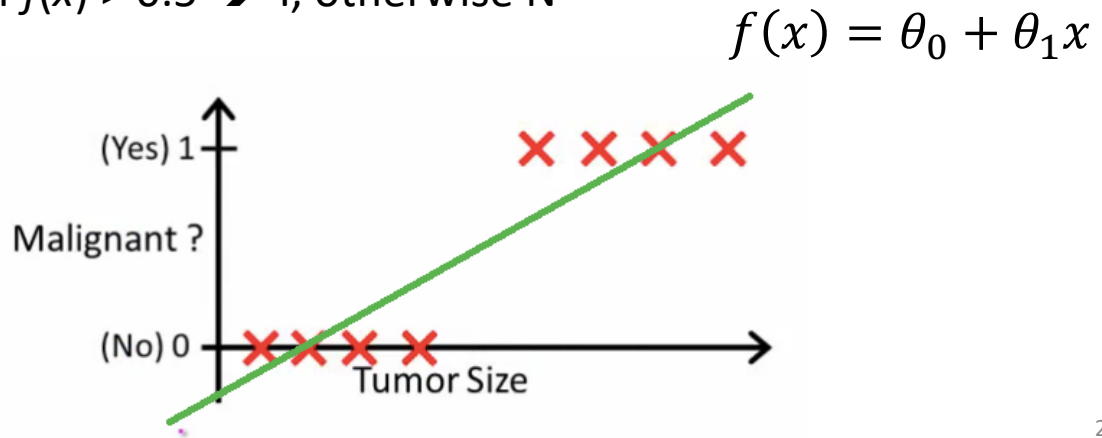


Logistic regression

Hung-Hsuan Chen

Classification by linear regression?

- Use tumor size (feature) to predict tumor type (binary label: malignant or not)
 - We've learned linear regression... can we leverage on such a model?
 - If $f(x) > 0.5 \rightarrow Y$, otherwise N



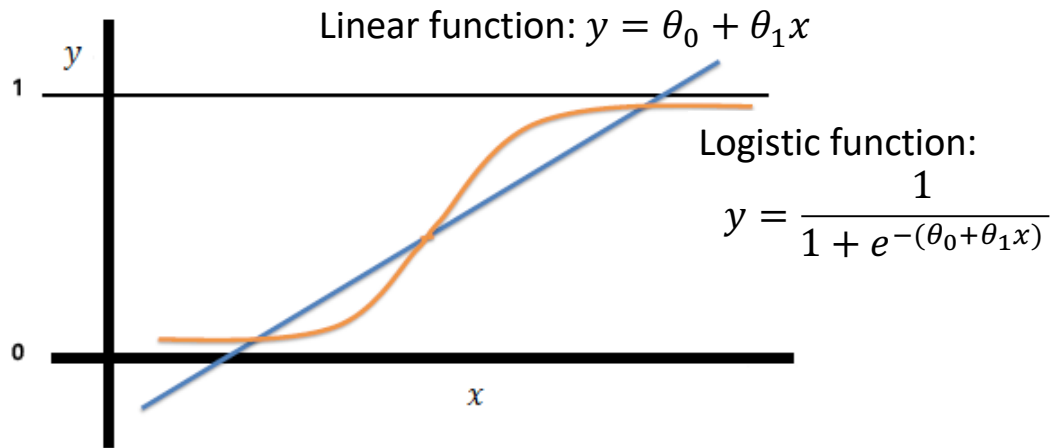
Linear regression is highly affected by the extreme values

- Observing a very large malignant tumor (or a very small benign tumor) should affect the model
 - However, linear regression is highly affected by the extreme values



Fitting an S-shaped function (instead of linear function)

- If the fitting curve is S-shaped, the attributes with extreme (very large or very small) values will affect very little to the fitted curve

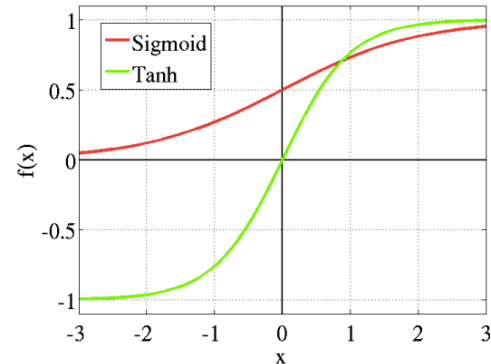


Sigmoid function

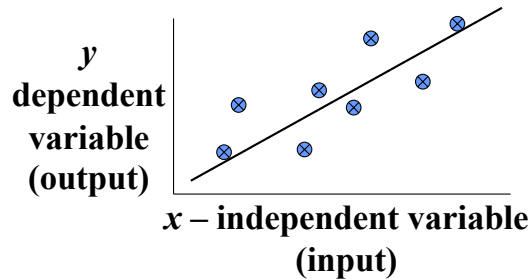
- A sigmoid function is a mathematical function having an "S" shaped curve (sigmoid curve)
 - Logistic function (the “classic” sigmoid function)
 - $f(x) = \frac{1}{1+e^{-x}}$
 - Hyperbolic tangent function (a.k.a. tanh function)
 - $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$

