Machine Learning
Week 6 Lecture: - Classification and Evaluation

Debasis Ganguly
Debasis.Ganguly@glasgow.ac.uk

School of Computing Science
University of Glasgow

November 7, 2024

ML course so far

- Supervised learning
  - Regression
    - Minimised loss (least squares)
    - Maximised likelihood
    - Bayesian approach
  - **Classification** (This and the next 3 weeks)
    - Logistic Regression, Evaluation (Week 6)
    - Softmax Regression, Naive Bayes, K-NN (Week 7)
    - Bayesian approaches for Classification (Week 8)
    - Support Vector Machines (Week 9)
- **Unsupervised learning**
  - Clustering and Dimensionality Reduction (Week 10)

Classification vs. Regression

Similarity with regression

- A data instance $x \mapsto \mathbf{x}$ (embedding function or feature vector).
- Example:
  - $\mathbf{x}$ can be a set of features, e.g., $(backyard\_area, postcode, \#bedrooms, \ldots)$.
  - Bag of words embedding of text indicating if a word is present or not
  - e.g., encode: document "the cat sat on the mat" $\mapsto$ Boolean vector with a high dimension (say $10,000$ comprised of common English words)
    - the components corresponding to words 'the', 'cat' etc. are 1.

Classification vs. Regression

Differences with regression

- Given an **x** the **task is to predict** a y, where $y \in \{0, 1, \ldots, k-1\}$ – a **categorical variable** of k possible values (also abbv. as $y \in \mathbb{Z}_k$).
  - For a given house with values of backyard_area, postcode etc., one may predict the house-price range as $\{low, medium, high\}$ ($k = 3$).
- Recall for regression: $y \in \mathbb{R}$.

Regression vs. Classification

Predicting continuous values by regression

- **Input**: a **training set** of labelled data: $\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)})\}$, where each $\mathbf{x}^{(i)} \in \mathbb{R}^d$ (d dimensional feature vectors or embeddings) and each $y^{(i)} \in \mathbb{R}$ (a continuous value).
- **Output**: A model vector $h_\theta(\mathbf{x}) = \theta^T \mathbf{x}$ or (in this course, we will interchangeably use both θ and $\mathbf{w}$ to denote parameter vectors).
- **Predictions**: $\hat{y}(\mathbf{x}_{new}) = h_\theta(\mathbf{x}_{new}) = \theta^T \mathbf{x}_{new}$

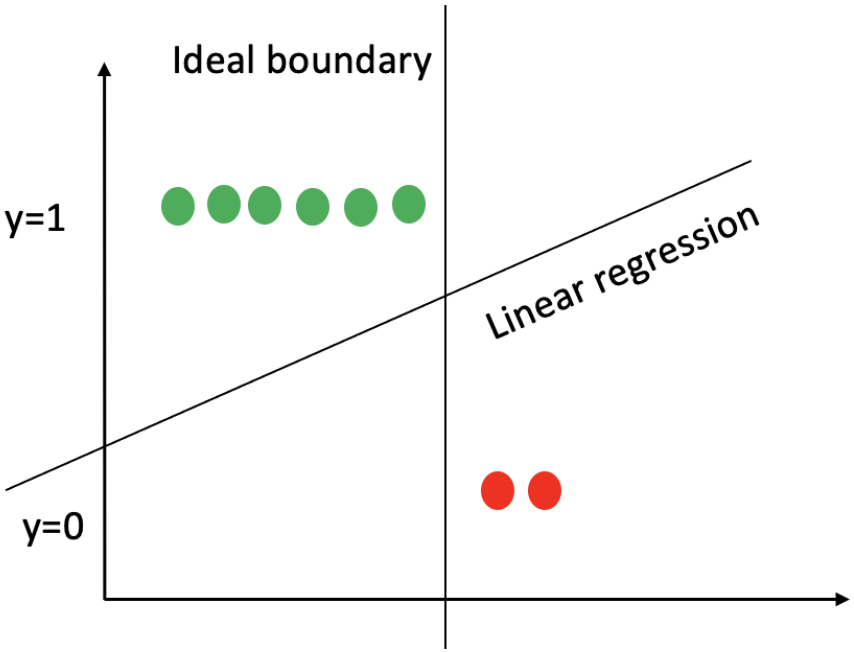
What needs to change?

- Obviously, the output $\hat{y}$ is now **constrained** (for now, let's say between 0 and 1).
- So, $\theta^T \mathbf{x}$ **doesn't work**. Why?
  - Because $\theta^T \mathbf{x}$ is a line in the data space ($\notin \{0, 1\}$).

Regression vs. Classification

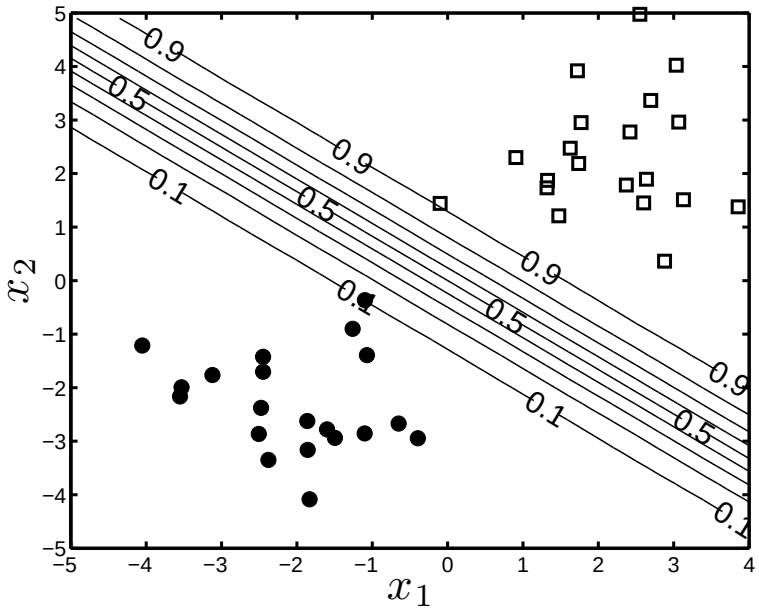
Does the naive solution work?

- Wait! But can't we just fit a line by **minimizing least squares** and then treat that line as a threshold (decision boundary)?
  - $\arg\min_\theta J(\theta) = (y - \theta^T \mathbf{x})^2$?
  - $\hat{y} = \mathbb{I}(\theta^T \mathbf{x})$? — $\mathbb{I}(z) = 1$ if $z > 0$ or 0 otherwise.



- A closely fitting line isn't a good decision boundary!
- Exercise: Try to find another arrangement of the points that also doesn't work.

What can we do about it (Intuitively)?

