Lecture 5: Linear Models for Classification (Part 3)

Recap: Linear model

- In dimension D, we can write

$$y = w^{(0)} + w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \ldots + w^{(D)}x^{(D)} = \mathbf{w}^T \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(D)} \\ 1 \end{bmatrix}$$

- In short, we can write

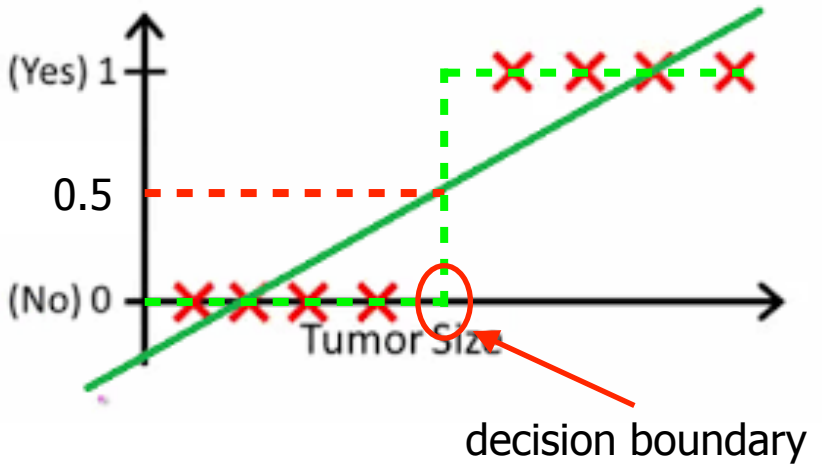
$$y = \mathbf{w}^T\mathbf{x}$$

with $\mathbf{x} \in \mathbb{R}^{D+1}$, where the extra dimension contains a 1 to account for $w^{(0)}$

Recap: Binary classification as regression

- Example (from Andrew Ng's course): Classify a tumor as benign vs malignant
  - 1D input (tumor size), 1D output (malignant vs benign)
  - Using a linear model, we can predict the label as $\hat{y} = w^{(1)}x + w^{(0)}$

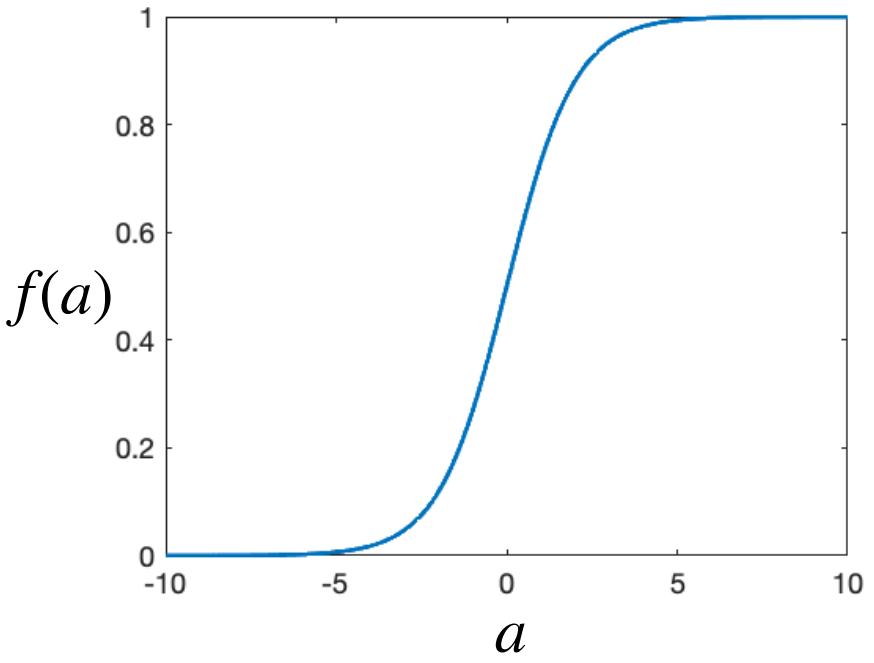
$$\text{label} = \begin{cases} 1 & \text{if } \hat{y} \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$



Recap: Adding non-linearity

- To model a classification problem, one can use a smooth approximation of the step function, such as the *logistic sigmoid function*

$$f(a) = \frac{1}{1 + \exp(-a)}$$



Recap: Logistic regression

- To better model the underlying binary classification problem, we can pass the output of the linear model through a logistic function

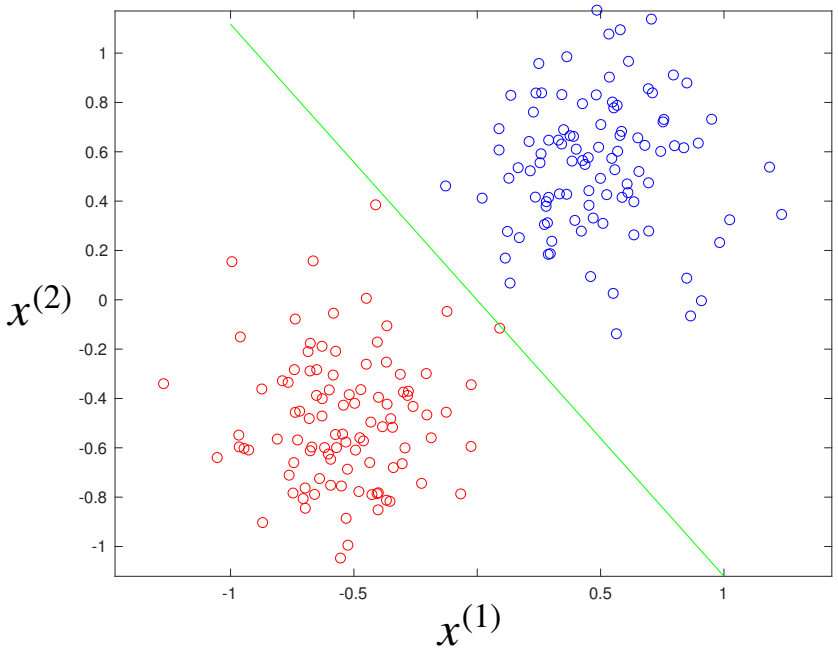
$$\hat{y}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

- Such a prediction can be interpreted as the probability that $\mathbf{x}$ belongs to the positive class (or as a score for the positive class)

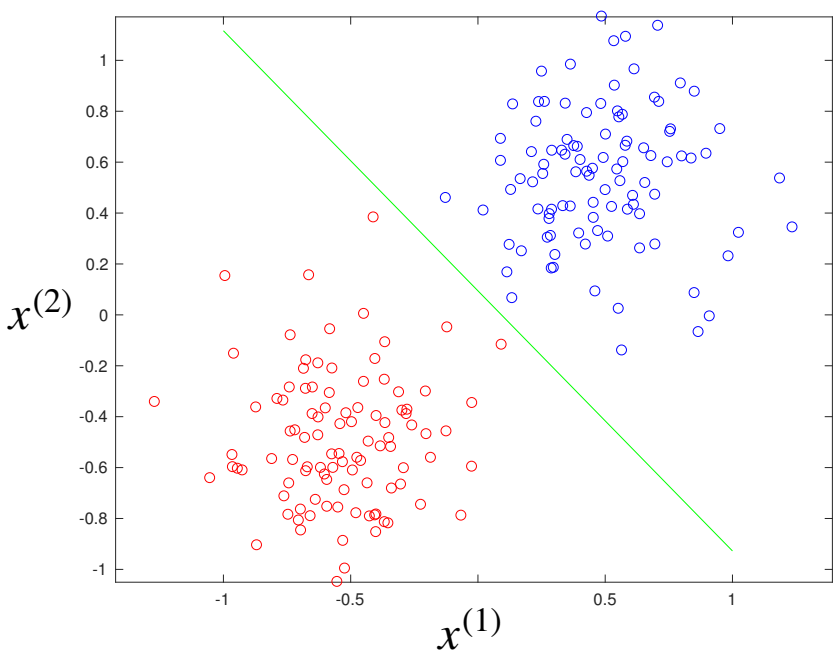
- The probability to belong to the negative class (or score for the negative class) is then computed as $1 - \hat{y}(\mathbf{x})$

Recap: Decision boundaries

- Different classifiers have different decision boundaries



Least-square classification



Logistic regression

- Can we define what a good decision boundary is?

Recap: Distance to the decision boundary

- The signed distance from a point $\mathbf{x}$ to the boundary is given by its prediction $\hat{y}(\mathbf{x})$ and the parameters $\tilde{\mathbf{w}}$ as

$$r = \frac{\hat{y}(\mathbf{x})}{\|\tilde{\mathbf{w}}\|}$$

- We can use this to find a classifier whose decision boundary is as far as possible from all the points

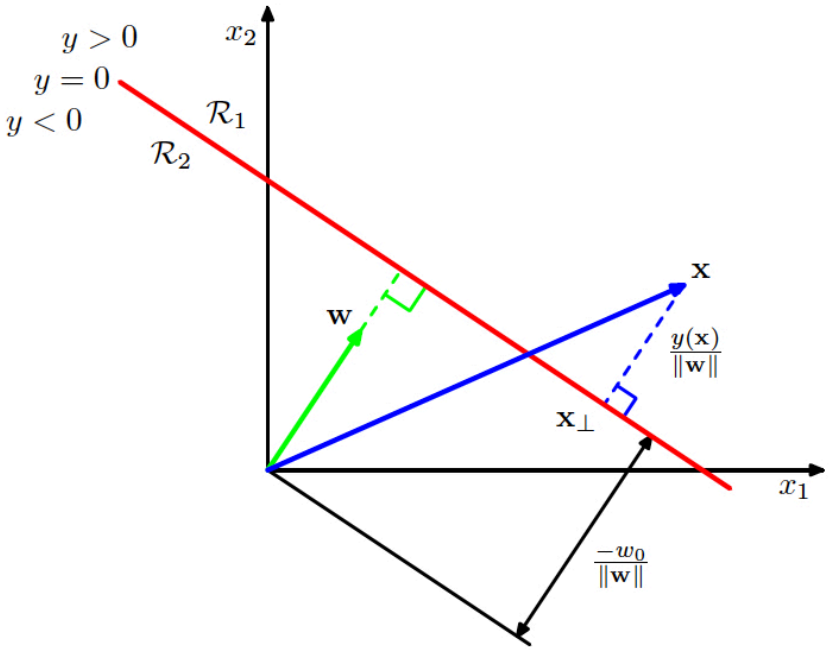


Figure from Bishop, Chapter 4.1.1

($\mathbf{w}$ in the fig. corresponds to $\tilde{\mathbf{w}}$)

Recap: Support Vector Machine

- The SVM formulation

$$\min_{\tilde{\mathbf{w}}, w^{(0)}} \frac{1}{2} \|\tilde{\mathbf{w}}\|^2$$

$$\text{subject to} \quad y_i \cdot (\tilde{\mathbf{w}}^T \mathbf{x}_i + w^{(0)}) \geq 1 \ , \quad \forall i$$

