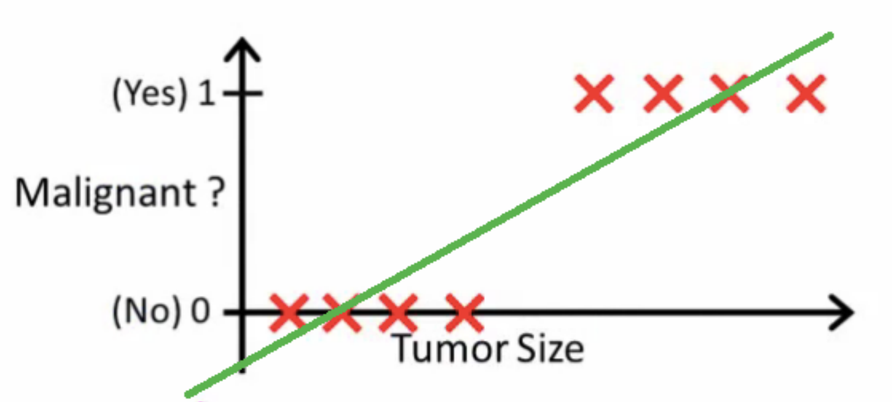
Logistic regression

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 $f(x) > 0.5$ ➔ Y, otherwise N