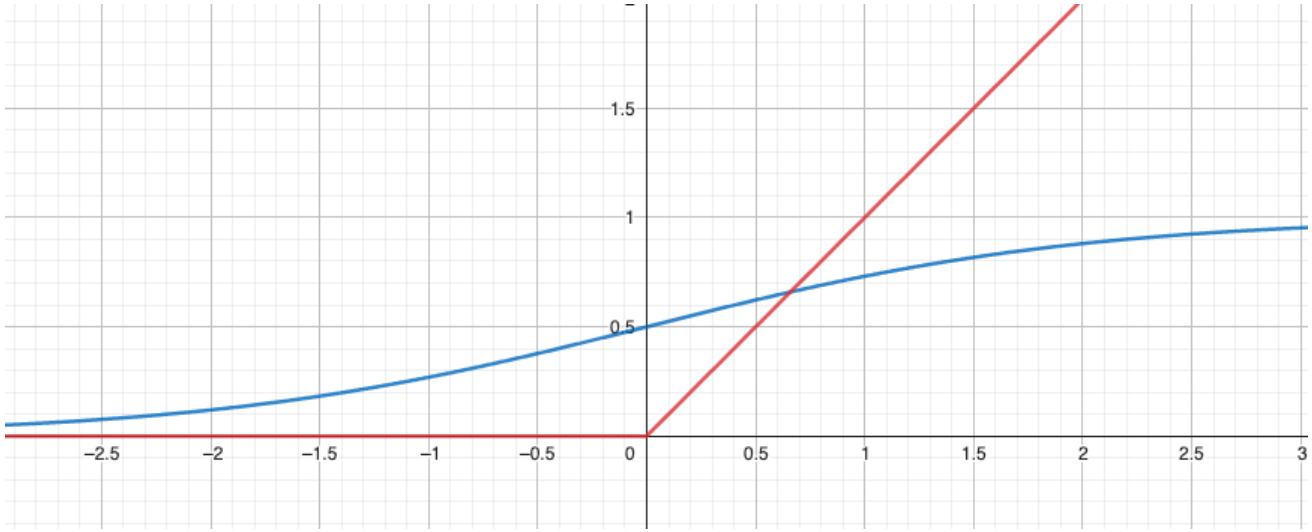


- Interpretation of the line $\theta^T \mathbf{x}$: Needs to change from a "**good fit**" to a **"decision boundary"**.

- Points just on the line – maximum uncertainty or $P(y|\mathbf{x}) = 0.5$.
- Points just above the line $\hat{y} = 1$ and $P(y|\mathbf{x}) \to 1$ with increasing distances from the boundary.
- Predictions then become $g(\theta^T \mathbf{x})$, where g is a function such that:
  - $g(0) = 0.5$
  - $g(z) \to 1$ as $z \to \infty$
  - $g(z) \to 0$ as $z \to -\infty$

Activation Functions

- The class of functions $g(z) \in [0, 1]$ with $g(0) = 0.5$ are called activation functions.
  - Sigmoid: $g(z) = \frac{1}{1+exp(-z)}$
  - Relu: $g(z) = \max(0, z)$



Model thus changes to:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

What's the loss function to minimize?

- **Mean Square Loss (MLS)**
  $L(\theta) = \arg\min_\theta (y - g(\theta^T \mathbf{x}))^2$ seems to work. Why?
  - $L(\theta)$ decreases if you predict a high value (e.g., 0.9) for $y = 1$
  - Also decreases if you predict a low value (e.g., 0.1) for $y = 0$.
  - All good: Mean squared error on activation functions potentially works as a classifier!
- Okay, but our *desired interpretation* of $g(\theta^T \mathbf{x})$ is that it's a **probability**.
- Isn't it then weird to see $(y - g(\theta^T \mathbf{x}))^2$? What does that even mean in terms of probabilities?

Cross-Entropy Loss

- Let's now revisit this in terms of modeling the probabilities in a theoretically sound manner!

Bernoulli Distribution

$$f = P(y|\mathbf{x};\theta) = g(\theta^T\mathbf{x})^y(1 - g(\theta^T\mathbf{x}))^{(1-y)}$$

- Note that $y \in \{0, 1\}$; $g(\theta^T\mathbf{x})$ is never 0 or 1.

Good prediction cases: (ϵ is a small number)

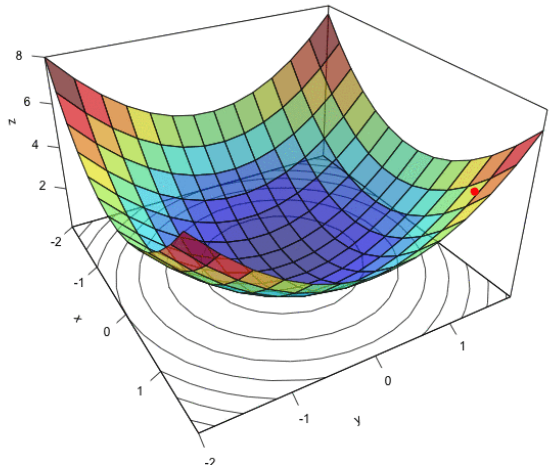
$y = 1$ and $g(\theta^T\mathbf{x}) \to 1$	$f = (1-\epsilon)^1\epsilon^0 \to 1$
$y = 0$ and $g(\theta^T\mathbf{x}) \to 0$	$f = \epsilon^0(1-\epsilon)^1 \to 1$

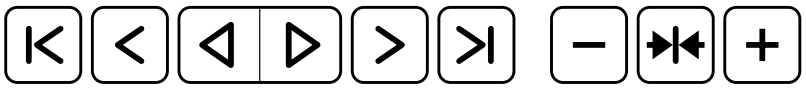
Exercise: Work out the 2 bad cases:

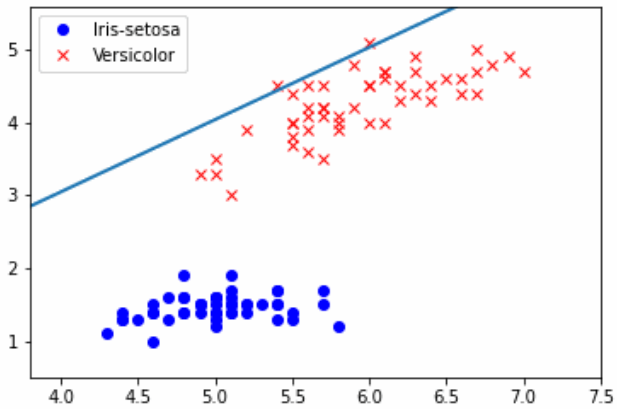
Loss/Objective function

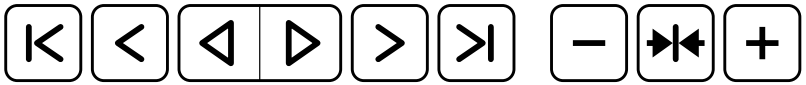
- $\arg\max_\theta \log P(y|\mathbf{x};\theta)$ works.
- $\arg\min_\theta -\log P(y|\mathbf{x};\theta)$ works (this is the negative log likelihood or the cross-entropy loss).
  - Note: Taking the **log** simplifies computation.

Visualizing the parameter space









- A **line** (boundary) in the **data space** $\equiv$ A **point** in the **parameter space**.
- Example: For a 2D data space (x_1, x_2), the parameter vector is three dimensional $\theta = (\theta_0, \theta_1, \theta_2)$.
- $y = 1$ for a point $\mathbf{x} = (x_1, x_2)$ if $\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0$

We need to compute the gradients!