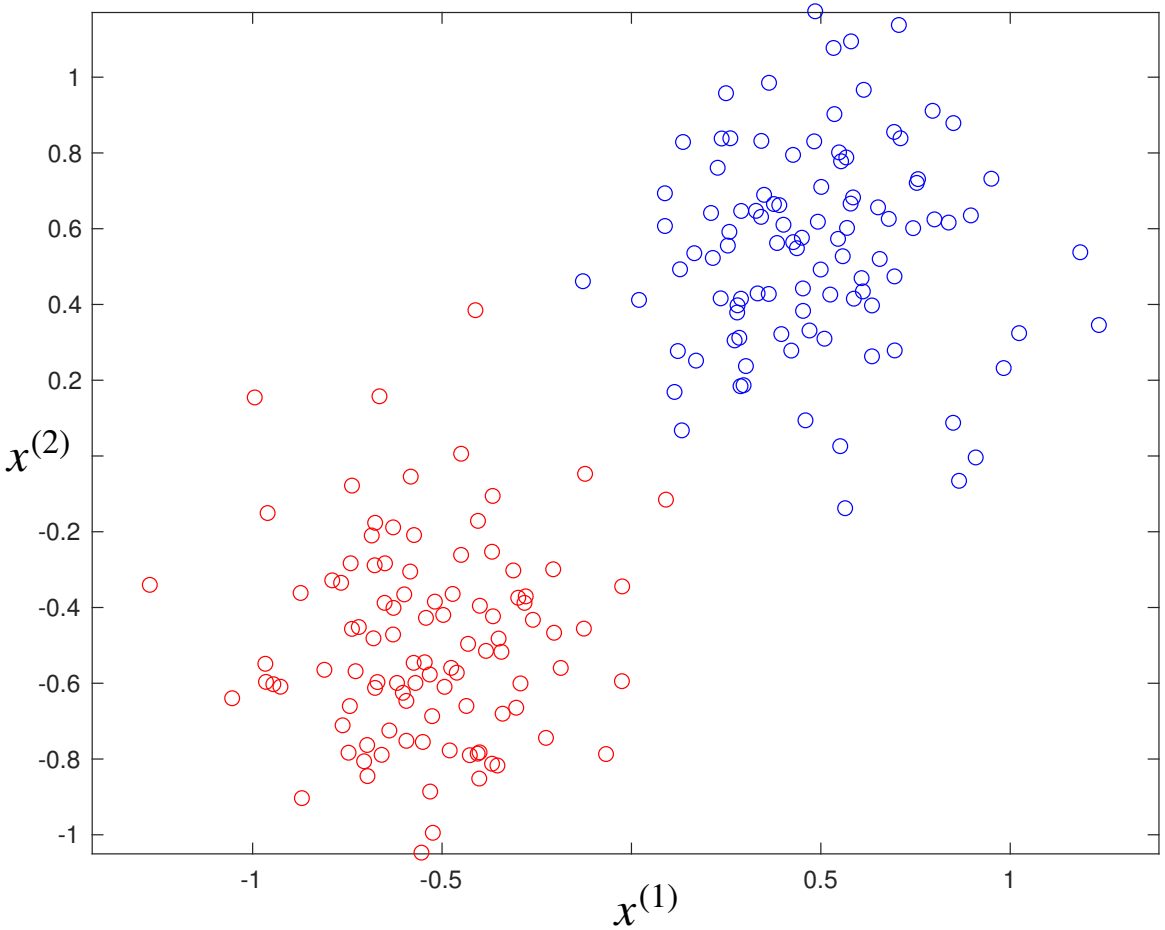
is a quadratic program

  - Quadratic objective function, with linear constraints

- Here, the problem is convex, and can be solved to optimality with standard iterative solvers

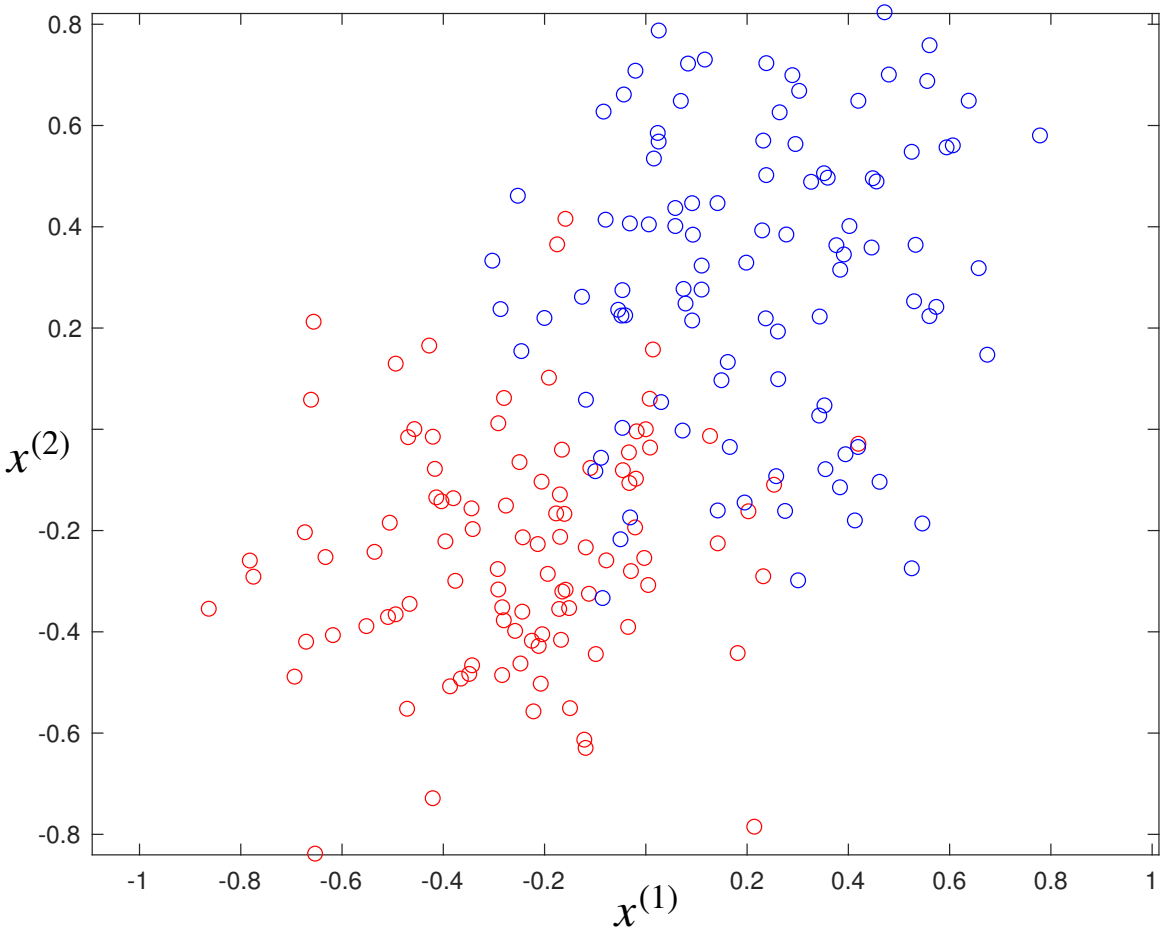
Recap: Overlapping classes

- In practice, the data essentially never looks like this



Recap: Overlapping classes

- but rather like this



Recap: Slack variables

- To handle this, we introduce an additional slack variable ξ_i for each sample
- We then write the SVM problem as

$$\min_{\tilde{\mathbf{w}}, w^{(0)}, \{\xi_i\}} \frac{1}{2}\|\tilde{\mathbf{w}}\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{subject to } \; y_i \cdot (\tilde{\mathbf{w}}^T \mathbf{x}_i + w^{(0)}) \geq 1 - \xi_i \; , \;\; \forall i$$

$$\xi_i \geq 0 \; , \;\; \forall i$$

where C sets the influence of the regularizer on the slack variables

Recap: SVM Prediction

- The SVM problem can be solved in its primal form or via its dual formulation in terms of the dual variables $\{\alpha_i\}$, $1 \leq i \leq N$
- Once the optimal solution $\{\alpha_i^*\}$ is obtained, one can use them to predict the label for a new sample $\mathbf{x}$ as

$$\hat{y}(\mathbf{x}) = (\tilde{\mathbf{w}}^*)^T \mathbf{x} + w^{(0)*} = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i^T \mathbf{x} + w^{(0)*}$$

- Because for all samples that are not support vectors, we have $\alpha_i^* = 0$, the sum can be computed on the support vectors only, i.e., with $\mathcal{S}$ the set of support vector indices,

$$\hat{y}(\mathbf{x}) = \sum_{i \in \mathcal{S}} \alpha_i^* y_i \mathbf{x}_i^T \mathbf{x} + w^{(0)*}$$

Recap: SVM: Loss function

- The SVM problem can also be written with the following loss

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

- The term

$$\max\left(0, 1 - y_i \cdot (\tilde{\mathbf{w}}^T \mathbf{x}_i + w^{(0)})\right)$$

is referred to as the *hinge loss*

Exercise: Solutions

- The slack variables allow SVM to handle the case where the classes are not truly linearly separable but still close to being so. Describe two reasons why this can happen in practice.
  1. There is some noise in the data that makes samples close to the decision boundary move to the wrong side
  2. There are some mistakes in the ground-truth annotations of the training data

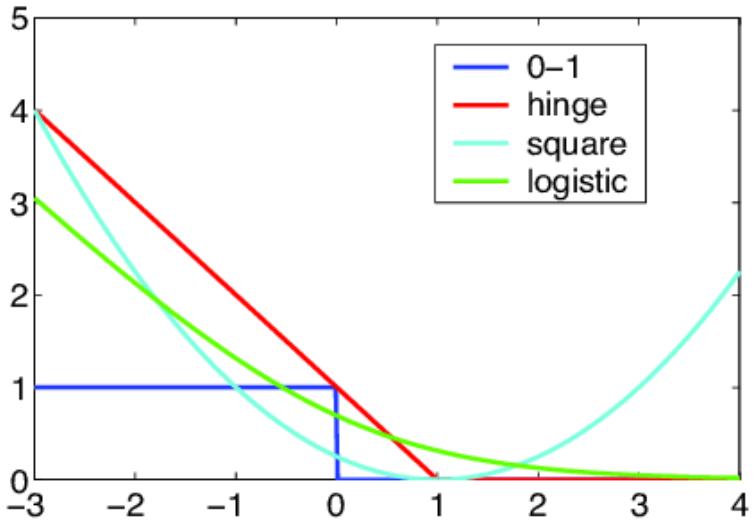
Linear models

- We have seen 3 ways to use a linear model for classification:
  - Least-square classification
  - Logistic regression
  - Support Vector Machine

- All three methods rely on the **same** linear model, yet they yield different classifiers
- Question: Why? What is the main difference between them?