Logistic vs tanh function

- Logistic
 - Target value range: $(0, 1)$
 - Binary classification
 - Positive: denoted by 1
 - Negative: denoted by 0
- Tanh
 - Target value range: $(-1, 1)$
 - Binary classification
 - Positive: denoted by 1
 - Negative: denoted by -1



Review: linear regression

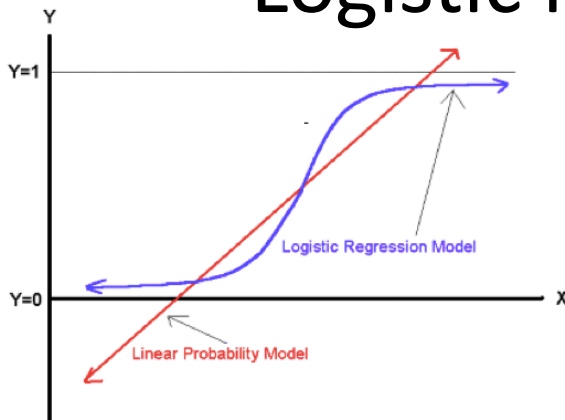


“Predictor”:

$$\hat{y} = \theta_0 + \theta_1 x$$

- Define the form of the function $f(x)$ explicitly
 - i.e., $\hat{y} = \theta_0 + \theta_1 x$ in this case
- Find a good $f(x)$ within that family
 - i.e. find good θ_0 and θ_1 such that $\hat{y}_i \approx y_i \forall i$
 - Loss function: **sum-of-squares**

Logistic regression



“Predictor”:

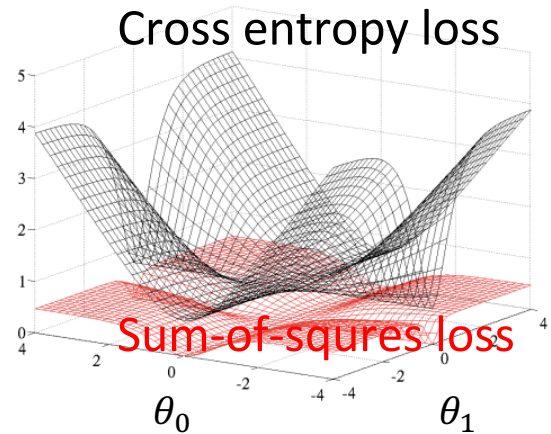
$$\hat{y} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

$f(x) = \hat{y}$ can be interpreted as the probability of $y=1$ given the feature vector \mathbf{x}

- Define form of function $f(x)$ explicitly
 - i.e., $f(x) = \hat{y} = \frac{1}{1+e^{-(\theta_0+\theta_1x)}} = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$ in this case
- Find a good $f(x)$ within that family
 - i.e. find good θ_0 and θ_1 such that $\hat{y}_i \approx y_i \forall i$
 - Loss function: **cross entropy loss** (will explain later)

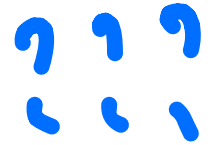
Why not using sum of square loss?

- Is it reasonable to use:
 - logistic regression as the model,
 - **sum-of-square as the loss function?**
- Conceptually, using sum-of-squares loss might be reasonable
- However, using cross-entropy-loss is more efficient computationally
- When the current value is far from the optimal, the derivative of sum-of-square loss is very small



(Adapted from
Glorot and Bengio,
AISTATS 2010)

Probability and likelihood



- The probability of the value of y

- $P(y = 1) = f(\mathbf{x}) = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$