- Finding way through the parameter space requires computing **gradients** $\rightarrow$ direction we should walk!
- To compute the gradient of the objective function, we need to compute the gradient of the **sigmoid**; because:
  - $\frac{d}{dx}g(f(x)) = \frac{d}{dy}g(y)\frac{d}{dx}f(x)$ (we call $f(x) = y$)

$$g'(z) = \frac{d}{dz}\frac{1}{1+e^{-z}}$$

$$= \frac{-1}{(1+e^{-z})^2}\frac{d}{dz}e^{-z}, \quad \because \frac{d}{dx}\frac{1}{x} = -\frac{1}{x^2}$$

$$= \frac{-1}{(1+e^{-z})^2}e^{-z}\frac{d}{dz}(-z), \quad \because \frac{d}{dx}e^{x} = e^{x}$$

$$= \frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{-z}}(1-\frac{1}{1+e^{-z}})$$

$$g'(z) = g(z)(1-g(z))$$

Gradient Descent

Loss function

$$l(\theta) = \log L(\theta) = y \log g(\theta^T \mathbf{x}) + (1-y)(1 - \log g(\theta^T \mathbf{x}))$$

- Partial derivative wrt one component of the parameter vector.

$$\frac{\partial}{\partial \theta_j} l(\theta) = (y \frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}) \frac{\partial}{\partial \theta_j} g(\theta^T x)$$

$$= (y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}) g(\theta^T x)(1-g(\theta^T x)) \frac{\partial}{\partial \theta_j}\theta^T x$$

$$= (y(1-g(\theta^T x)) - (1-y)g(\theta^T x)) x_j$$

$$= (y - g(\theta^T x))x_j = (y - h_\theta(x))x_j$$

Gradient Descent Algorithm

Loss function

$$\frac{\partial}{\partial \theta_j} l(\theta) = (y - h_\theta(x))x_j$$

- The form of this derivative is **identical** to that of the linear regression with square loss.
- However, these are **not the same algorithm**. Why?
  - $h_\theta(\mathbf{x}) = \theta^T \mathbf{x}$ for linear regression
  - $h_\theta(\mathbf{x}) = g(\theta^T \mathbf{x})$ for logistic regression
- In fact, these are same because both linear regression and logistic regression belong to the same family of models - that of **Generalized Linear Models** (out of the scope of our syllabus).

Logistic Regression Algorithm

Recap

- Activation: $h_\theta(\mathbf{x}) = g(\theta \cdot \mathbf{x}) = 1/(1 + \exp(-\theta \cdot \mathbf{x}))$
- Objective function: $h_\theta(\mathbf{x}) \log y + (1 - h_\theta(\mathbf{x})) \log(1 - y)$

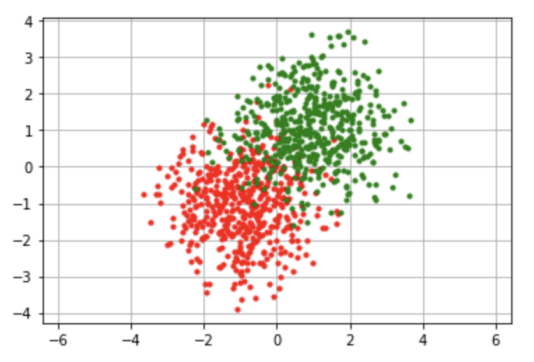
Algorithm

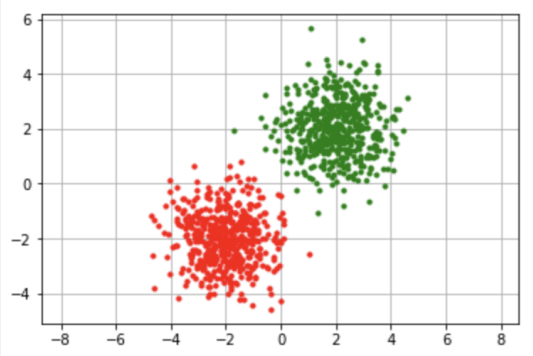
- For each training point $\mathbf{x}^{(i)}$:
  - For each parameter vector component j:
    - $\theta_j \leftarrow \theta_j + (y^{(i)} - h_\theta(\mathbf{x}^{(i)}))\mathbf{x}_j^{(i)}$

- Is the above algorithm sensitive to:
  - Order $\mathbf{x}^{(i)}$ of the training set?
  - Order of iterating over $j = 1, \ldots, d$?
  - Iterations?
  - Aggregating gradients over a batch of training instances?

Sensitivity of Logistic Regression Algorithm

Separability for 2D Gaussian samples



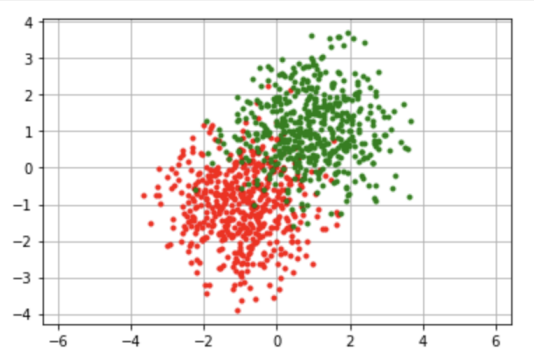


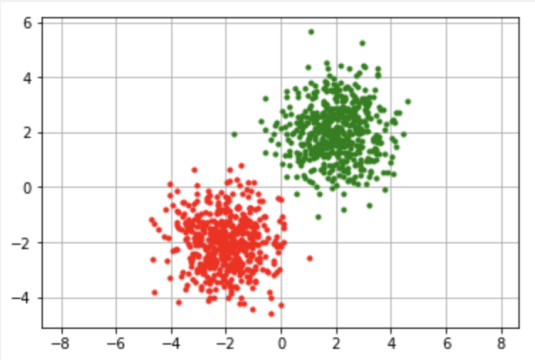
- Left: $g = \mathcal{N}((-1,-1), I_2), r = \mathcal{N}((1,1), I_2)$
- Right: $g = \mathcal{N}((-2,-2), I_2), r = \mathcal{N}((2,2), I_2)$

- Order $\mathbf{x}^{(i)}$ of the training set?
  - Medium to high sensitivity, specially when classes are not well-separable.
- Order of iterating over $j = 1, \ldots, d$?
  - Relatively low sensitivity.

Sensitivity of Logistic Regression Algorithm

Separability for 2D Gaussian samples



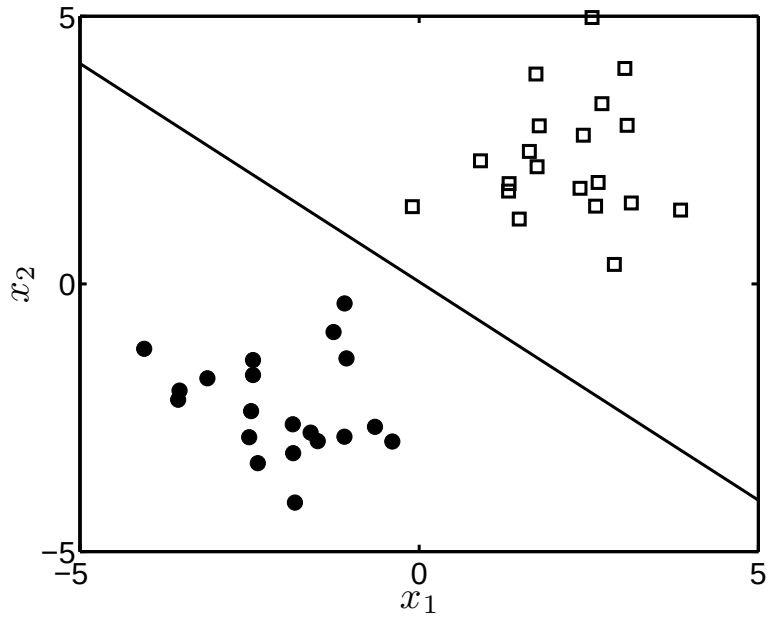


- Left: $g = \mathcal{N}((-1,-1), I_2), r = \mathcal{N}((1,1), I_2)$
- Right: $g = \mathcal{N}((-2,-2), I_2), r = \mathcal{N}((2,2), I_2)$

- Iterations:
  - Convergence much faster if classes are well-separable.
- Aggregating gradients over a batch of training instances?
  - Medium to high sensitivity, specially when classes are not well-separable.