Linear models

- The real difference between these methods is the loss function



- Square: Least-square classification
- Logistic: Logistic regression
- Hinge: SVM
- 0-1: Ideal

- Choosing an appropriate loss function has an impact on the resulting algorithm

Goals of today's lecture

- Move from binary to multiclass classification problems
- Understand how to represent multiple classes
- Extend the formulation of the linear classifiers seen so far to the multiclass scenario

From binary to multiclass classification

- So far, the three methods we have studied can only handle two classes: positive vs negative
- In general, one would typically like to discriminate between more than two categories. E.g., for image recognition


Label 1

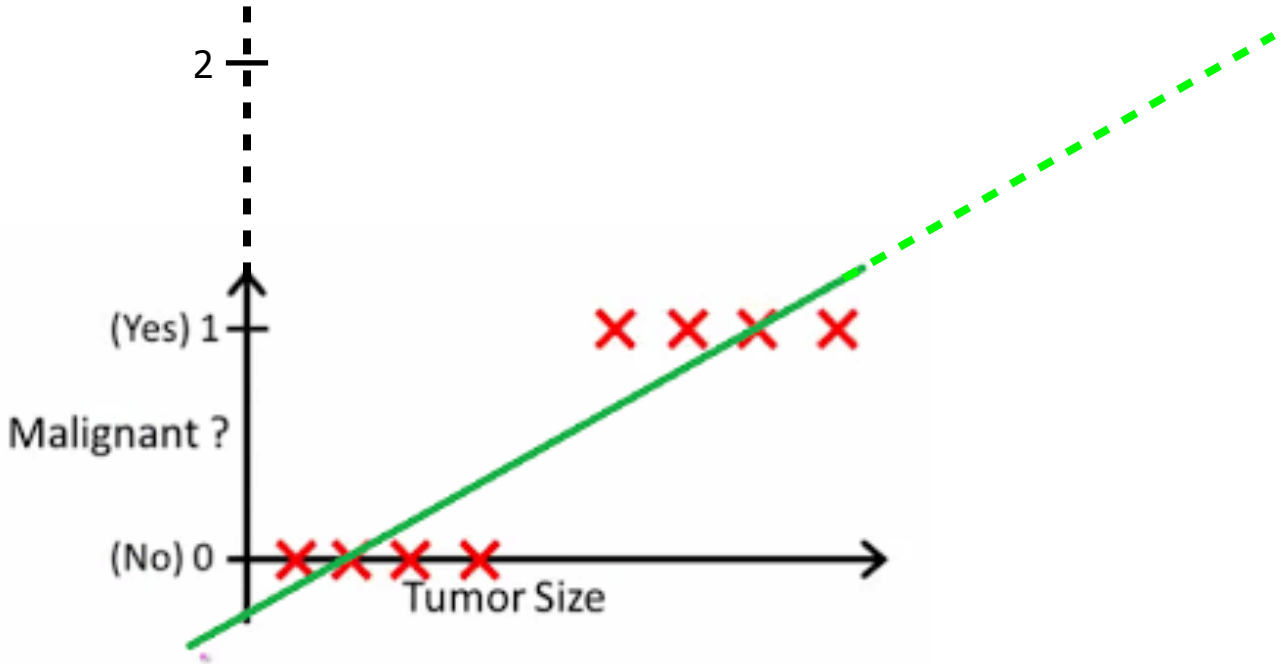

Label 2


Label 3

- Let us know see how our three linear methods can handle this scenario
  - We will start with least-square classification

Least-square classification

- In essence, least-square classification treats the problem as a regression task
  - The model predicts a continuous value $\hat{y} = w^{(1)}x + w^{(0)}$
  - There is nothing that explicitly makes this value stop at 1, so we could continue the line further



Multiclass least-square classification: Naive approach

- The simplest way to go from the binary case to handling multiple classes would then be to use one integer value for each category


Label 1


Label 2


Label 3

- And round the prediction of the model to the nearest integer to obtain the predicted class value

- The same framework as for least-square binary classification could then directly apply

Multiclass least-square classification: Naive approach

- This, however, makes no sense, because:
  - There is no natural order between the different categories
  - E.g., mistaking a car for a tree would be less strongly penalized than mistaking it for a cat
    - This is because the least-square loss would be $(3-2)^2 = 1$ in the first case and $(3-1)^2 = 4$ in the second

Label 1

Label 2

Label 3

Dealing with multiple classes

- The most common approach to modeling a multi-class problem consists of encoding the class label as a vector with a single 1 at the index corresponding to the category and 0s elsewhere
  - In a 5-class problem, a sample in class 2 is represented as

$$\mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- This is referred to as *one-hot encoding* (or 1-of-C encoding)

Multi-output linear model

- Predicting such a one-hot encoding is then similar to making the model output multiple values, as we saw for multi-output regression

- We therefore again use a matrix $\mathbf{W} \in \mathbb{R}^{(D+1)\times C}$, such that

$$\hat{\mathbf{y}}_i = \mathbf{W}^T \mathbf{x}_i = \begin{bmatrix} \mathbf{w}_{(1)}^T \\ \mathbf{w}_{(2)}^T \\ \vdots \\ \mathbf{w}_{(C)}^T \end{bmatrix} \mathbf{x}_i$$

where each $\mathbf{w}_{(j)}$ is a $(D+1)$-dimensional vector, used to predict one output dimension, i.e., the score for one class

Multi-class least-square classification

- Multi-class classification is then similar to a multi-output regression problem

- We can then write training as

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \|\mathbf{W}^T \mathbf{x}_i - \mathbf{y}_i\|^2$$

where each $\mathbf{y}_i$ is now a C-dimensional vector with a single one at the index corresponding to the class label, and zeros everywhere else

Multi-class least-square classification: Solution

- We can again group the inputs and the outputs in two matrices, as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times D+1} \qquad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1^T \\ \mathbf{y}_2^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix} \in \mathbb{R}^{N \times C}$$

- Then, the solution has the same form as in the regression case (obtained by zeroing out the gradient w.r.t. $\mathbf{W}$), i.e.,

$$\mathbf{W}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y} = \mathbf{X}^\dagger\mathbf{Y}$$

Multi-class least-square classification

- Given the optimal parameters, the prediction for a new sample is computed as

$$\hat{\mathbf{y}} = (\mathbf{W}^*)^T \mathbf{x}$$

- The sample is then assigned to class k if

$$\hat{\mathbf{y}}^{(k)} > \hat{\mathbf{y}}^{(j)} \ , \ \forall j \neq k$$

- Or in other words the final label k is obtained as

$$k = \text{argmax}_j \ \hat{\mathbf{y}}^{(j)}$$

Exercises

- You are training a multi-class least-square classifier to recognize cat, dog and car images. For the two images, $\mathbf{x}_1$, $\mathbf{x}_2$, below, what do the ground-truth vectors, $\mathbf{y}_1$, $\mathbf{y}_2$, look like numerically?
- Assuming the images are color images of size 32×32, what is the size of the parameter matrix $\mathbf{W}$?

$\mathbf{x}_1$

$\mathbf{x}_2$

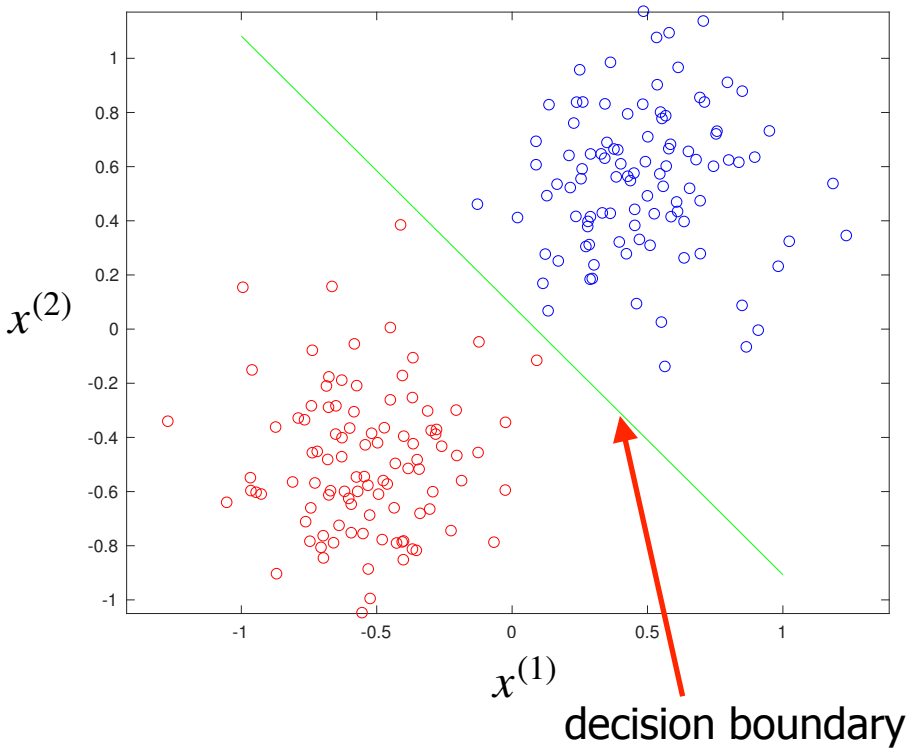
Interlude

Decision Boundary

(The following applies to all linear classification models, not just least-square classification)

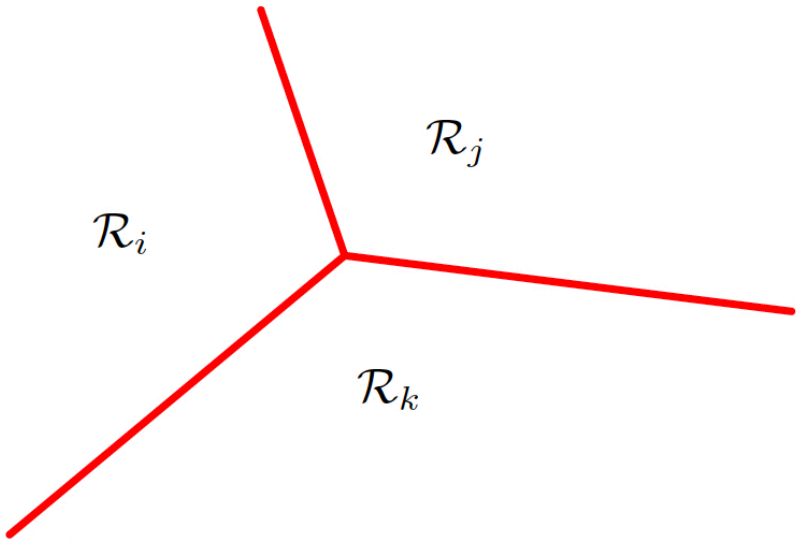
Recap: Decision boundary

- In the binary case, the point where the predicted label changes from 0 to 1 forms a *decision boundary*
  - With 2D inputs, it is a line