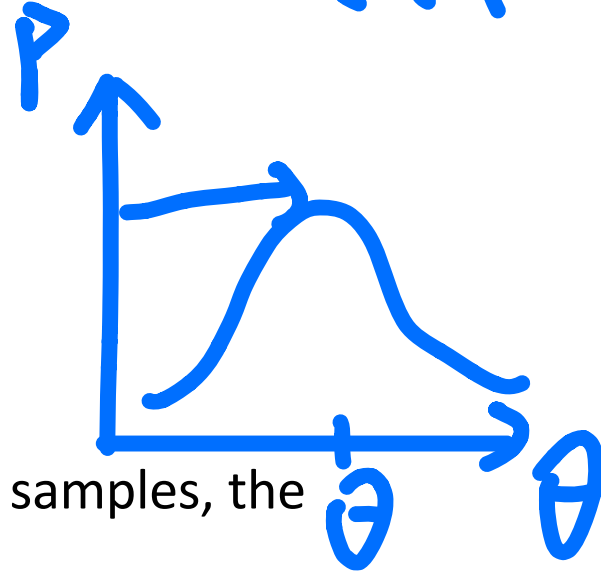
- $P(y = 0) = 1 - f(\mathbf{x})$

- ➔ $P(y) = f(\mathbf{x})^y (1 - f(\mathbf{x}))^{1-y}$

- If we have n independent training samples, the likelihood of the parameter θ is

- $L(\theta) = \prod_{i=1}^n p(y_i) = \prod_{i=1}^n f(\mathbf{x}_i)^{y_i} (1 - f(\mathbf{x}_i))^{(1-y_i)}$

- \mathbf{x}_i : the i th training instance (a vector); y_i : the i th training label (a scalar)



Likelihood and log likelihood

- We want to find $\boldsymbol{\theta} = (\theta_0, \theta_1)$ to maximize the likelihood
 $\boldsymbol{\theta} := \arg \max_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$

$$= \arg \max_{\boldsymbol{\theta}} \prod_{i=1}^n f(\mathbf{x}_i)^{y_i} (1 - f(\mathbf{x}_i))^{(1-y_i)}$$

- The $\boldsymbol{\theta}$ to maximize the likelihood is the same as the $\boldsymbol{\theta}$ to maximize the log-likelihood

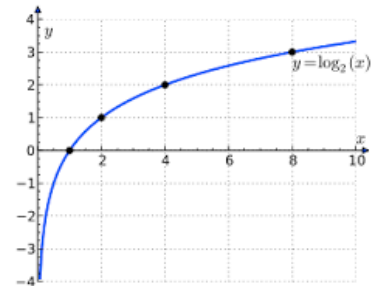
– Log function monotonically increasing

$$\boldsymbol{\theta} := \arg \max_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

$$= \arg \max_{\boldsymbol{\theta}} \log(L(\boldsymbol{\theta}))$$

$$= \arg \max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$$

$$= \arg \max_{\boldsymbol{\theta}} \sum \{y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))\}$$



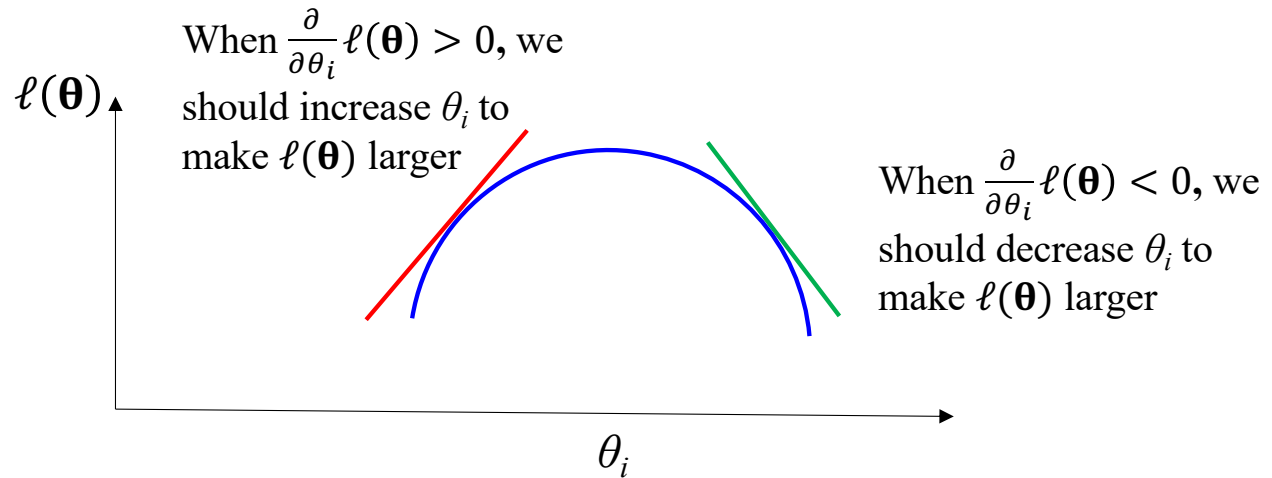
Why use log-likelihood?