Decision boundary

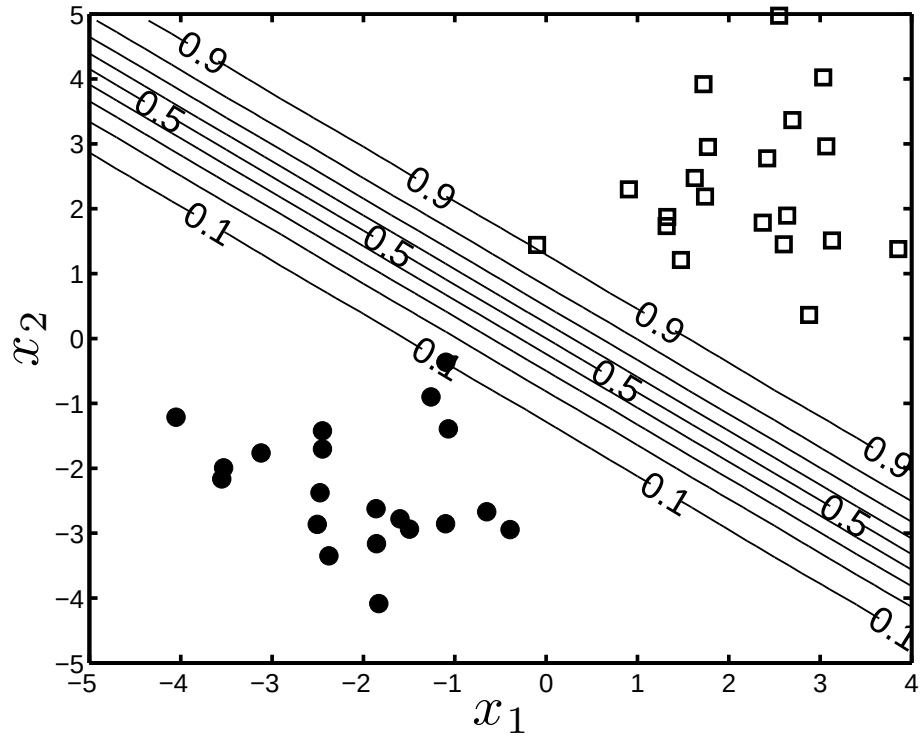
- Once we have θ, we can classify new examples.



Line corresponding to $P(y_{\text{new}} = 1 | \mathbf{x}_{\text{new}}; \theta) = 0.5$

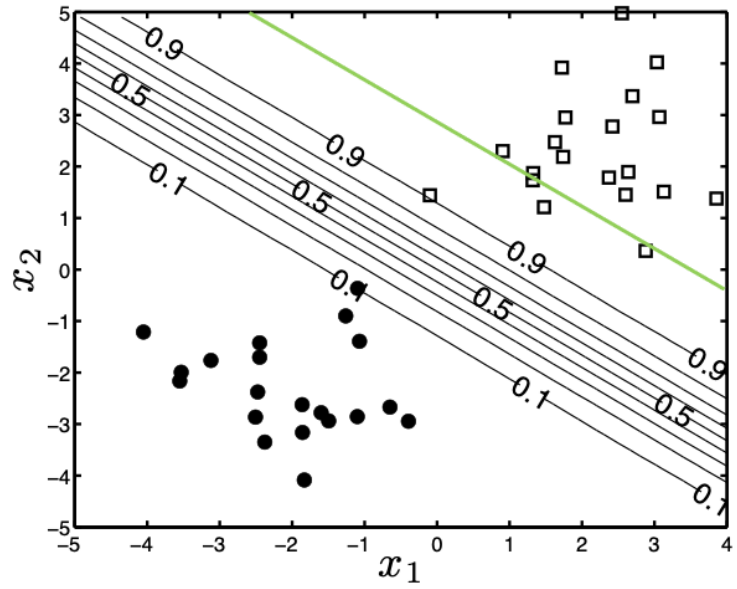
- $\theta^T \mathbf{x}_{new} = 0 \implies \exp(\theta^T \mathbf{x}_{new}) = 1$
- $\frac{1}{1+\exp(-\theta^T \mathbf{x}_{new})} = \frac{1}{2}$
- The classifier is the **most uncertain** (**least confident**) along the boundary.

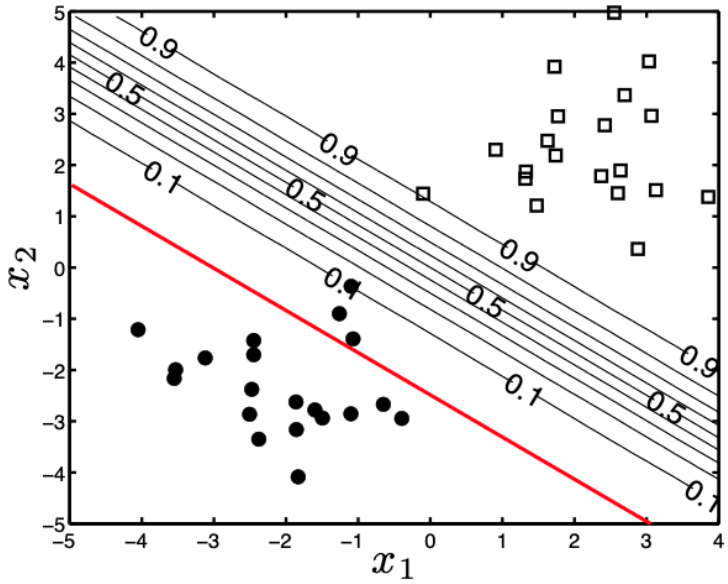
Contours of the posterior probabilities



- Contours of $P(y_{\text{new}} = 1 | \mathbf{x}_{\text{new}}, \theta)$.
- Do they look sensible?