Decision boundaries: General definition

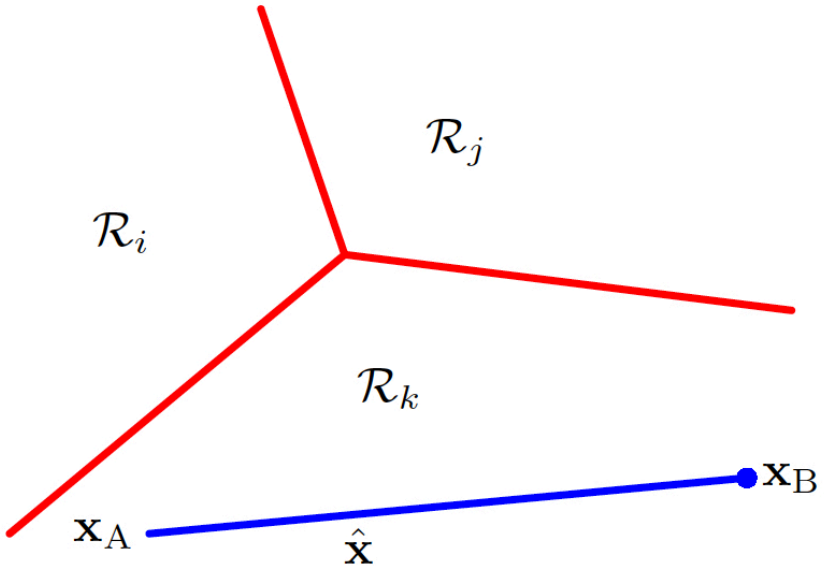
- The decision boundary between two classes k and j is defined as the set of points $\mathbf{x}$ such that $\hat{y}^{(k)}(\mathbf{x}) = \hat{y}^{(j)}(\mathbf{x})$
- This corresponds to the $(D-1)$-dimensional hyperplane



- The binary case is a special case of this definition

Decision boundaries

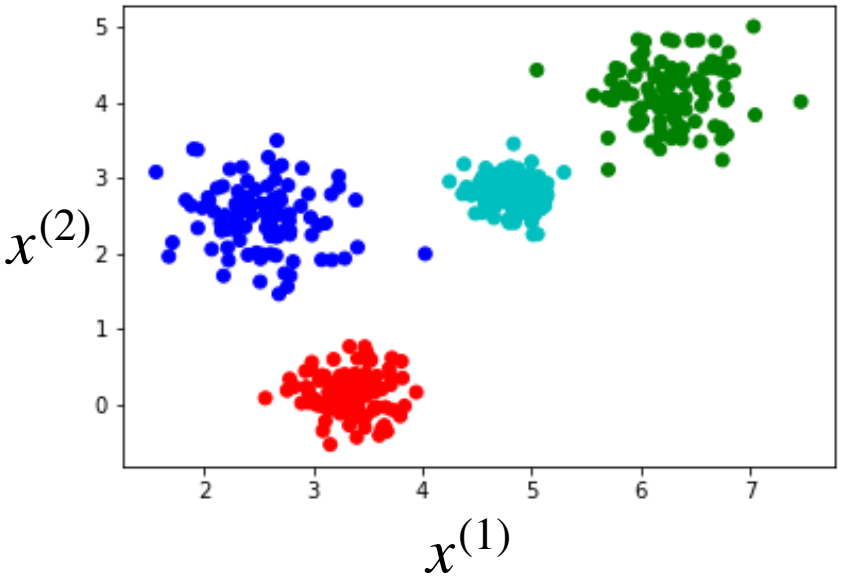
- The decision regions are singly-connected and convex

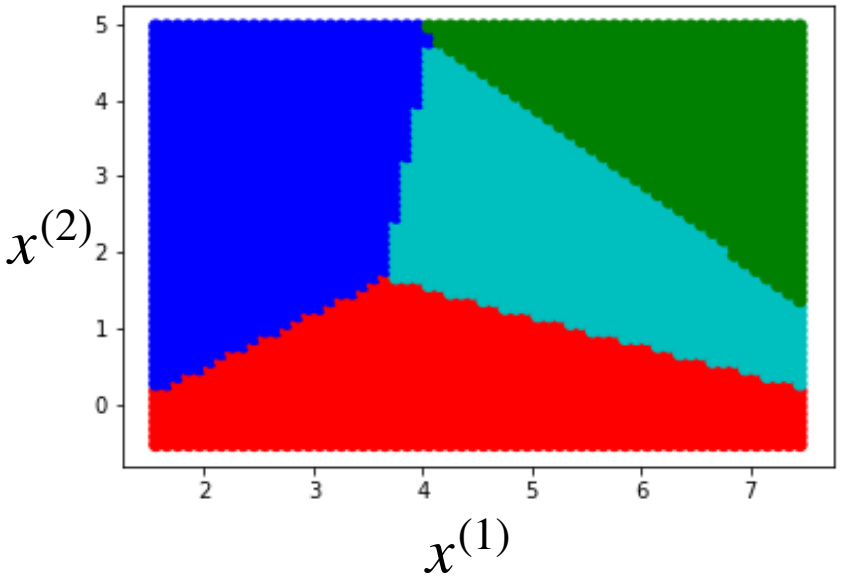


- Any point $\hat{\mathbf{x}}$ between two points $\mathbf{x}_A$ and $\mathbf{x}_B$ in the same region (e.g., k) will also be in the same region