- Change multiplications to summations
 - Summations are faster than multiplications
 - Numerically, multiplying many tiny numbers tend to underflow; multiplying many huge numbers tend to overflow

How to find θ ?

- Method 1: closed form solution
 - Unfortunately, there is no closed form solution for logistic regression
 - Method 2: gradient descent
 - Set the loss function as $-\ell(\theta)$
 - Assign random values to the initial θ_0 and θ_1
 - Gradually adjust θ_0 and θ_1 such that $-\ell(\theta)$ becomes smaller
 - Method 3: gradient ascent
 - Assign random values to the initial θ_0 and θ_1
 - Gradually adjust θ_0 and θ_1 such that $\ell(\theta)$ becomes larger
- $\ell(\theta)$ is a concave function

Gradient ascent



Derivative of a logistic function


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

$$\begin{aligned}\Rightarrow g'(x) &= -1(1 + e^{-x})^{-2}e^{-x}(-1) = \frac{e^{-x}}{(1+e^{-x})^2} \\ &= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\ &= g(x)(1 - g(x))\end{aligned}$$

If $g(x)$ is the logistic function, then:

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

Derivative of the log-likelihood function $\ell(\boldsymbol{\theta})$

$$f(\mathbf{x}_i) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x}_i)} \equiv g(\boldsymbol{\theta}^T \mathbf{x}_i)$$


$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \sum_{i=1}^n \{y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))\} = \sum_{i=1}^n \{y_i \log(g(\boldsymbol{\theta}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - g(\boldsymbol{\theta}^T \mathbf{x}_i))\} \\ &\Rightarrow \frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_j} \\ &= \sum_{i=1}^n \left\{ \frac{\partial y_i \log(g(\boldsymbol{\theta}^T \mathbf{x}_i))}{\partial \log(g(\boldsymbol{\theta}^T \mathbf{x}_i))} \cdot \frac{\partial \log(g(\boldsymbol{\theta}^T \mathbf{x}_i))}{\partial g(\boldsymbol{\theta}^T \mathbf{x}_i)} \cdot \frac{\partial g(\boldsymbol{\theta}^T \mathbf{x}_i)}{\partial \boldsymbol{\theta}^T \mathbf{x}_i} \cdot \frac{\partial \boldsymbol{\theta}^T \mathbf{x}_i}{\partial (\theta_j)} \right\} \\ &\quad + \sum_{i=1}^n \left\{ \frac{\partial (1-y_i) \log(1-g(\boldsymbol{\theta}^T \mathbf{x}_i))}{\partial \log(1-g(\boldsymbol{\theta}^T \mathbf{x}_i))} \cdot \frac{\partial \log(1-g(\boldsymbol{\theta}^T \mathbf{x}_i))}{\partial (1-g(\boldsymbol{\theta}^T \mathbf{x}_i))} \cdot \frac{\partial (1-g(\boldsymbol{\theta}^T \mathbf{x}_i))}{\partial g(\boldsymbol{\theta}^T \mathbf{x}_i)} \cdot \frac{\partial g(\boldsymbol{\theta}^T \mathbf{x}_i)}{\partial \boldsymbol{\theta}^T \mathbf{x}_i} \cdot \frac{\partial \boldsymbol{\theta}^T \mathbf{x}_i}{\partial (\theta_j)} \right\} \\ &= \sum_{i=1}^n \left\{ y_i \cdot \frac{1}{g(\boldsymbol{\theta}^T \mathbf{x}_i)} \cdot g(\boldsymbol{\theta}^T \mathbf{x}_i) \cdot (1 - g(\boldsymbol{\theta}^T \mathbf{x}_i)) \cdot x_{ij} \right\} \\ &\quad + \sum_{i=1}^n \left\{ (1 - y_i) \cdot \frac{1}{1 - g(\boldsymbol{\theta}^T \mathbf{x}_i)} \cdot (-1) \cdot g(\boldsymbol{\theta}^T \mathbf{x}_i) \cdot (1 - g(\boldsymbol{\theta}^T \mathbf{x}_i)) \cdot x_{ij} \right\} \\ &= \sum_{i=1}^n \{x_{ij}(y_i - \hat{y}_i)\} \end{aligned}$$

Using gradient ascend to find θ

- Matrix form: $\frac{\nabla \ell(\theta)}{\nabla \theta} = (\mathbf{y} - \hat{\mathbf{y}})^T \mathbf{X}$

- Gradient ascent algorithm:

- $\theta^{(k+1)} := \theta^{(k)} + \alpha \left(\frac{\nabla \ell(\theta)}{\nabla \theta} \right)^T$

```
1   Repeat until converge {  
2        $\hat{\mathbf{y}} = 1/(1 + e^{-\mathbf{X}\theta})$   
3        $\theta^{(k+1)} := \theta^{(k)} + \alpha \mathbf{X}^T (\mathbf{y} - \hat{\mathbf{y}})$   
4   }
```