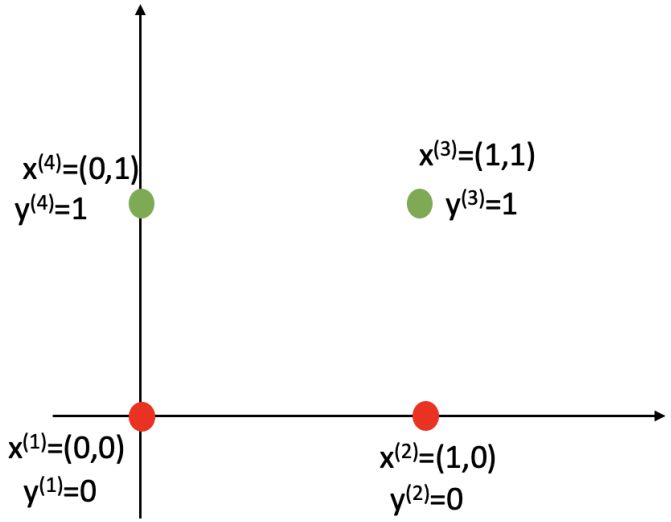
Model Calibration





- An easy way to change model predictions to be more conservative towards predicting a particular class.
- Confidence threshold set to a value $\tau \in [0, 1]$.
- Example: Predict $y = 1$ only if $P(y|x_{new}) > \tau$.
- Such calibrations are needed for critical tasks, such as cancer prediction, e.g., predict "`not cancer`" only if confidence > 0.95.

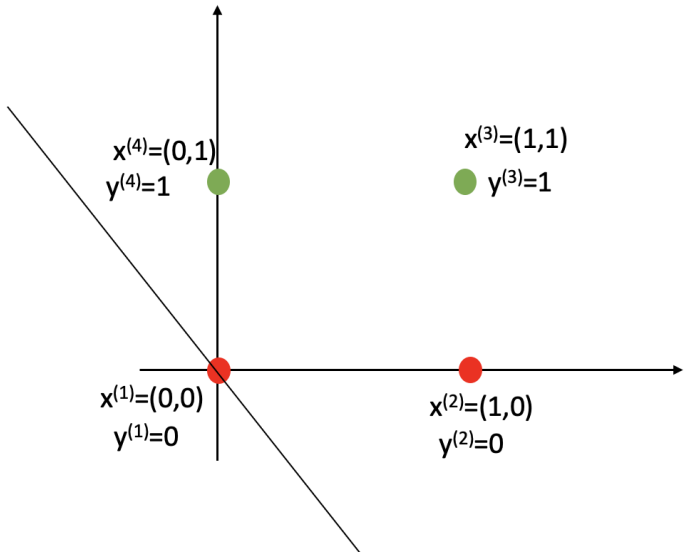
Logistic Regression Working Example



- Let's take $\theta = (0, 0.1, 0.1, 0)$ (the first term being the bias term). $\theta \in \mathbb{R}^3$.
- Each $\mathbf{x}$ needs to be prepended with a '1', e.g., we work with $\mathbf{x}^{(1)} = (1, 0, 0)$.

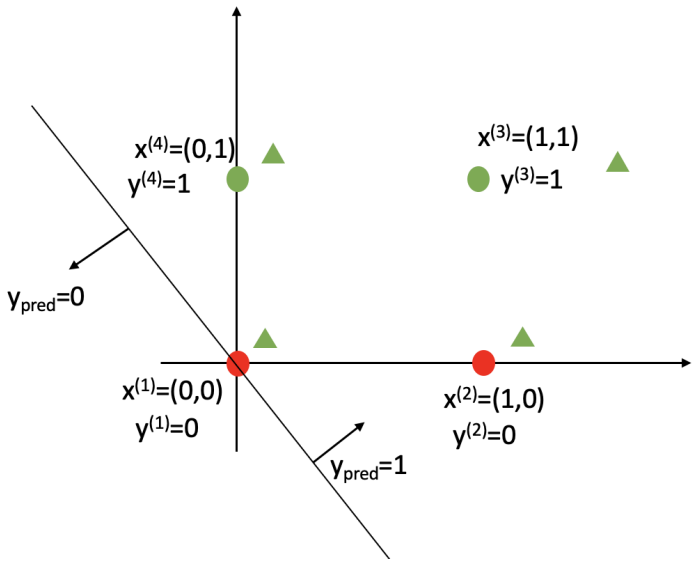
Logistic Regression Working Example

- How do we plot the decision boundary?
  - Remember at the boundary: $\theta \cdot \mathbf{x} = 0$.
- Visually plot the line $\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0$.
  - Substituting: $0 + 0.1x_1 + 0.1x_2 = 0 \implies x_2 = -x_1$.
- This is in the slope intercept form and hence we plot the boundary as shown below.



Logistic Regression Working Example

- Is our boundary good?
  - Let's denote our predictions with triangles.
  - How many misclassifications? 2.
- Now let's see how we can modify the boundary to do better.



Logistic Regression Working Example

Gradient Update

$$\theta_j \leftarrow \theta_j + (y^{(i)} - \mathit{sigmoid}(\theta \cdot \mathbf{x}^{(i)}))x_j^{(i)}.$$

- **Select a point** and **a component**.
  - Let's take the point $\mathbf{x}^{(2)}$. (Note that this is a point for which the current classifier makes a mistake!).
  - And take the second component, i.e., we update θ_1.

Logistic Regression Working Example

Gradient Update

$$\theta_j \leftarrow \theta_j + (y^{(i)} - sigmoid(\theta \cdot \mathbf{x}^{(i)}))x_j^{(i)}.$$

- **Select a point** and **a component**.
  - Let's take the point $\mathbf{x}^{(2)}$. (Note that this is a point for which the current classifier makes a mistake!).
  - And take the second component, i.e., we update θ_1.
- Compute:
  - $\theta \cdot \mathbf{x}^{(2)} = (0, 0.1, 0.1) \cdot (1, 1, 0) =$ $0 \times 1 + 0.1 \times 1 + 0.1 \times 0 =$ 0.1.

Logistic Regression Working Example

Gradient Update

$\theta_j \leftarrow \theta_j + (y^{(i)} - sigmoid(\theta \cdot \mathbf{x}^{(i)}))x_j^{(i)}$.

- **Select a point** and **a component**.
  - Let's take the point $\mathbf{x}^{(2)}$. (Note that this is a point for which the current classifier makes a mistake!).
  - And take the second component, i.e., we update θ_1.
- Compute:
  - $\theta \cdot \mathbf{x}^{(2)} = (0, 0.1, 0.1) \cdot (1, 1, 0) = 0 \times 1 + 0.1 \times 1 + 0.1 \times 0 = 0.1$.
  - $sigmoid(\theta \cdot \mathbf{x}^{(2)}) = 1/(1 + \exp(-0.1)) = \exp(0.1)/(1 + \exp(0.1)) = 1.1/2.1 = 0.52$.

Logistic Regression Working Example

Gradient Update

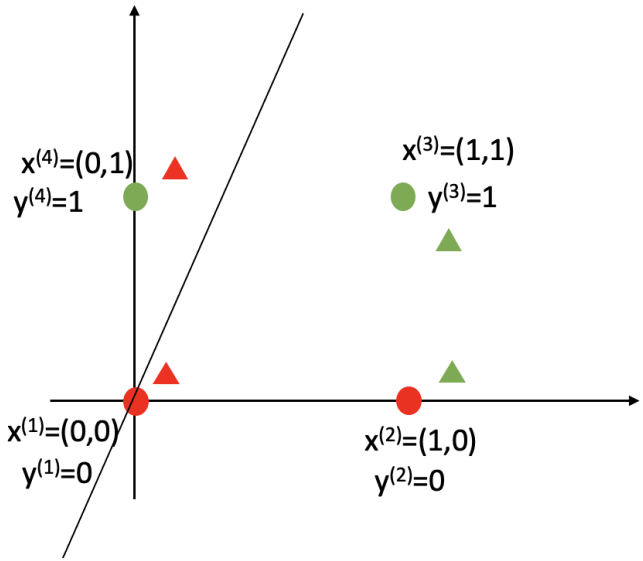
$\theta_j \leftarrow \theta_j + (y^{(i)} - sigmoid(\theta \cdot \mathbf{x}^{(i)}))x_j^{(i)}$.