Decision boundaries for $C > 3$ classes

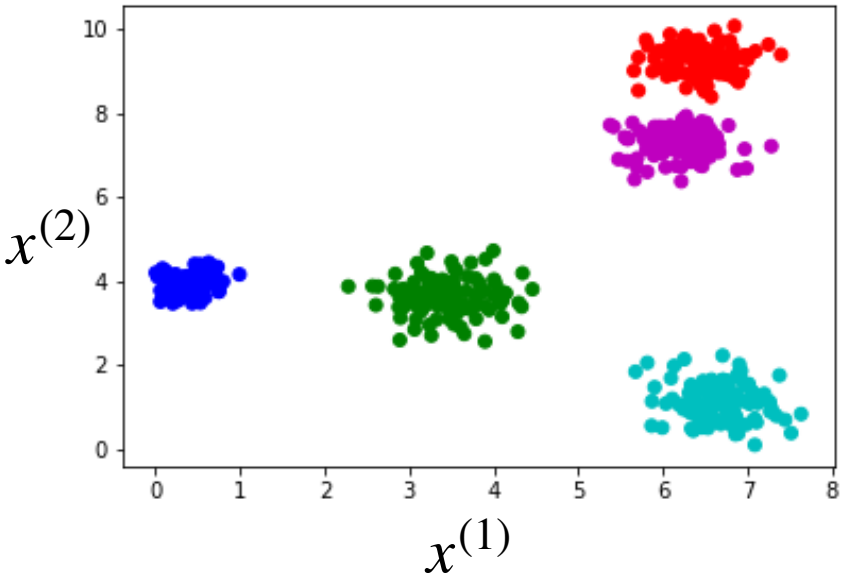
- $C = 4$ classes

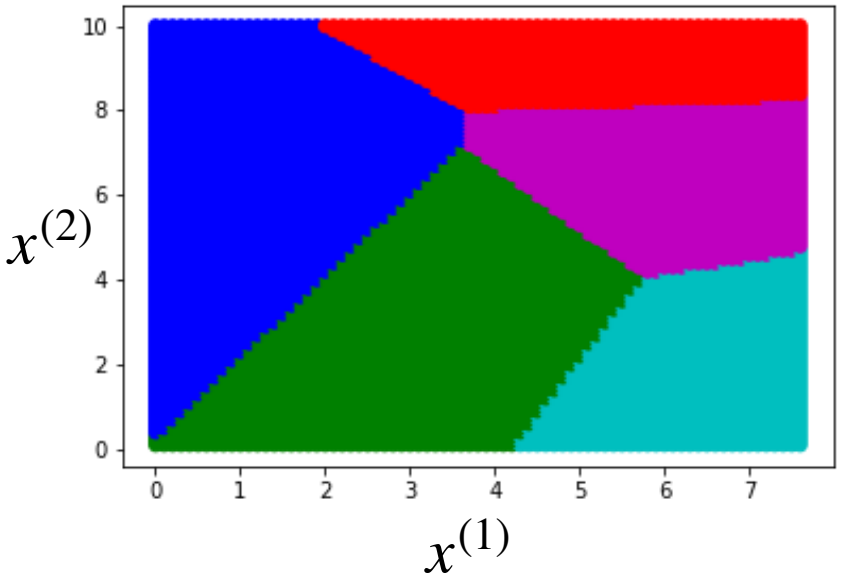




Decision boundaries for $C > 3$ classes

- $C = 5$ classes





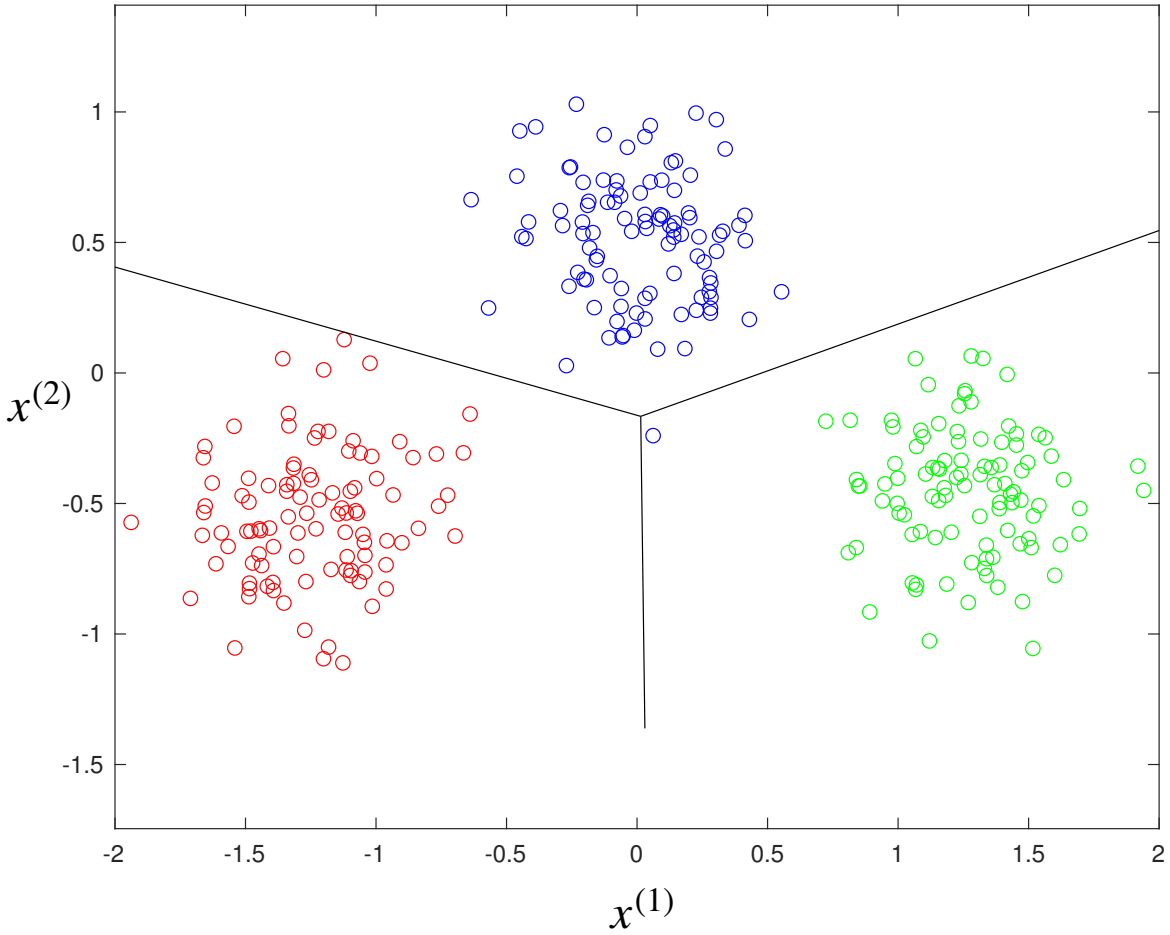
Code available on the Moodle page

End of the interlude

Let’s look at a few least-square classification examples

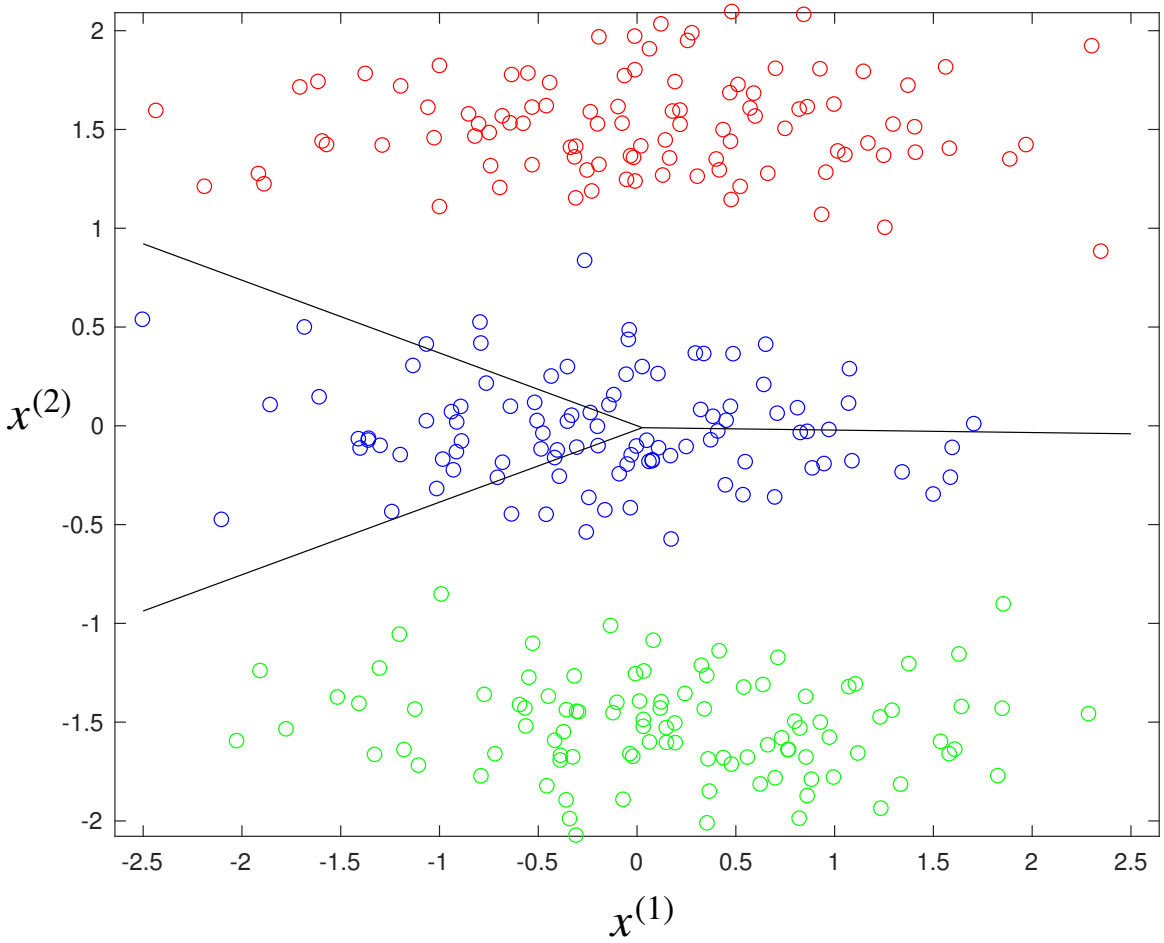
Multi-class least-square classification

- Example: 2D inputs, 3 classes



Multi-class least-square classification

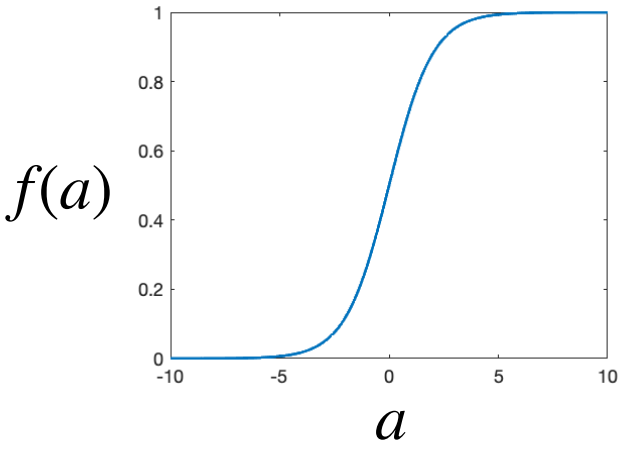
- Failure case



Multi-class least-square classification: Drawbacks

- As in the binary case, multi-class least-square classification suffers from the fact that the loss function does not reflect the underlying classification task

- In the binary case, we addressed this by adding a nonlinearity to the output of the linear model via the logistic function, leading to logistic regression



- Let us now see how this extends to the multi-class scenario

Multi-class logistic regression

- As in the least-square classification case, we can use one-hot encodings to represent multi-class labels

- Then, we again need to use a matrix $\mathbf{W} \in \mathbb{R}^{(D+1)\times C}$ to represent the model parameters

- In this case, the probability for a class k is given by the *softmax* function

$$\hat{y}^{(k)}(\mathbf{x}) = \frac{\exp(\mathbf{w}_{(k)}^T \mathbf{x})}{\sum_{j=1}^{C} \exp(\mathbf{w}_{(j)}^T \mathbf{x})}$$

Multi-class logistic regression

- The empirical risk can then be derived from the multi-class version of the cross entropy, which yields

$$R(\mathbf{W}) = -\sum_{i=1}^{N}\sum_{k=1}^{C} y_i^{(k)} \ln \hat{y}_i^{(k)}$$

- As in the binary case, a single $y_i^{(k)}$ is 1 for every sample i, so we really have one term per training sample

- As in the binary case, training is done by minimizing this loss using gradient descent

Recap: Gradient descent

- Algorithm:
  1. Initialize $\mathbf{w}_0$ (e.g., randomly)
  2. While not converged
     2.1. Update $\mathbf{w}_k \leftarrow \mathbf{w}_{k-1} - \eta \nabla R(\mathbf{w}_{k-1})$

- η is a single value referred to as the learning rate
- Convergence can again be measured by
  - Thresholding the change in function value
  - Thresholding the change in parameters, e.g., $\|\mathbf{w}_{k-1} - \mathbf{w}_k\| < \delta_w$

Gradient descent: Dealing with a matrix

- With multiple classes, the parameters are shaped as a matrix $\mathbf{W} \in \mathbb{R}^{(D+1)\times C}$, not as a vector anymore
- This does not affect the algorithm:
  - It just means that the gradient itself will also be a matrix, i.e.

$$\nabla R(\mathbf{W}_{k-1}) \in \mathbb{R}^{(D+1)\times C}$$

- So the whole algorithm works with matrix entities
  - Convergence check can be done based on the Frobenius norm: $\|\mathbf{W}_{k-1} - \mathbf{W}_k\|_F < \delta_w$

Multi-class logistic regression: Gradient

- By following the chain rule and using the gradient of the softmax function, we can derive the gradient of the multi-class cross-entropy as (details in Bishop Chap. 4.3.4)

$$\nabla R(\mathbf{W}) = \sum_{i=1}^{N} \mathbf{x}_i \left(\hat{\mathbf{y}}_i - \mathbf{y}_i\right)^T$$

- The gradient follows the same intuition as in the binary case: It multiplies the input by the error of the current prediction for each class $\left(\hat{y}_i^{(k)} - y_i^{(k)}\right)$

Multi-class logistic regression: Prediction

- At test time, given a new input sample $\mathbf{x}$, the probability for any class k is computed via the softmax function as

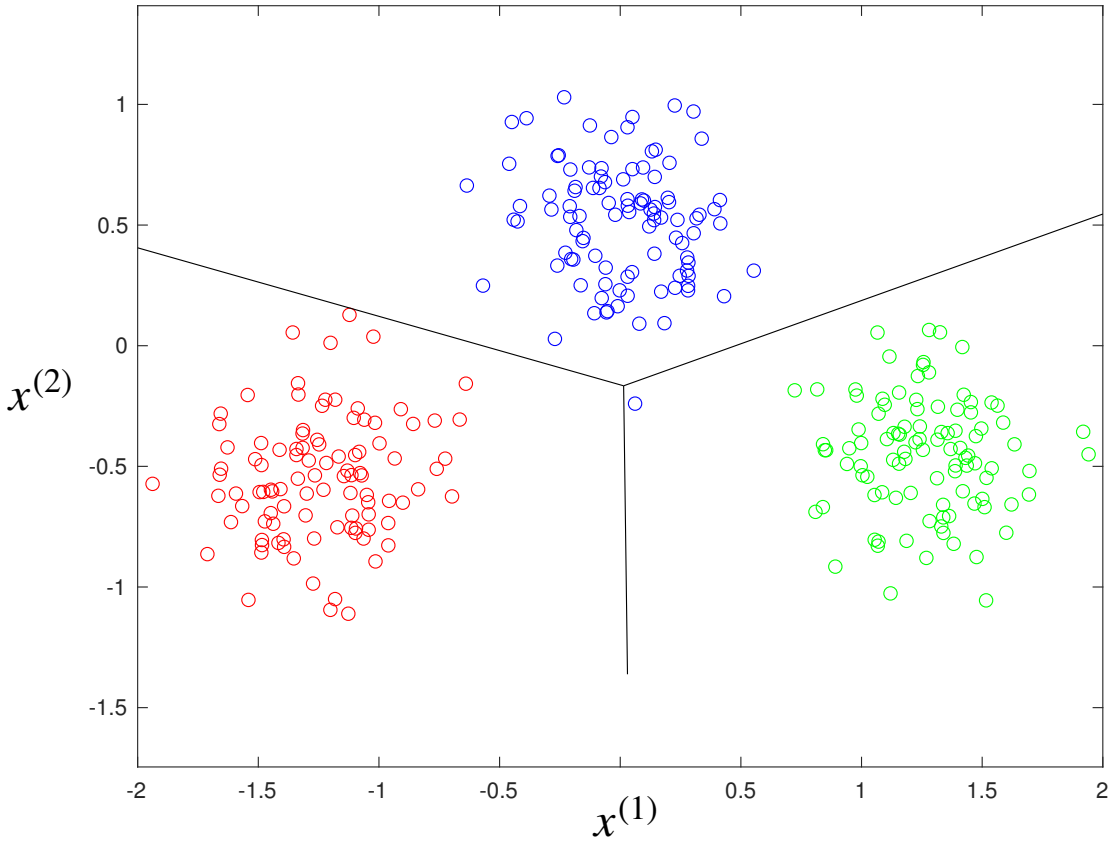
$$\hat{y}^{(k)}(\mathbf{x}) = \frac{\exp((\mathbf{w}^*_{(k)})^T \mathbf{x})}{\sum_{j=1}^{C} \exp((\mathbf{w}^*_{(j)})^T \mathbf{x})}$$