$$f(x) = \theta_0 + \theta_1 x$$

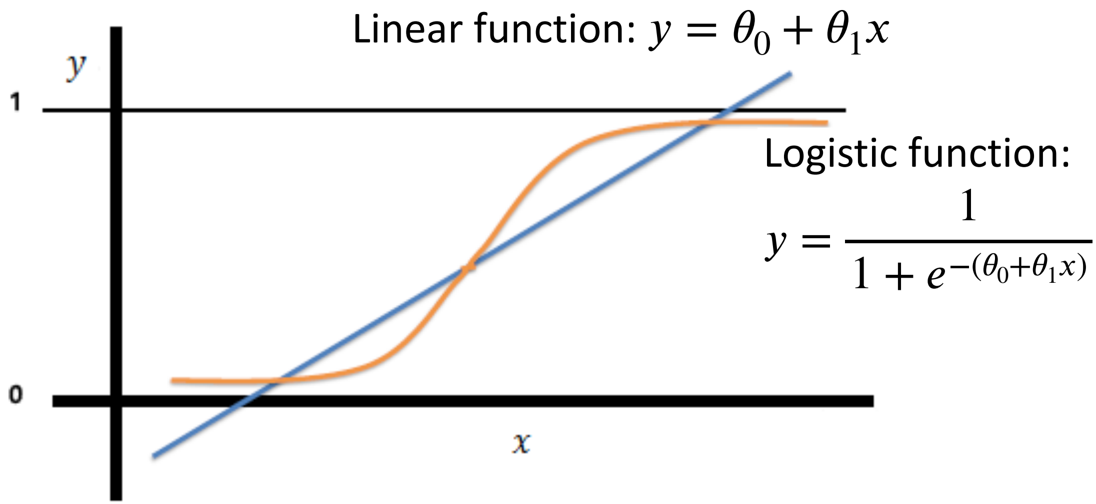


Linear regression is highly affected by the extreme values

- 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

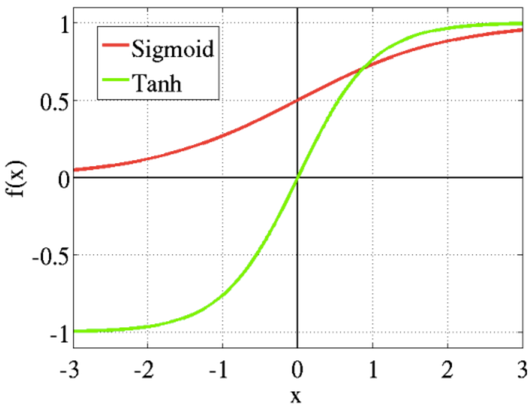


Sigmoid function

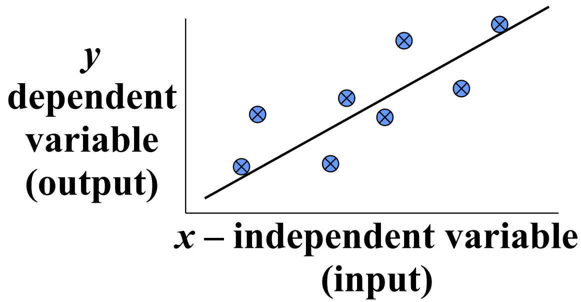
- A sigmoid function is a mathematical function having an "S" shaped curve (sigmoid curve)
  - Logistic function (the “classic” sigmoid function)
    - $f(x) = \dfrac{1}{1 + e^{-x}}$
  - Hyperbolic tangent function (a.k.a. tanh function)
    - $f(x) = \dfrac{1 - e^{-2x}}{1 + e^{-2x}}$

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

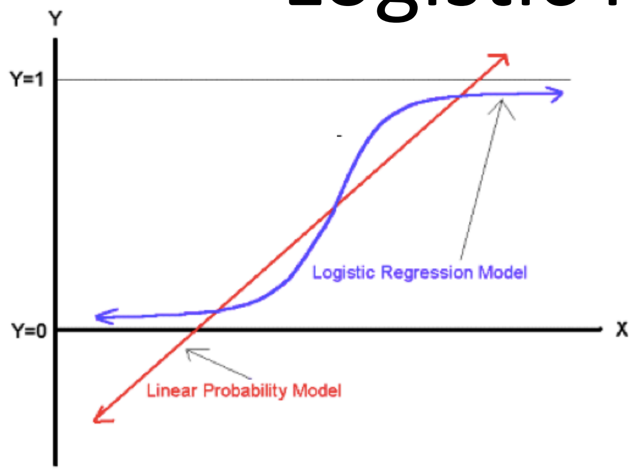


"Predictor":

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

- 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":

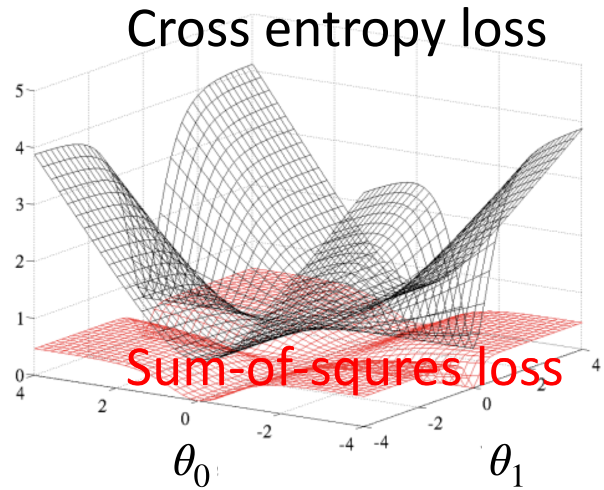
$$\hat{y} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

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

- Define form of function *f*(*x*) explicitly
  - i.e., $f(x) = \hat{y} = \dfrac{1}{1 + \mathrm{e}^{-(\theta_0 + \theta_1 x)}} = \dfrac{1}{1 + \mathrm{e}^{-\boldsymbol{\theta}^T \mathbf{x}}}$ 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