- **Select a point** and **a component**.
  - Let's take the point $\mathbf{x}^{(2)}$. (Note that this is a point for which the current classifier makes a mistake!).
  - And take the second component, i.e., we update θ_1.
- Compute:
  - $\theta \cdot \mathbf{x}^{(2)} = (0, 0.1, 0.1) \cdot (1, 1, 0) = 0 \times 1 + 0.1 \times 1 + 0.1 \times 0 = 0.1$.
  - $sigmoid(\theta \cdot \mathbf{x}^{(2)}) = 1/(1 + \exp(-0.1)) = \exp(0.1)/(1 + \exp(0.1)) = 1.1/2.1 = 0.52$.
- Now modify each component of θ.
  - $\theta_1 \leftarrow 0.1 + (0 - 0.52) \times 1$.
- New parameter vector: $\theta = (0, -0.42, 0.1)$.

Logistic Regression Working Example

- How does the new boundary look like?
  - New parameter vector: $\theta = (0, -0.42, 0.1)$.
  - Put it in the slope intercept form.

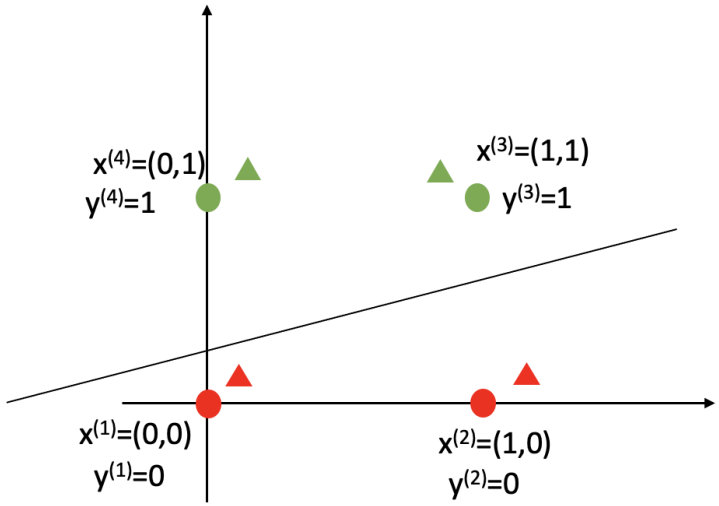
$$-0.42x_1 + 0.1x_2 = 0, \text{ i.e., } x_2 = \frac{0.42}{0.1}x_1 = 4.2x_1$$



- #Misclassifications: still 2; but we're making progress towards the ideal boundary.

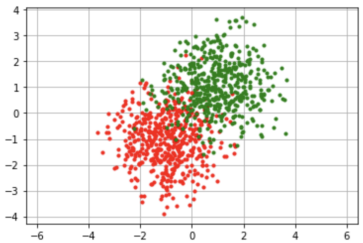
Logistic Regression Working Example

- Homework: Work out on paper one more update.
- If you run the updates for an adequate number of times (say 5 times), what do you expect the decision boundary to be?



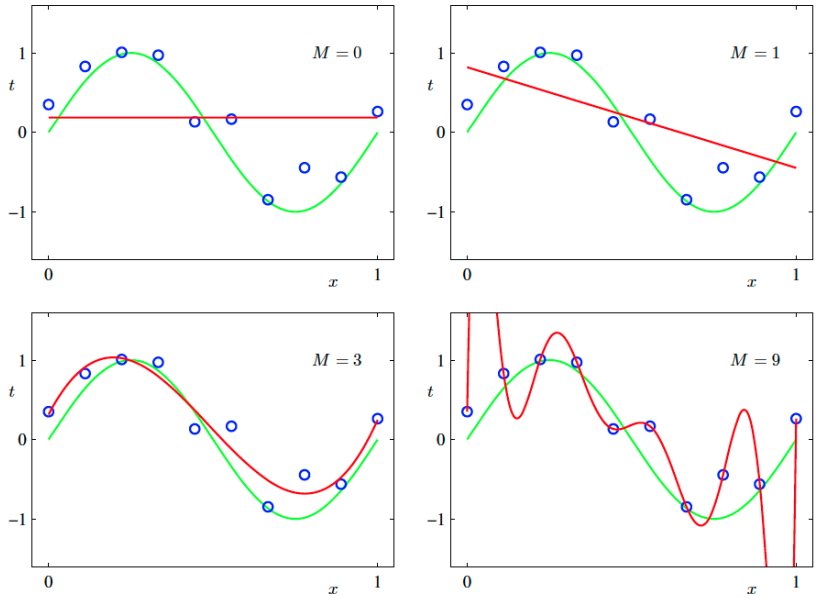
- If the classes are linearly separable logistic regression is guaranteed to converge to the perfect classifier.

What do we do for linearly inseparable classes?



- Logistic regression algorithm **can yield non-linear decision boundaries**.
- Use a feature map function of higher order features.
  - $\mathbf{x} = (x_1, \ldots, x_d)$
  - Higher order feature map examples:
    - $\phi_1(\mathbf{x}) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2)$
    - $\phi_2(\mathbf{x}) = (x_1, \ldots, x_d, x_1x_2, \ldots, x_{d-1}x_d)$
  - Apply logistic regression on $\phi(\mathbf{x})$ instead of on $\mathbf{x}$.
- This means that decision boundary $\theta_1 x_1 + \theta_{d+1} x_1^2 + \ldots$ has now non-linear terms.

Problems of using a feature map



- Minimise:
  $J(\theta) = -(h_\theta(\mathbf{x}) \log y + (1 - h_\theta(\mathbf{x})) \log(1 - y)) + ||\theta||_p$

- The same problem as in linear regression - **Overfitting**.
- **Solution**: Encourage sparse solutions - higher order features but you don't want the boundary to depend on too many of them.
  - Modify objective function with an added term for the norm of the parameter vector
  - Penalize those θs whose norm is high.

Classifier Performance Evaluation

- How do we know how accurate are our predictions?
  - Which algorithm? Regularization or without
  - What model calibration?
- Need performance indicators.

Classifier Performance Evaluation

- How do we know how accurate are our predictions?
  - Which algorithm? Regularization or without
  - What model calibration?
- Need performance indicators.
- We'll cover:
  - Accuracy
  - Precision/Recall
  - Confusion matrix and Precision-recall curves

Accuracy

- How many correct classifications in total.
- Consider a set of predictions $\hat{y}_1, \ldots, \hat{y}_N$ and a set of true labels $y_1^{'} \ldots, y_N$.
- Mean accuracy is defined as:

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{y}_i \neq y_i)$$

- $\mathbb{I}(A)$ is 1 if A is true and 0 otherwise

Accuracy

- How many correct classifications in total.
- Consider a set of predictions $\hat{y}_1, \ldots, \hat{y}_N$ and a set of true labels $y_1^{'} \ldots, y_N$.
- Mean accuracy is defined as:

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{y}_i \neq y_i)$$

- $\mathbb{I}(A)$ is 1 if A is true and 0 otherwise
- Advantages:
  - Can do binary or multi-class classification.
  - Simple to compute.
  - Single value.