- The final class label is then predicted as

$$k = \text{argmax}_j \; \hat{\mathbf{y}}^{(j)}(\mathbf{x})$$

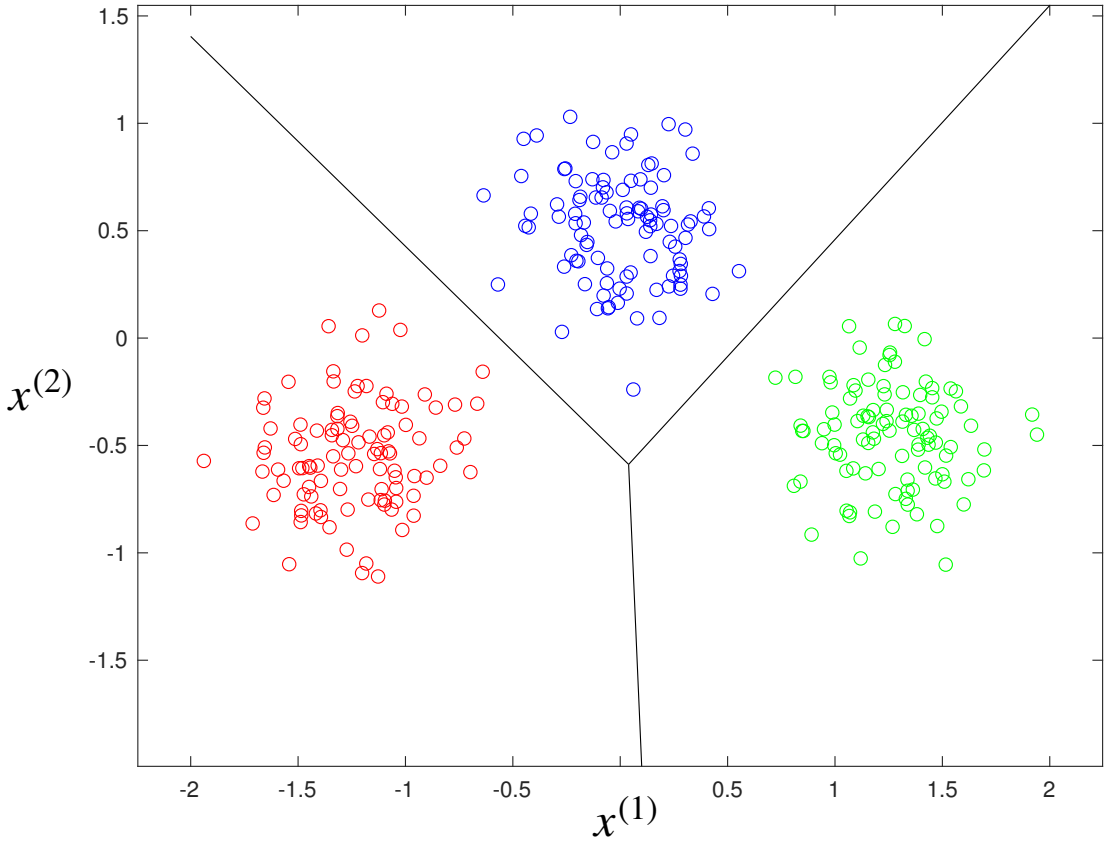
Multi-class logistic regression: Example

- 2D inputs, 3 classes: Least-square classification results



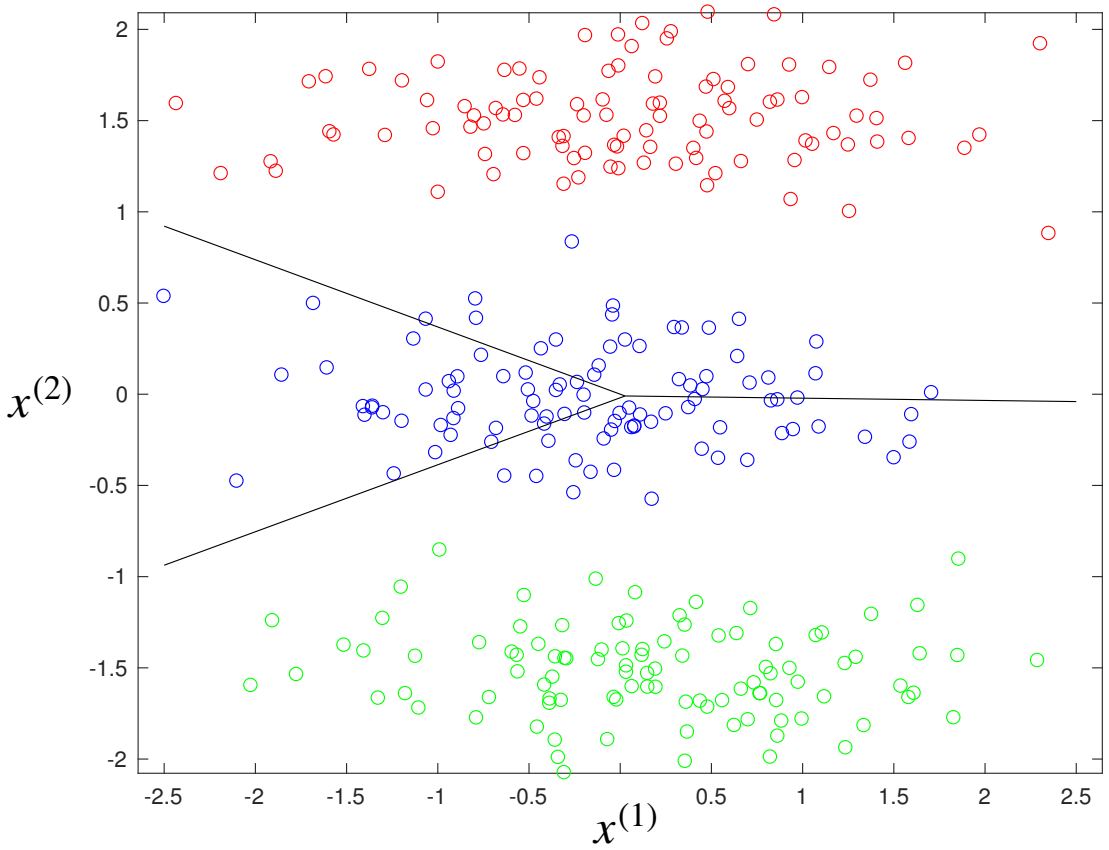
Multi-class logistic regression: Example

- 2D inputs, 3 classes: Logistic regression results



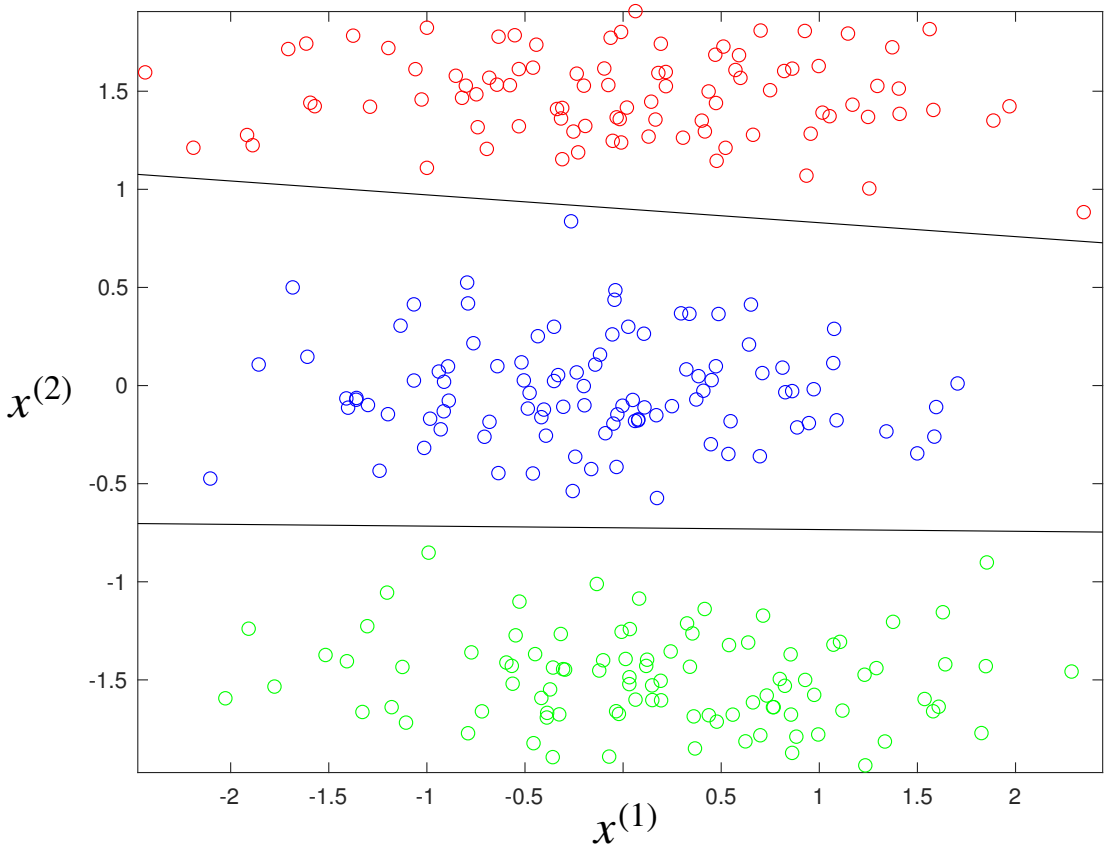
Multi-class logistic regression: Example

- 2D inputs, 3 classes: Least-square classification results



Multi-class logistic regression

- 2D inputs, 3 classes: Logistic regression results



Interlude

Evaluation metrics for multi-class classification

Recap: Classification evaluation

- Confusion matrix: Binary case (e.g., spam vs non-spam email)