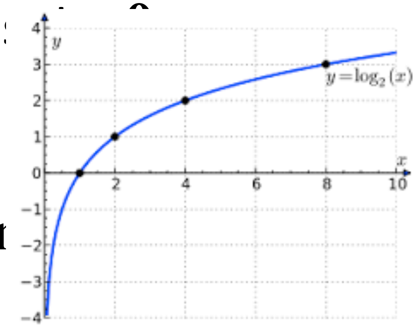
- The probability of the value of *y*
  - $P(y=1) = f(\mathbf{x}) = \dfrac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}$
  - $P(y=0) = 1 - f(\mathbf{x})$
  - ➔ $P(y) = f(\mathbf{x})^y \left(1 - f(\mathbf{x})\right)^{1-y}$
- If we have *n* independent training samples, the likelihood of the parameter $\boldsymbol{\theta}$ is
  - $L(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i) = \prod_{i=1}^{n} f(\mathbf{x}_i)^{y_i} (1 - f(\mathbf{x}_i))^{(1-y_i)}$
    - $\mathbf{x}_i$: the ith training instance (a vector); y_i: the ith training label (a scalar)

Likelihood and log likelihood

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

$$\boldsymbol{\theta} := \operatorname*{argmax}_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$
$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{n} f(\mathbf{x}_i)^{y_i}(1 - f(\mathbf{x}_i))^{(1-y_i)}$$

- The $\boldsymbol{\theta}$ to maximize the likelihood is the same a
  maximize the log-likelihood
  - Log function monotonically increasing



$$\boldsymbol{\theta} := \operatorname*{argmax}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta}} \log(L(\boldsymbol{\theta})) = \text{arg}$$
$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum \left\{ y_i \log\left(f(\mathbf{x}_i)\right) + (1 - y_i)\log(1 - f(\mathbf{x}_i)) \right\}$$

Why use log-likelihood?

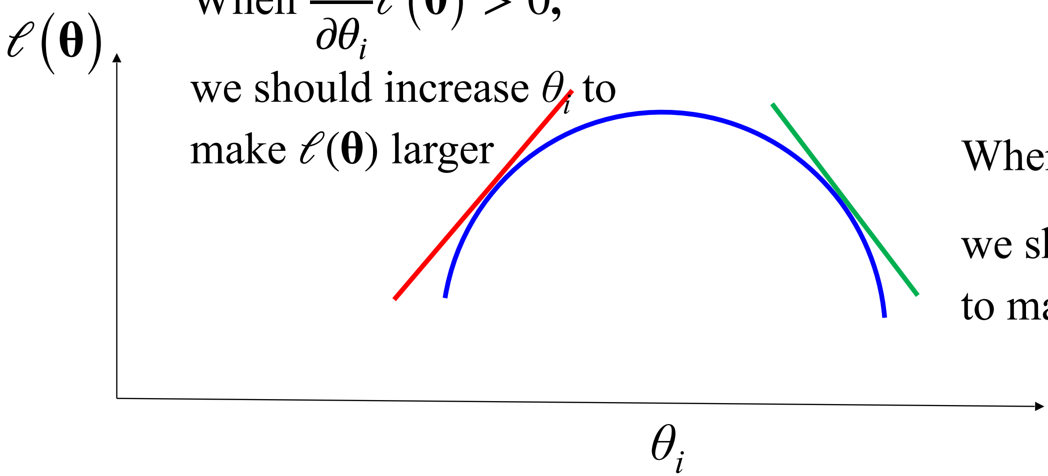
- Change multiplications to summations
  - Summations are faster than multiplications
  - Numerically, multiplying many tiny numbers tend to underflow; multiplying many huge numbers tend to overflow

How to find $\boldsymbol{\theta}$?

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

Gradient ascent

When $\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{\theta}) > 0$, we should increase θ_i to make $\ell(\boldsymbol{\theta})$ larger



When $\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{\theta}) < 0$, we should decrease θ_i to make $\ell(\boldsymbol{\theta})$ larger