Accuracy

Disadvantage: Doesn't take into account **class imbalance**.

- We're building a classifier to detect a rare disease.
- Assume only 1% of population is diseased.
- Diseased: $y = 1$
- Healthy: $y = 0$
- What if we always predict healthy? $(y = 0)$
- Accuracy 99%
- But classifier is rubbish!

Precision and Recall

- We'll stick with our disease example.
- Need to define 4 quantities. The numbers of:

Precision and Recall

- We'll stick with our disease example.
- Need to define 4 quantities. The numbers of:
- **True positives (TP)** – the number of objects with $y = 1$ that are classified as $\hat{y} = 1$ (diseased people diagnosed as diseased).

Precision and Recall

- We'll stick with our disease example.
- Need to define 4 quantities. The numbers of:
- True positives (TP) – the number of objects with $y = 1$ that are classified as $\hat{y} = 1$ (diseased people diagnosed as diseased).
- **True negatives (TN)** – the number of objects with $y = 0$ that are classified as $\hat{y} = 0$ (healthy people diagnosed as healthy).

Precision and Recall

- We'll stick with our disease example.
- Need to define 4 quantities. The numbers of:
- True positives (TP) – the number of objects with $y = 1$ that are classified as $\hat{y} = 1$ (diseased people diagnosed as diseased).
- True negatives (TN) – the number of objects with $y = 0$ that are classified as $\hat{y} = 0$ (healthy people diagnosed as healthy).
- **False positives (FP)** – the number of objects with $y = 0$ that are classified as $\hat{y} = 1$ (healthy people diagnosed as diseased).

Precision and Recall

- We'll stick with our disease example.
- Need to define 4 quantities. The numbers of:
- True positives (TP) – the number of objects with $y = 1$ that are classified as $\hat{y} = 1$ (diseased people diagnosed as diseased).
- True negatives (TN) – the number of objects with $y = 0$ that are classified as $\hat{y} = 0$ (healthy people diagnosed as healthy).
- False positives (FP) – the number of objects with $y = 0$ that are classified as $\hat{y} = 1$ (healthy people diagnosed as diseased).
- **False negatives (FN)** – the number of objects with $y = 1$ that are classified as $\hat{y} = 0$ (diseased people diagnosed as healthy).

Precision vs. Recall

$$P = \frac{TP}{TP + FP}$$

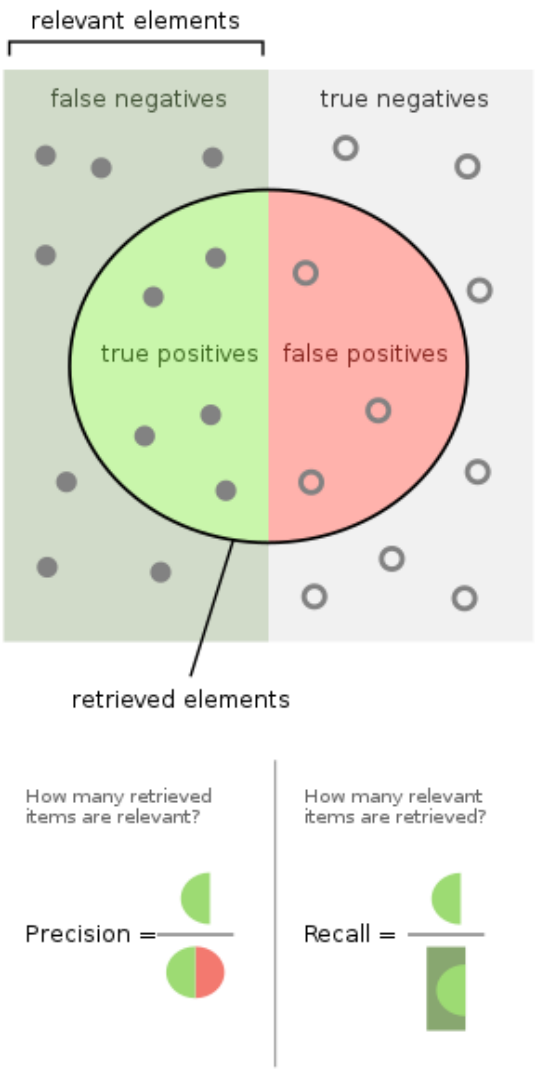
- The proportion of correctly identified diseased people from all our 'diseases' predictions.
- The higher the better.
- In our example, $P = 0$.

$$R = \frac{TP}{TP + FN}$$

- The proportion of diseased people that are able to identify out of all the diseased people out there.
- The higher the better.
- In our example, $R = 0$.

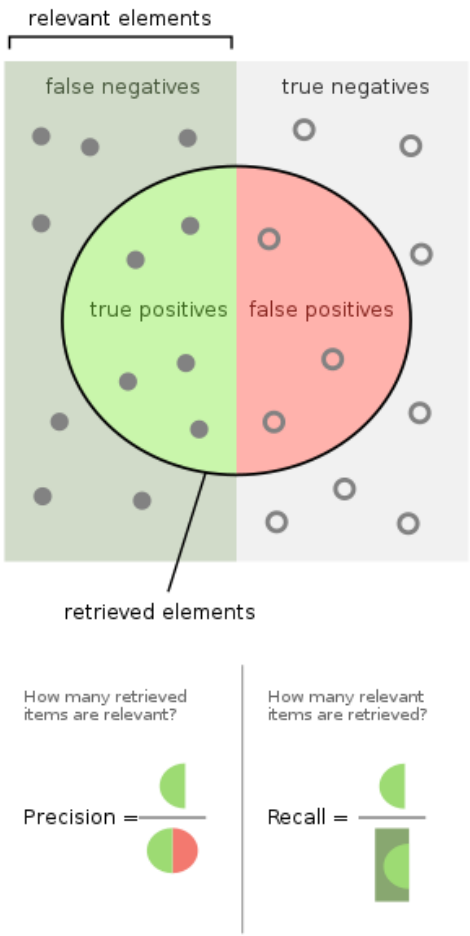
- In biomedical domain:
  - Recall of the positive class: 'Sensitivity'.
  - Recall of the negative class: 'Specificity'.