Cross entropy and log-likelihood of classification problem

- Maximize the log-likelihood function
 - $\theta = \arg \text{max}(\sum\{y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\})$
- This is the same as minimizing the negative of the log-likelihood function
 - $\theta = \arg \text{min}(-\sum\{y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\})$
 - This is called the “**cross entropy loss**” function

Cross entropy loss

- **Cross entropy loss function**
 - $\ell = -\sum\{y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\}$, where
 - $\hat{y}_i = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_i)}}$
 - $\boldsymbol{\theta} = \arg \min(-\sum\{y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\})$
- Check:
 - If $y_i = 0$ and $\hat{y}_i \rightarrow 0$: $\ell \rightarrow 0$
 - If $y_i = 0$ and $\hat{y}_i \rightarrow 1$: $\ell \rightarrow \infty$
 - If $y_i = 1$ and $\hat{y}_i \rightarrow 0$: $\ell \rightarrow \infty$
 - If $y_i = 1$ and $\hat{y}_i \rightarrow 1$: $\ell \rightarrow 0$

Regularization to avoid overfitting

- Original goal:

$$\begin{aligned}\boldsymbol{\theta} &:= \arg \max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) \\ &= \arg \max_{\boldsymbol{\theta}} \sum \{y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))\}\end{aligned}$$

- We also want the θ 's to be small (prevent overfitting)
- New goal:

$$\boldsymbol{\theta} := \arg \max_{\boldsymbol{\theta}} \left[\sum \{y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i))\} - \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \right]$$

— Penalize high weights, like we did in linear regression!

Derivative of $\ell(\boldsymbol{\theta})$

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^n \{y_i \log(g(\boldsymbol{\theta}^T \mathbf{x}_i)) + (1 - y_i) \log(1 - g(\boldsymbol{\theta}^T \mathbf{x}_i))\} - \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2$$

$$\Rightarrow \frac{\partial \ell(\boldsymbol{\theta})}{\partial \theta_j} = \sum_{i=1}^n \{x_{ij}(y_i - \hat{y}_i)\} - \lambda \theta_j$$

Concept drift

- **Concept drift**
 - The statistical properties of the target variable change over time
- Offline (batch) learning
 - Generate the best predictor by learning on the entire training data set **at once**
 - Need to re-train the model every once a while
- Online machine learning
 - Data becomes available in a sequential order
 - Use the latest data instances to gradually update the model

Gradient descent/stochastic gradient descent/mini-batch gradient descent

- All of them iteratively update the parameters such that the target function gradually becomes smaller
- If we have n training instances
 - (Batch) gradient descent: every parameter update requires seeing all training instances once
 - Stochastic gradient descent: every parameter update requires seeing one of n training instances
 - Mini-batch: every parameter update requires seeing b training instances (if batch size = b)

Stochastic gradient descent for online learning

- As the distribution of the data shifted, the model gradually influenced by the latest data instances
- SGD can be applied on linear regression and logistic regression

Quiz

- Compare similarities and differences of linear regression and logistic regression
- What is “cross entropy loss”?
- How many parameter updates per epoch if we applying SGD on n training instances?
- Is decision tree classifier an online learning or a batch learning algorithm?

Classification metrics

Classification metrics

- Accuracy
- Precision
- Recall
- F1 score
- Precision recall curve
- Sensitivity vs specificity
- ROC curve and AUC

Accuracy

- $\text{Accuracy}(y, \hat{y}) = \frac{1}{n} \sum I(y_i = \hat{y}_i)$
- Is 0.5 a good accuracy?
- Is 0.9 a good accuracy?
 - Imbalanced binary classification
- Is 0.1 a bad accuracy?
 - Multi-class classification

Confusion matrix

(assume binary classification)

		predicted condition	
total population		prediction positive	prediction negative
true condition	condition positive	True Positive (TP)	False Negative (FN) (type II error)
	condition negative	False Positive (FP) (Type I error)	True Negative (TN)

Example

		predicted condition	
total population		prediction positive	prediction negative
true condition	condition positive	True Positive (TP) 20	False Negative (FN) (type II error) 10
	condition negative	False Positive (FP) (Type I error) 30	True Negative (TN) 40

$$\text{Accuracy} = \frac{20 + 40}{20 + 10 + 30 + 40} = 0.6$$

Precision

- Out of the instances I predicted as “positive”, how many percentage of them are correct?
- $\text{Precision}(y, \hat{y}) = \frac{\sum I(y_i = \hat{y}_i = 1)}{\sum I(\hat{y}_i)} = \frac{\text{TP}}{\text{TP} + \text{FP}}$
 - Assuming a binary classification task
- Commonly used to evaluate the quality of a search engine
 - How useful the search results are