Derivative of a logistic function

- $g(x) = \dfrac{1}{1+e^{-x}} = \left(1+e^{-x}\right)^{-1}$

$\rightarrow \quad g'(x) = -1\left(1+e^{-x}\right)^{-2} e^{-x}(-1) = \dfrac{e^{-x}}{(1+e^{-x})^2}$

$$= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right)$$

$$= g(x)\left(1 - g(x)\right)$$

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

$$g'(x) = g(x)\left(1 - g(x)\right)$$

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

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

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n}\left\{y_i\log\left(f(\mathbf{x}_i)\right) + (1-y_i)\log(1-f(\mathbf{x}_i))\right\} = \sum_{i=1}^{n}\left\{y_i\log\left(g(\boldsymbol{\theta}^T\mathbf{x}_i)\right) + (1-y_i)\log(1-g(\boldsymbol{\theta}^T\mathbf{x}_i))\right\}$$

$$\Rightarrow \frac{\partial\ell(\boldsymbol{\theta})}{\partial\theta_j} = \sum_{i=1}^{n}\left\{\frac{\partial y_i\log\left(g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)}{\partial\log\left(g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)}\cdot\frac{\partial\log\left(g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)}{\partial g(\boldsymbol{\theta}^T\mathbf{x}_i)}\cdot\frac{\partial g(\boldsymbol{\theta}^T\mathbf{x}_i)}{\partial\boldsymbol{\theta}^T\mathbf{x}_i}\cdot\frac{\partial\boldsymbol{\theta}^T\mathbf{x}_i}{\partial(\theta_j)}\right\}$$

$$+\sum_{i=1}^{n}\left\{\frac{\partial(1-y_i)\log\left(1-g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)}{\partial\log\left(1-g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)}\cdot\frac{\partial\log\left(1-g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)}{\partial\left(1-g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)}\cdot\frac{\partial(1-g(\boldsymbol{\theta}^T\mathbf{x}_i))}{\partial g(\boldsymbol{\theta}^T\mathbf{x}_i)}\cdot\frac{\partial g(\boldsymbol{\theta}^T\mathbf{x}_i)}{\partial\boldsymbol{\theta}^T\mathbf{x}_i}\cdot\frac{\partial\boldsymbol{\theta}^T\mathbf{x}_i}{\partial(\theta_j)}\right\}$$

$$=\sum_{i=1}^{n}\left\{y_i\cdot\frac{1}{g(\boldsymbol{\theta}^T\mathbf{x}_i)}\cdot g(\boldsymbol{\theta}^T\mathbf{x}_i)\cdot\left(1-g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)\cdot x_{ij}\right\}$$

$$+\sum_{i=1}^{n}\left\{(1-y_i)\cdot\frac{1}{1-g(\boldsymbol{\theta}^T\mathbf{x}_i)}\cdot(-1)\cdot g(\boldsymbol{\theta}^T\mathbf{x}_i)\cdot\left(1-g(\boldsymbol{\theta}^T\mathbf{x}_i)\right)\cdot x_{ij}\right\}$$

$$=\sum_{i=1}^{n}\left\{x_{ij}\left(y_i-\hat{y}_i\right)\right\}$$

Using gradient ascend to find $\boldsymbol{\theta}$

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

- Gradient ascent algorithm:

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

```
Repeat until converge {
    ŷ = 1/(1 + e^(−Xθ))
    θ^(k+1) := θ^(k) + αX^T(y − ŷ)
}
```

Cross entropy and log-likelihood of classification problem

- Maximize the log-likelihood function
  - $$\boldsymbol{\theta} = \arg\max\left(\sum \left\{ y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log(1 - \hat{y}_i) \right\} \right)$$

- This is the same as minimizing the negative of the log-likelihood function
  - $$\boldsymbol{\theta} = \arg\min\left(-\sum \left\{ y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log(1 - \hat{y}_i) \right\} \right)$$
  - This is called the "**cross entropy loss**" function