Visual representation of precision and recall



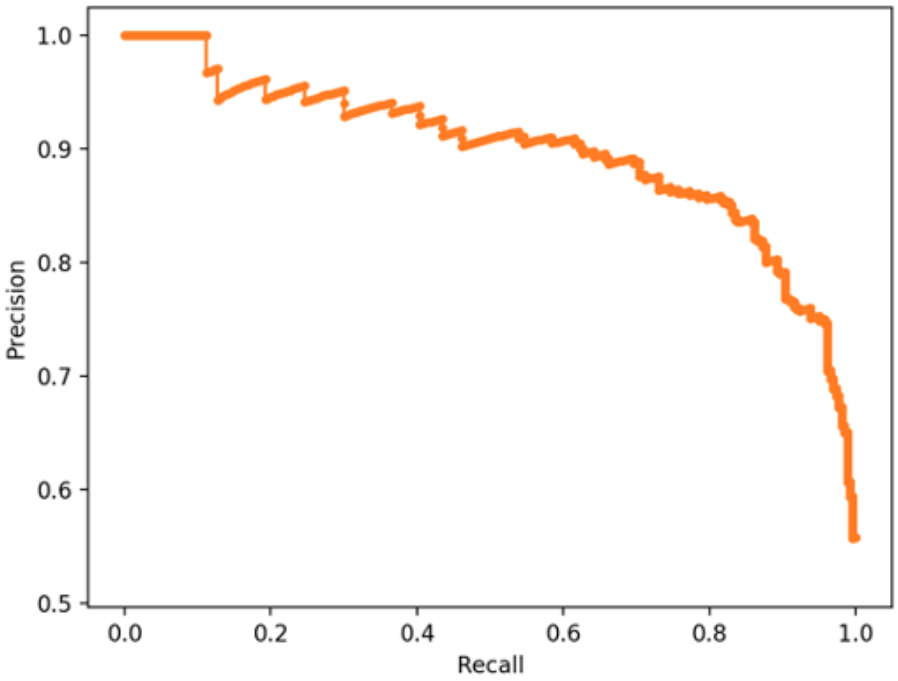
- Effective for class imbalanced problems.
- Two different performance measures used for specific types of tasks - **precision oriented** or **recall oriented**.
- Trade-off between the two. Why?
- Examples:
  - Detecting cancer cells from images is a recall-oriented task. Why?
  - Whether to turn an AI-driven car is a precision-oriented task. Why?

Precision-Recall Curve



- Useful to analyze the effects of model calibration.
- Default threshold in logistic regression is 0.5.
- However, we could use any threshold we like.
- Too conservative a threshold:
  - Moves the circle towards the right (Why?)
  - Precision↑, Recall↓.
- Too liberal a threshold
  - Moves the circle towards the left.
  - Precision↓, Recall↑.

Precision-Recall Curves curve



- Helps calibrating a system by comparing its performance over a range of different thresholds.
- A curve representing an effective calibration **mostly stays above** any other line.
- With an increase from low to high recall, the precision should stay as high as possible for an ideal classifier.
- Each error brings precision down.

Confusion matrices

The quantities we used to compute precision and recall can be neatly summarised in a table:

		True class 1	True class 0
Predicted class	1	TP	FP
	0	FN	TN

- This is known as a *confusion matrix*
- It is particularly useful for multi-class classification.
- Tells us where the mistakes are being made.

		True class										
	...	10	11	12	13	14	15	16	18	18	19	20
Predicted class 1	...	4	2	0	2	10	4	7	1	12	7	47
2	...	0	0	4	18	7	8	2	0	1	1	3
3	...	0	0	1	0	1	0	1	0	0	0	0
4	...	1	0	1	28	3	0	0	0	0	0	0
						⋮						
16	...	3	2	2	5	17	4	376	3	7	2	**68**
17	...	1	0	9	0	3	1	3	325	3	**95**	19
18	...	2	1	0	2	6	2	1	2	325	4	5
19	...	8	4	8	0	10	21	1	16	19	**185**	7
20	...	0	0	1	0	1	1	2	4	0	1	**92**

Predicted class		True class										
	...	10	11	12	13	14	15	16	18	18	19	20
1	...	4	2	0	2	10	4	7	1	12	7	47
2	...	0	0	4	18	7	8	2	0	1	1	3
3	...	0	0	1	0	1	0	1	0	0	0	0
4	...	1	0	1	28	3	0	0	0	0	0	0
						⋮						
16	...	3	2	2	5	17	4	376	3	7	2	**68**
17	...	1	0	9	0	3	1	3	325	3	**95**	19
18	...	2	1	0	2	6	2	1	2	325	4	5
19	...	8	4	8	0	10	21	1	16	19	**185**	7
20	...	0	0	1	0	1	1	2	4	0	1	**92**

- Algorithm is getting 'confused' between classes 20 and 16, and 19 and 17.
  - 17: talk.politics.guns
  - 19: talk.politics.misc

Predicted class	True class											
	...	10	11	12	13	14	15	16	18	18	19	20
1	...	4	2	0	2	10	4	7	1	12	7	47
2	...	0	0	4	18	7	8	2	0	1	1	3
3	...	0	0	1	0	1	0	1	0	0	0	0
4	...	1	0	1	28	3	0	0	0	0	0	0
						⋮						
16	...	3	2	2	5	17	4	376	3	7	2	**68**
17	...	1	0	9	0	3	1	3	325	3	**95**	19
18	...	2	1	0	2	6	2	1	2	325	4	5
19	...	8	4	8	0	10	21	1	16	19	**185**	7
20	...	0	0	1	0	1	1	2	4	0	1	**92**

- Algorithm is getting 'confused' between classes 20 and 16, and 19 and 17.
  - 17: talk.politics.guns
  - 19: talk.politics.misc
  - 16: talk.religion.misc
  - 20: soc.religion.christian

Predicted class \ True class	...	10	11	12	13	14	15	16	18	18	19	20
1	...	4	2	0	2	10	4	7	1	12	7	47
2	...	0	0	4	18	7	8	2	0	1	1	3
3	...	0	0	1	0	1	0	1	0	0	0	0
4	...	1	0	1	28	3	0	0	0	0	0	0
⋮												
16	...	3	2	2	5	17	4	376	3	7	2	**68**
17	...	1	0	9	0	3	1	3	325	3	**95**	19
18	...	2	1	0	2	6	2	1	2	325	4	5
19	...	8	4	8	0	10	21	1	16	19	**185**	7
20	...	0	0	1	0	1	1	2	4	0	1	**92**

- Algorithm is getting 'confused' between classes 20 and 16, and 19 and 17.
  - 17: talk.politics.guns
  - 19: talk.politics.misc
  - 16: talk.religion.misc
  - 20: soc.religion.christian
- Maybe these should be just one class?
- Maybe we need more data in these classes?

Predicted class \ True class	...	10	11	12	13	14	15	16	18	18	19	20
1	...	4	2	0	2	10	4	7	1	12	7	47
2	...	0	0	4	18	7	8	2	0	1	1	3
3	...	0	0	1	0	1	0	1	0	0	0	0
4	...	1	0	1	28	3	0	0	0	0	0	0
⋮												
16	...	3	2	2	5	17	4	376	3	7	2	**68**
17	...	1	0	9	0	3	1	3	325	3	**95**	19
18	...	2	1	0	2	6	2	1	2	325	4	5
19	...	8	4	8	0	10	21	1	16	19	**185**	7
20	...	0	0	1	0	1	1	2	4	0	1	**92**

- Algorithm is getting 'confused' between classes 20 and 16, and 19 and 17.
  - 17: talk.politics.guns
  - 19: talk.politics.misc
  - 16: talk.religion.misc
  - 20: soc.religion.christian
- Maybe these should be just one class?
- Maybe we need more data in these classes?
- Confusion matrix helps us direct our efforts to improving the classifier.

A word about the practical coursework (25%)

- Will be released tomorrow!
- Dataset: Features with Labels
- Task: Breast Cancer Detection
- Quite flexible - choose your own favorite classifier, e.g., just use logistic regression (that we learned today).
- **Not just a coding exercise**.
- Write a report **explaining** and **motivating** your design choices and hyper-parameters - e.g.:
  - feature maps used?
  - regularization used?
  - calibration threshold?
- Also asks you to **calibrate** the model to investigate the **trade-off between precision and recall**.