Example

		predicted condition	
total population		prediction positive	prediction negative
true condition	condition positive	True Positive (TP) 20	False Negative (FN) (type II error) 10
	condition negative	False Positive (FP) (Type I error) 30	True Negative (TN) 40

$$\text{Precision} = \frac{20}{20 + 30} = 0.4$$

Recall

- Out of all the truly positive instances, how many percentage I correctly predicted?
- $\text{Recall}(y, \hat{y}) = \frac{\sum I(y_i = \hat{y}_i = 1)}{\sum I(y_i)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$
- A.k.a.: true positive rate (TPR)
- How easy to evaluate recall of a search engine?

Example

		predicted condition	
total population		prediction positive	prediction negative
true condition	condition positive	True Positive (TP) 20	False Negative (FN) (type II error) 10
	condition negative	False Positive (FP) (Type I error) 30	True Negative (TN) 40

$$\text{Recall} = \frac{20}{20 + 10} = \frac{2}{3}$$

Precision and recall tradeoff

- If I want a very high precision
 - Return only the **most confident** positive instances (# returns is small)
- If I want a very high recall
 - Return all the instances (# returns is huge)
- The two metrics are usually a tradeoff

F1-score

- F1-score considers both precision (p) and recall (r)

- $$F_1(y, \hat{y}) = \frac{2}{\frac{1}{p} + \frac{1}{r}} = 2 \frac{pr}{p+r}$$

- Harmonic mean of p and r , i.e., $1 / \frac{1}{2} \left(\frac{1}{p} + \frac{1}{r} \right)$

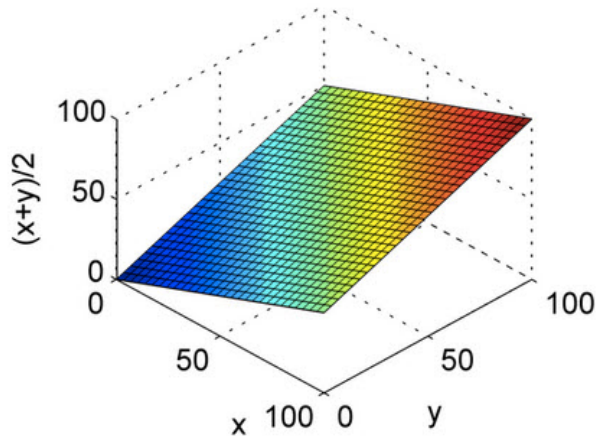
- General form (F_β score)

- $$F_\beta(y, \hat{y}) = (1 + \beta^2) \frac{pr}{\beta^2 p + r}$$

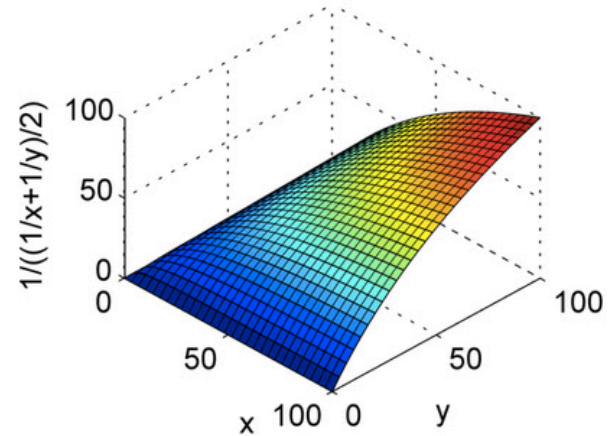
Why harmonic mean? – a numerical explanation

- If we have $n - 1$ negative samples and 1 positive sample, and a classifier returns everything as positive
 - When $n \rightarrow \infty$: Precision = $\frac{1}{n} \approx 0$, recall = 1
 - Arithmetic mean: 0.5
 - Harmonic mean: $\frac{1}{2} \left(\frac{1}{n} \times \frac{n-1}{n} \right) \approx 0$
 - Penalize the extreme values

Why harmonic mean? – a numerical explanation (cont')



Arithmetic mean



Harmonic mean

Why harmonic mean? – a theoretical explanation

- For the average to be valid, the values have to be in the same scaled units

Why harmonic mean? – a theoretical explanation (cont')