Cross entropy loss

- **Cross entropy loss** function
  - $\ell = -\sum \left\{ y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log(1 - \hat{y}_i) \right\}$, where
    - $\hat{y}_i = \dfrac{1}{1 + \mathrm{e}^{-(\theta_0 + \theta_1 x_i)}}$
  - $\boldsymbol{\theta} = \mathrm{argmin}\left(-\sum \left\{ y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log(1 - \hat{y}_i) \right\} \right)$
- Check:
  - If $y_i = 0$ and $\hat{y}_i \to 0$: $\ell \to 0$
  - If $y_i = 0$ and $\hat{y}_i \to 1$: $\ell \to \infty$
  - If $y_i = 1$ and $\hat{y}_i \to 0$: $\ell \to \infty$
  - If $y_i = 1$ and $\hat{y}_i \to 1$: $\ell \to 0$

Regularization to avoid overfitting

- Original goal:

$$\boldsymbol{\theta} := \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$$
$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum \left\{ y_i \log\left(f(\mathbf{x}_i) \right) + (1 - y_i) \log(1 - f(\mathbf{x}_i)) \right\}$$

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

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

  – **Penalize high weights, like we did in linear regression!**

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

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

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

Two different forms of cross entropy loss

- Form 1:
  - $$\ell = -\sum_i \left(y_i \log \hat{y}_i + (1 - y_i) \log \left(1 - \hat{y}_i \right) \right)$$
- Form 2:
  - $$\ell = \sum_i \log \left(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i} \right)$$
- Form 1: when encoding targets as 1/0
- Form 2: when encoding targets as 1/-1

Cross entropy loss with +1/-1 as classes

Times $e^{\theta^T \boldsymbol{x}}$ to both numerator and denominator

- $P(y=1) = \frac{1}{1+e^{-\theta^T \boldsymbol{x}}}$
- $P(y=-1) = 1 - P(y=1) = \frac{e^{-\theta^T \boldsymbol{x}}}{1+e^{-\theta^T \boldsymbol{x}}} = \frac{1}{1+e^{\theta^T \boldsymbol{x}}}$
- $P(y) = \left(\frac{1}{1+e^{-\theta^T \boldsymbol{x}}}\right)^{\frac{y+1}{2}} \left(\frac{1}{1+e^{\theta^T \boldsymbol{x}}}\right)^{\frac{1-y}{2}}$
- $\log P(y) = \frac{y+1}{2}\left(-\log\left(1+e^{-\theta^T \boldsymbol{x}}\right)\right) + \frac{1-y}{2}\left(-\log\left(1+e^{\theta^T \boldsymbol{x}}\right)\right)$
  - When $y=1, \log P(y) = -\log\left(1+e^{-\theta^T \boldsymbol{x}}\right)$
  - When $y=-1, \log P(y) = -\log\left(1+e^{\theta^T \boldsymbol{x}}\right)$
  - → $\log P(y) = -\log\left(1+e^{-y\theta^T \boldsymbol{x}}\right)$
- Cross entropy loss is negative log-likelihood
  - $\ell = \log\left(1+e^{-y\theta^T \boldsymbol{x}}\right)$
  - A.k.a. logistic loss

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

- $\text{Accuracy}\left(y,\ \hat{y}\right) = \frac{1}{n}\sum I(y_i = \hat{y}_i)$
- 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?
- $$\text{Precision}\left(y, \hat{y}\right) = \frac{\sum I(y_i = \hat{y}_i = 1)}{\sum I(\hat{y}_i)} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
  – 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?
- $$\text{Recall}\left(y, \hat{y}\right) = \frac{\sum I(y_i = \hat{y}_i = 1)}{\sum I(y_i)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
- A.k.a.: true positive rate (TPR)
- How easy to evaluate recall of a search engine?