True class → / Hypothesized class ↓	Pos	Neg
Yes	TP	FP
No	FN	TN
	P=TP+FN	N=FP+TN

TP: True positive (number of samples correctly classified as positive)
FP: False positive (number of samples incorrectly classified as positive)

TN: True negative (number of samples correctly classified as negative)
FN: False negative (number of samples incorrectly classified as negative)

P: Total number of positive samples
N: Total number of negative samples

Figure from N. Japkowicz's ICMLA tutorial

Recap: Classification metrics

- In the binary case, we saw the following metrics that can be computed from the confusion matrix:
  - Accuracy: Percentage of correctly classified samples
  - Precision: Percentage of samples classified as positives that are truly positives
  - Recall: Percentage of positives samples that are correctly classified as positives
  - False positive rate: Percentage of negative samples that are incorrectly classified as positives
  - F1 score: Single number that combines precision and recall

Classification evaluation

- Confusion matrix: Multi-class case
  - Numerical example for a dataset with 3 classes (UCI Iris dataset)
  - Note that the matrix is transposed (predicted labels as columns and actual labels as rows), which can also be used in practice

		Predicted		
		Iris-setosa	*Iris-versicolor*	*Iris-virginica*
Actual	*Iris-setosa*	12	1	1
	Iris-versicolor	0	16	0
	Iris-virginica	0	1	16

Figure from Z. Waks's slides

Metrics for multi class problems

- Accuracy:
  - A global accuracy can be obtained by summing the correct predictions for all classes (diagonal values of the confusion matrix), and dividing this by the total number of samples (sum of all entries in the confusion matrix)
- Precision, recall, FP-rate and F1-score:
  - The simplest approach to handling these metrics is to compute them for each individual class, and then average over the classes
  - In this case, each individual class is in turn considered as the “positive” class while the others act as the “negative” class
  - This does not account for class imbalance (some classes may have many more samples than others; more about this in a future lecture)
  - This can be handled by weighting the metric for each class by the proportion of samples from that class

End of the interlude

Let’s look at a practical multi-class logistic regression example

Multi-class logistic regression: Application

- Predict the state of a fetus from cardiotocography: UCI Cardiotocography Dataset
  - 21 input features
  - Predict fetal state as Normal vs Suspect vs Pathologic

S. number	Name of the features	Description
1	LB	FHR baseline (beats per minute)
2	AC	Number of accelerations per second
3	FM	Number of fetal movements per second
4	UC	Number of uterine contractions per second
5	DL	Number of light decelerations per second
6	DS	Number of severe decelerations per second
7	DP	Number of prolonged decelerations per second
8	ASTV	Percentage of time with abnormal short term variability
9	MSTV	Mean value of short term variability
10	ALTV	Percentage of time with abnormal long term variability
11	MLTV	Mean value of long term variability
12	Width	Width of FHR histogram
13	Min	Minimum of FHR histogram
14	Max	Maximum of FHR histogram
15	Nmax	Number of histogram peaks
16	Nzeros	Number of histogram zeros
17	Mode	Histogram mode
18	Mean	Histogram mean
19	Median	Histogram median
20	Variance	Histogram variance
21	Tendency	Histogram tendency: −1 = left asymmetric; 0 = symmetric; 1 = right asymmetric

Multi-class logistic regression: Application

- Predict the state of a fetus from cardiotocography: UCI Cardiotocography Dataset
  - Logistic regression yields an accuracy of 92.1%
  - Confusion matrix

Predicted label

True label			
	404	2	4
	24	48	0
	2	10	38

  - Still 26 suspicious/pathologic samples classified as normal!

Dealing with multiple classes

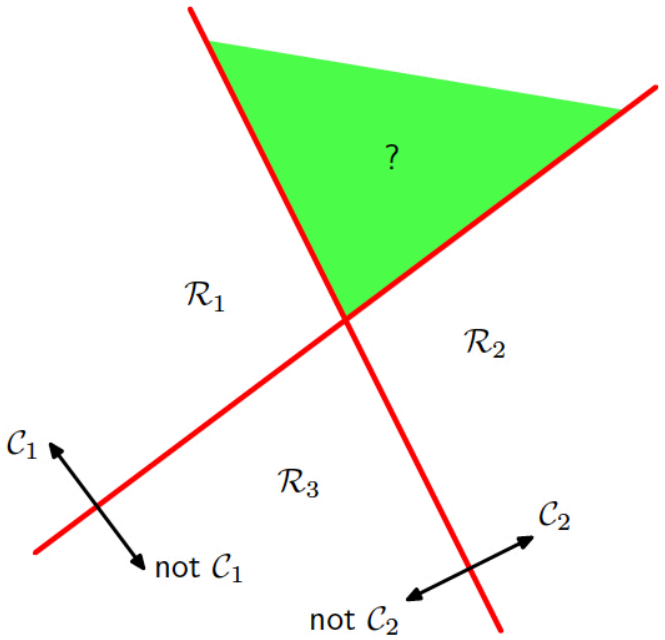
- So far, we have seen how to extend binary least-square classification and binary logistic regression to the multi-class scenario. How about SVM?

- Unfortunately, there is no ideal solution to handle multiple classes in SVM

- Instead, one typically relies on one of the two solutions described in the next slides
  - Note that these solutions are quite general, not specific to SVM

Dealing with multiple classes

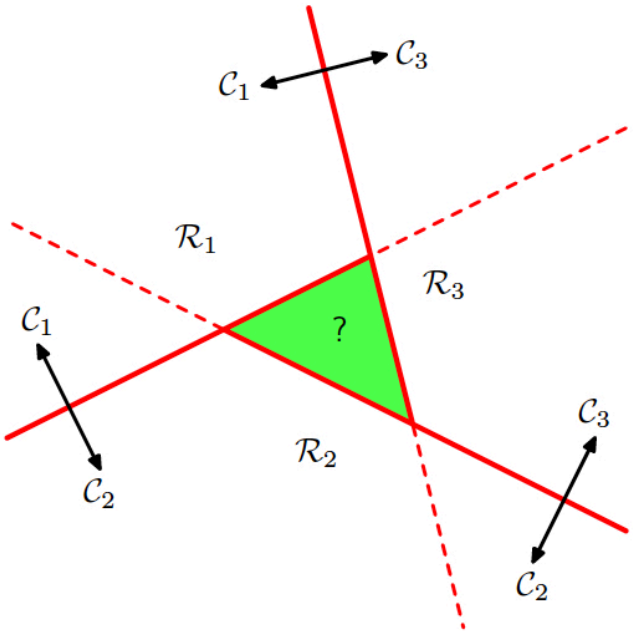
- To deal with multiple classes, one can use several binary classifiers
  - E.g., one-vs-rest: Classify each individual category vs all the other ones jointly



  - This, however, leaves ambiguities (e.g., a sample can be classified as being from both class 1 and class 2)

Dealing with multiple classes

- To deal with multiple classes, one can use several binary classifiers
  - E.g., one-vs-one: Train one classifier for every pair of classes



  - This also may result in inconsistencies (e.g., a sample can be classified as being from class 1, class 2 and class 3)

Multi-class SVM: Application

- Thyroid disease classification:
  - 30 attributes
  - 3 classes:
    - Hyperthyroid (excessive hormone production)
    - Hypothyroid (insufficient hormone production)
    - Normal

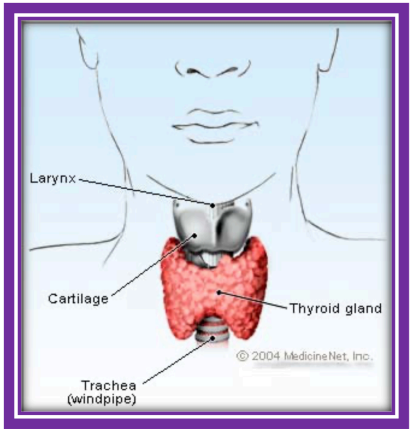


Image from Geetha et al., Global Journal of Computer Science and Technology , 2016

Multi-class SVM: Application

- Comparison of one-against-all (OAASVM) with one-against-one (OAOSVM)
  - Results from Chamasemani & Singh, International Conference on Bio-Inspired Computing: Theories and Applications, 2011

Method	Accuracy		Computation Time	
	Training	Test	Training	Test
OAASVM	95.6%	95.3%	212.4s	227.8s
OAOSVM	96.1%	94.48%	79.3s	81s

Multi-class SVM: Application

- Evaluation of the influence of the hyper-parameter C (strength of the regularizer on the slack variables)
  - Results from Chamasemani & Singh, International Conference on Bio-Inspired Computing: Theories and Applications, 2011

C	Accuracy		Computation Time	
	Training	Test	Training	Test
10	93.8%	92.5%	58.8s	52.9s
40	94.1%	93.8%	92.2s	93.3s
60	**97.2%**	**96.4%**	**40.74s**	**40.21s**
80	93.4%	92.9%	123.9s	134.6s

Comparing linear classifiers

- We have now seen 3 different linear (multi-class) classifiers:
  - Least-square classification
  - Logistic regression
  - SVM
- In practice, one almost never uses the least-square one
  - It does not reflect the underlying classification task, but is useful from a pedagogical standpoint
- Let us now compare logistic regression and SVM
  - The following results come from https://martin-thoma.com/comparing-classifiers/

Comparison on the Iris dataset

- UCI Iris dataset
  - 150 samples
  - 4 attributes (petal and sepal length and width)
  - 3 iris species (i.e., 3 classes)

	Accuracy	Training time	Testing time
SVM, linear	88.00%	0.0006s	0.0002s
Logistic Regression (C=1)	88.00%	0.0011s	**0.0001s**
Logistic Regression (C=1000)	92.00%	0.0010s	0.0002s

The value C is a hyper-parameter encoding "regularization" strength. We will talk about this in a future lecture; let's ignore it for now

- These results may suggest that logistic regression is better, however…

Comparison on the MNIST dataset

- MNIST handwritten digit recognition dataset
  - 60'000 training samples, 10'000 test samples
  - 28×28 images
  - 10 digits: 0,…,9 (i.e., 10 classes)

	Accuracy	Training time	Testing time
Linear SVM	94.16%	168.6950s	158.0101s
Logistic Regression (C=1)	91.47%	272.1309s	0.0531s
Logistic Regression (C=10000)	91.23%	1807.0624s	0.0529s

The value C is a hyper-parameter encoding "regularization" strength. We will talk about this in a future lecture; let's ignore it for now

- The choice of the best classifier depends on the data

SVM confusion matrix on MNIST

- The errors are reasonably evenly spread across the classes
  - If it were not the case, one could try to re-weight the samples in the empirical risk to focus more strongly on the weaker classes