- Example: if a vehicle travels a certain distance d (e.g., 120km) outbound at a speed x (e.g., 60 km/h) and returns the same distance at a speed y (e.g., 20 km/h),
 - Average speed is the harmonic mean of x and y (30 km/h), not the arithmetic mean (40 km/h)
 - Km/h need to be compared over the same number of hours, not over the same number of kms

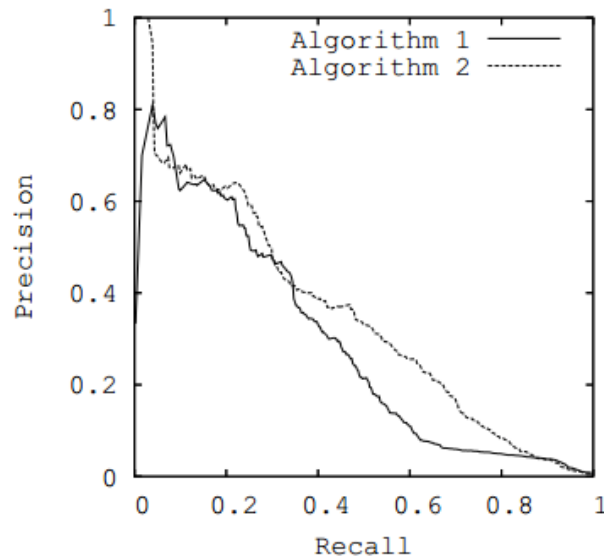
$$\frac{2d}{\frac{d}{x} + \frac{d}{y}} = \frac{2d}{\frac{d(x+y)}{xy}} = \frac{2xy}{x+y}$$

Why harmonic mean? – a theoretical explanation (cont')

- $P = \frac{TP}{TP+FP}$
- $R = \frac{TP}{TP+FN}$
- Arithmetic mean of the two are probably not reasonable
 - They are not compared over the unit
- Harmonic mean is probably more appropriate
 - As the semantic of the numerators are the same

Precision and recall curve

- Precision vs recall, as we vary the threshold of the “confidence”



Sensitivity and specificity

- These two terms are usually used in medical field
- If we define a positive case as “a person who has a disease”
 - Sensitivity: $\frac{TP}{TP+FN}$ (the same as recall)
 - The percentage of sick people being tested as positive
 - Specificity: $\frac{TN}{TN+FP}$
 - The percentage of healthy people being tested as negative

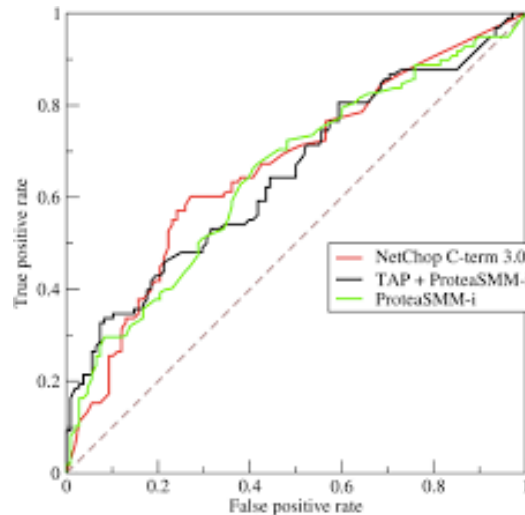
Example

		predicted condition	
total population		prediction positive	prediction negative
true condition	condition positive	True Positive (TP) 20	False Negative (FN) (type II error) 10
	condition negative	False Positive (FP) (Type I error) 30	True Negative (TN) 40

$$\text{Sensitivity} = \frac{20}{20+10} = \frac{2}{3}, \text{ Specificity} = \frac{40}{30+40} = \frac{4}{7}$$

ROC curve

- True positive rate (recall) vs false positive rate, as we vary the threshold of the “confidence”
- I personally prefer ROC curve over PR-curve



Precisions, recalls (TPRs), and FPRs of different thresholds

Seq	\hat{y}	y
1	0.95	1
2	0.93	1
3	0.91	0
4	0.88	0
5	0.60	1
6	0.33	0
7	0.07	0
8	0.04	1
9	0.03	0
10	0.01	0

