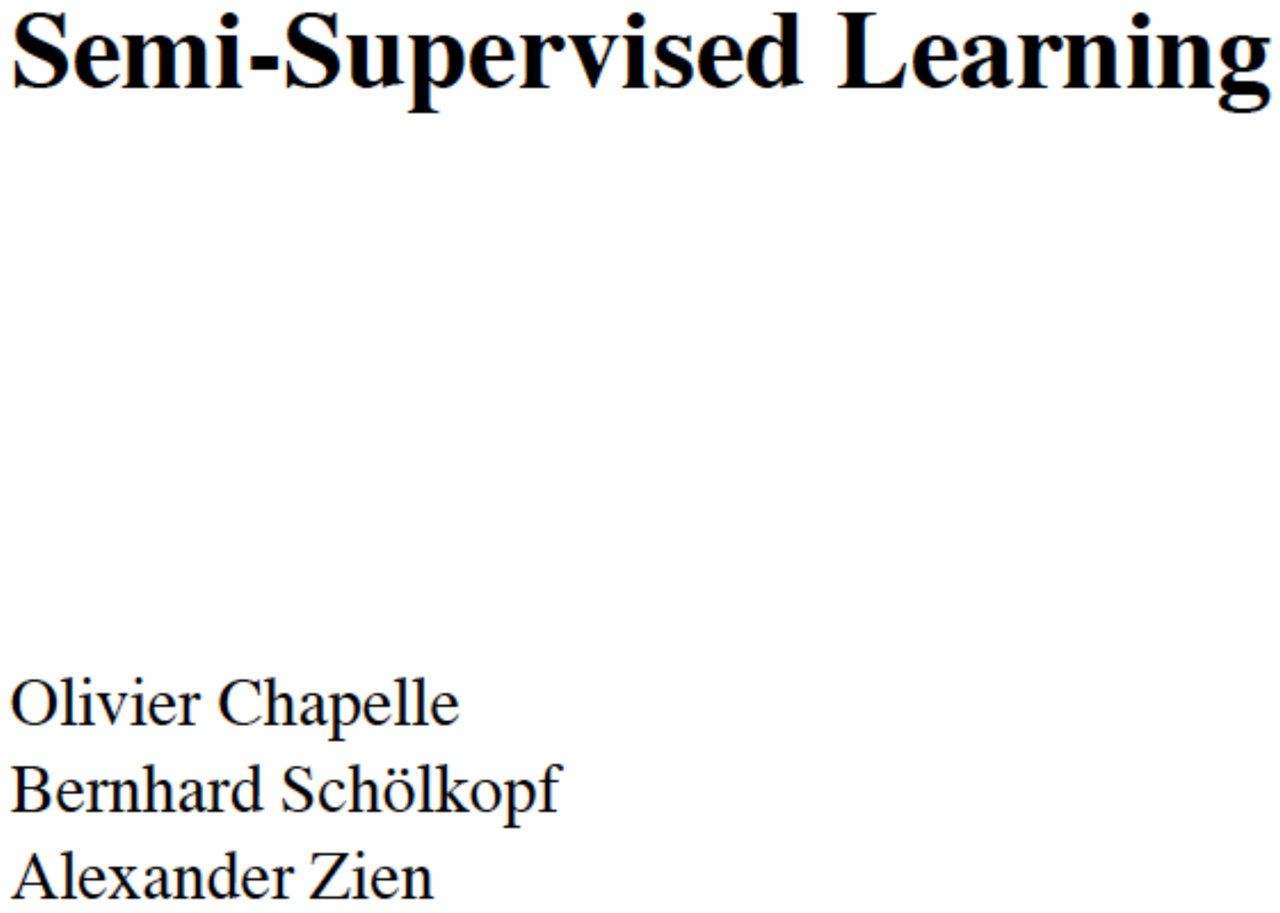
Self-learning algorithms

- Also known as self-training, self-labeling, or decision-directed learning: probably the earliest idea about using unlabeled data in

- Idea: a wrapper-algorithm that repeatedly uses a supervised learning method
  - It starts by training on the labeled data only. In each step a part of the unlabeled points is labeled according to the current decision function
  - then the supervised method is retrained using its own predictions as additional labeled points

- An unsatisfactory aspect of self-learning is that the effect of the wrapper depends on the supervised method used inside it
  - Sometimes it is unclear what the self-learning is really doing, and which assumption it corresponds to

More information



Semi-Supervised Learning

Olivier Chapelle
Bernhard Schölkopf
Alexander Zien

Acknowledgements

- Lecture materials are based on:
  - The textbook
  - Lecture materials by Stuart Russell (the co-author of the textbook) in Berkeley