Linear SVM ¶

```
Classifier: linear SVM
Training time: 168.6950s
Testing time: 158.0101s
Confusion matrix:
[[2226    0    9    2    6   12    8    3   11    1]
 [   1 2537   18    3    3    1    1    7   17    0]
 [  12   16 2158   25   24    6   27   19   25    2]
 [   3    7   46 2188    4   47    3   18   27    5]
 [   2    5   19    1 2117    1    8    6    3   49]
 [  18   13   11   73   20 1872   31    0   26    5]
 [  20    6   22    1   10   30 2179    0    3    0]
 [   5   10   32   11   30    5    0 2268    5   51]
 [  11   39   26   47   10   40    7    7 2018   10]
 [  11    9    9   24   64    8    0   61   14 2189]]
Accuracy: 0.9416
```

Exercise

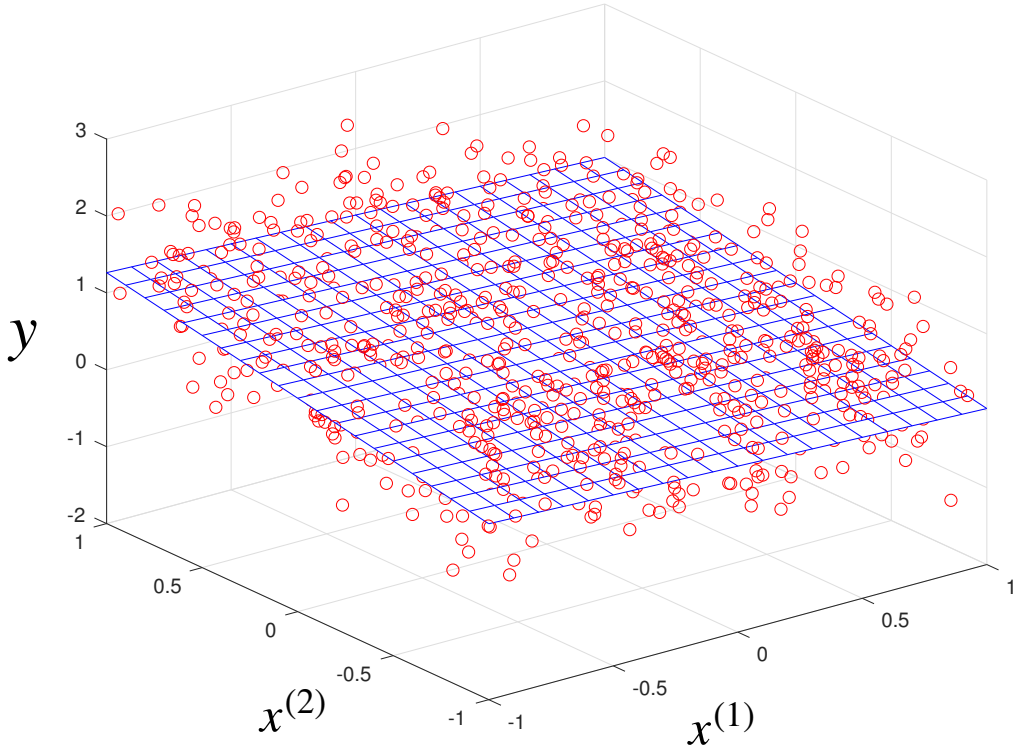
- You have trained a multi-class logistic regression classifier to recognize cat, dog and car images. For the two test images, $\mathbf{x}_1$, $\mathbf{x}_2$, below, assuming that the model predicts the correct class, what would you expect the outputs of the model, $\hat{\mathbf{y}}_1$, $\hat{\mathbf{y}}_2$, to look like numerically? Explain why.

$\mathbf{x}_1$

$\mathbf{x}_2$

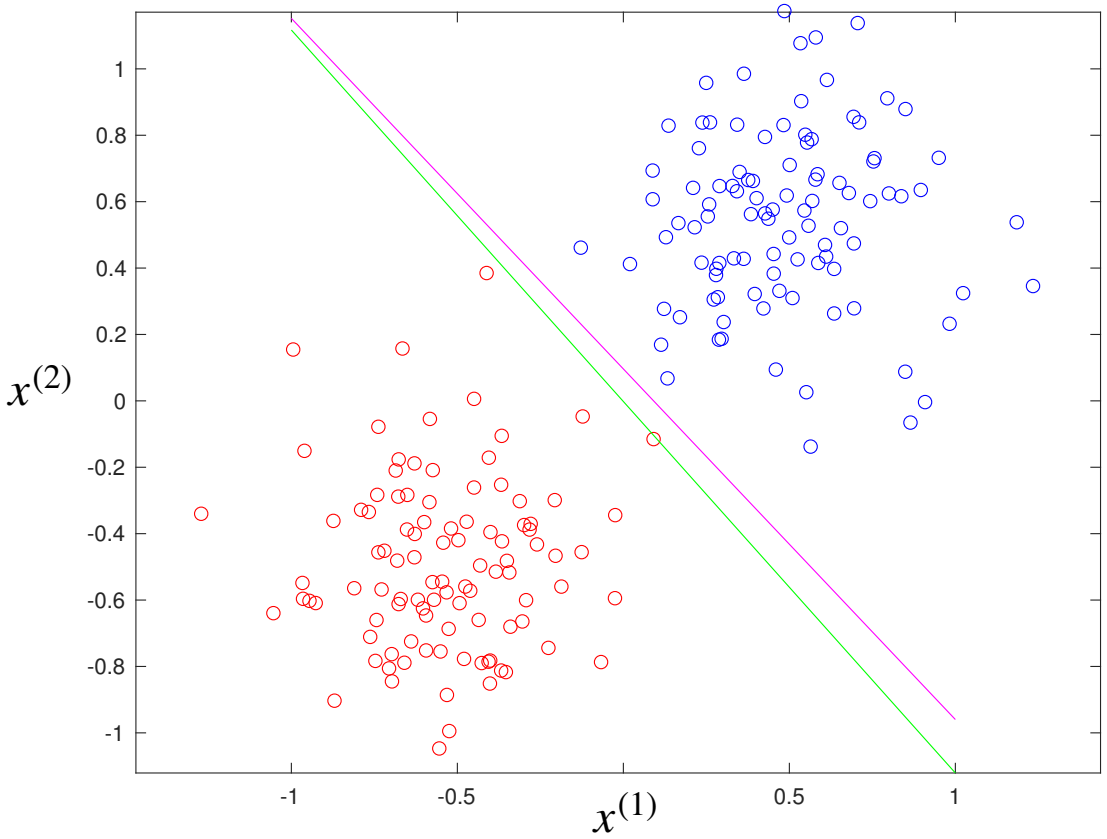
Until now

- Linear regression: Fit a linear model to the observations



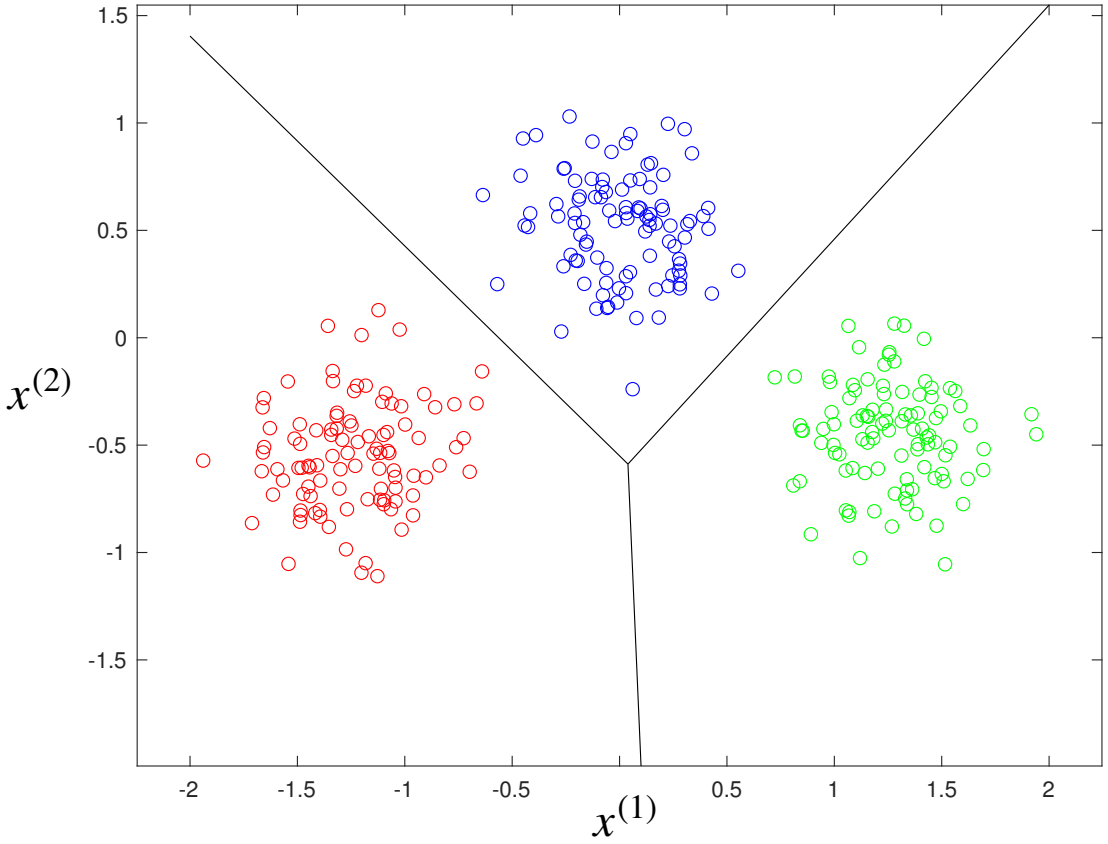
Until now

- Linear classification: Separate the classes by a linear model



Until now

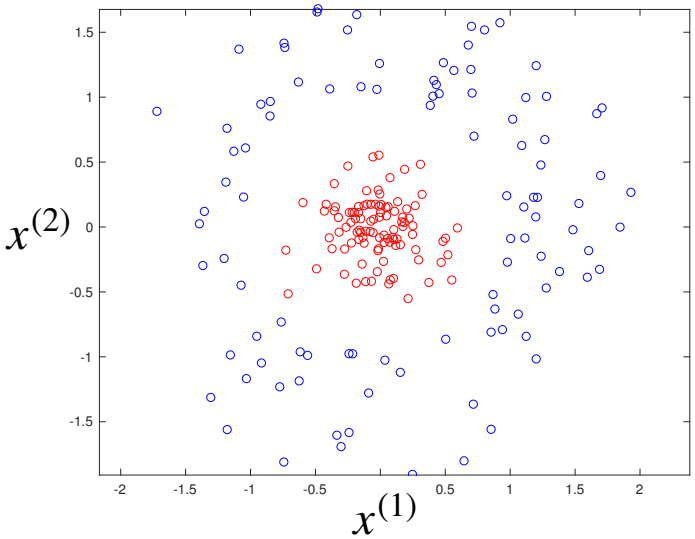
- Linear classification: Separate the classes by a linear model (multiple lines for the multi-class scenario)



From linear to nonlinear

- Unfortunately, no linear model can directly fit this data
  - 1D input, 2 classes (colors)

- Or this data
  - 2D input, 2 classes (colors)



From linear to nonlinear

- Or this even more complicated data