← Accuracy: 6/10, precision: 0; TPR (recall): 0/4; FPR: 0/6

← Accuracy: 7/10, precision: 1/1; TPR (recall): 1/4; FPR: 0/6

← Accuracy: 8/10, precision: 2/2; TPR (recall): 2/4; FPR: 0/6

← Accuracy: 7/10, precision: 2/3; TPR (recall): 2/4; FPR: 1/6

← Accuracy: 6/10, precision: 2/4; TPR (recall): 2/4; FPR: 2/6

← Accuracy: 7/10, precision: 3/5; TPR (recall): 3/4; FPR: 2/6

← Accuracy: 6/10, precision: 3/6; TPR (recall): 3/4; FPR: 3/6

← Accuracy: 5/10, precision: 3/7; TPR (recall): 3/4; FPR: 4/6

← Accuracy: 6/10, precision: 4/8; TPR (recall): 4/4; FPR: 4/6

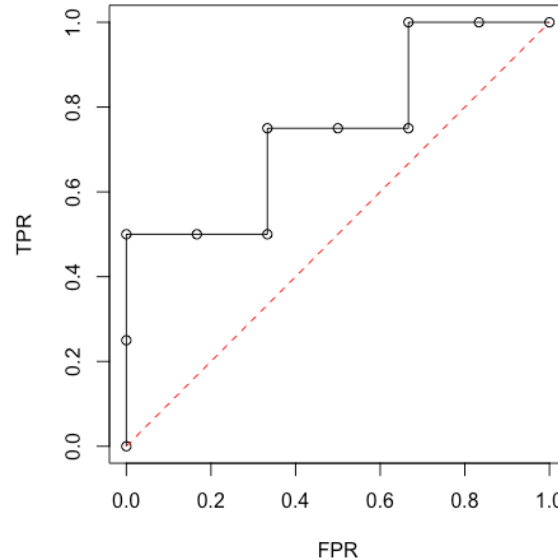
← Accuracy: 5/10, precision: 4/9; TPR (recall): 4/4; FPR: 5/6

← Accuracy: 4/10, precision: 4/10; TPR (recall): 4/4; FPR: 6/6

Plotting the ROC curve

Seq	\hat{y}	y
1	0.95	1
2	0.93	1
3	0.91	0
4	0.88	0
5	0.60	1
6	0.33	0
7	0.07	0
8	0.04	1
9	0.03	0
10	0.01	0

TPR	FPR
0/4	0/6
1/4	0/6
2/4	0/6
2/4	1/6
2/4	2/6
3/4	2/6
3/4	3/6
3/4	4/6
4/4	4/6
4/4	5/6
4/4	6/6



$$AUC = \frac{2}{4} * \frac{2}{6} + \frac{3}{4} * \left(\frac{4}{6} - \frac{2}{6} \right) + \frac{4}{4} * \left(\frac{6}{6} - \frac{4}{6} \right) = \frac{3}{4}$$

Properties of the ROC curve

- The diagonal line represents the expected result of random guess, with probability p predicting positive and probability $1-p$ predicting negative
 - $AUC = 0.5$
- Perfect condition
 - $(0,0) \rightarrow (0,1) \rightarrow (1,1)$
 - $AUC = 1.0$

Why diagonal line represents random guess

Note:

$$TPR = TP / (TP + FN)$$

$$FPR = FP / (FP + TN)$$

- If n instances, p' of them are truly positive, $1 - p'$ of them are negative
- A predictor performs random guess, with p predicting positive, $1 - p$ predicting negative
- For every k prediction, kp are predicted as positive on average
 - $E[TP] = kpp'$
 - $E[FP] = kp(1 - p')$
 - $E[TN] = k(1 - p)(1 - p')$
 - $E[FN] = k(1 - p)p'$
 - $E[TPR] = kpp' / (kpp' + k(1 - p)p') = p$
 - $E[FPR] = kp(1 - p') / (kp(1 - p') + k(1 - p)(1 - p')) = p$

Cross entropy loss of multiple-class classification

- Cross entropy loss:
 - $-\sum_i \sum_k p(y_i = k) \log[p(\hat{y}_i = k)]$
 - i : the index of the data instances
 - k : the k 'th class type
- Example: 3 classes
 - $p(\hat{y}_i = k) = [1/4 \quad 1/4 \quad 1/2]$
 - $p(y_i = k) = [0 \quad 1 \quad 0]$
 - $-\sum_k p(y_i = k) \log[p(\hat{y}_i = k)] = -\left[0 \log \frac{1}{4} + 1 \log \frac{1}{4} + 0 \log \frac{1}{2}\right] = 2$

Summary (1/2)

- Binary classification: encode y_i as 0/1 or $-1/1$
- The output of a logistic function is in $[0,1]$
- Logistic regression: find the parameters to fit a logistic function
- Apply l_k -norm on parameters to prevent overfitting
- Gradient ascent vs gradient descent

Summary (2/2)

- Accuracy, precision, recall, TPR, FPR, etc.
- ROC curve and area under ROC curve (AUROC)
- Evaluating multi-class classification by cross-entropy loss

Quiz

- What is accuracy?
- What is precision?
- What is recall?
- What are advantages and disadvantages of ROC curve and AUROC?