ex: covid need high recall

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

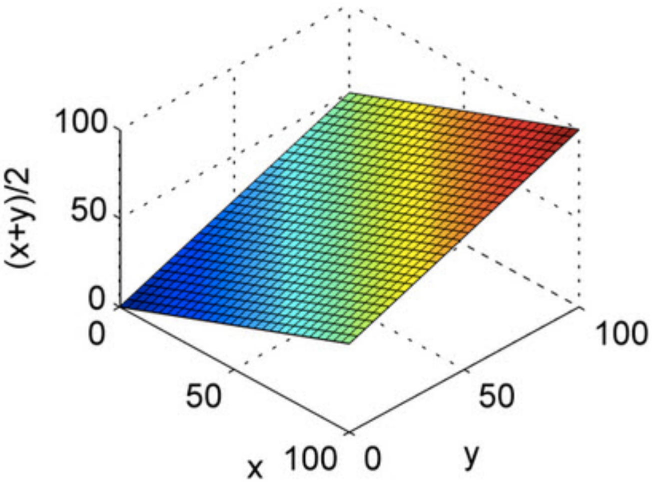
ex

	P	R
M1	0.5	0.5
M2	0	1

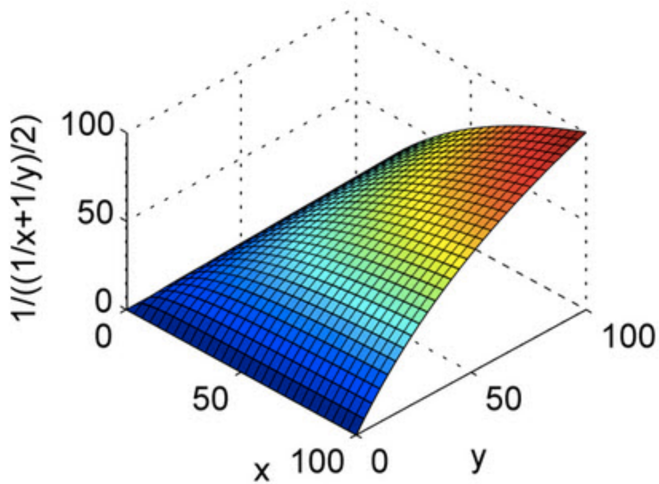
Why harmonic mean? – a numerical explanation

- If we have $n-1$ negative samples and 1 positive sample, and a classifier returns everything as positive
  - ➢ When $n \to \infty$: $\text{Precision} = \frac{1}{n} \approx 0, \text{recall} = 1$
  - ➢ 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)
  - Returns the same distance at a speed *y* (e.g., 20 km/h)
- Average speed is not arithmetic mean of *x* and *y* (40 km/h)
- Average speed should be harmonic mean (30 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')

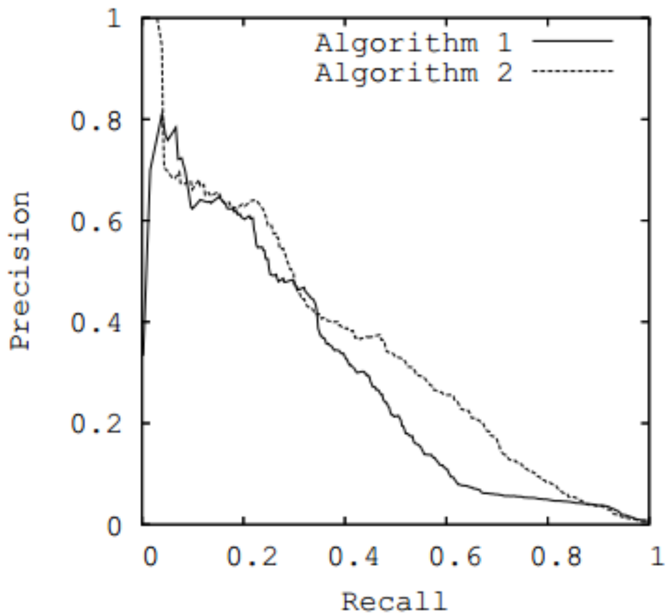
- $P = \dfrac{TP}{TP + FP}$
- $R = \dfrac{TP}{TP + FN}$

$$\rightarrow \frac{1}{P} + \frac{1}{R} = \underbrace{\frac{TP+FP}{TP} + \frac{TP+FN}{TP}}$$

- Arithmetic mean of the two are probably not reasonable
  - They are not compared over the same 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

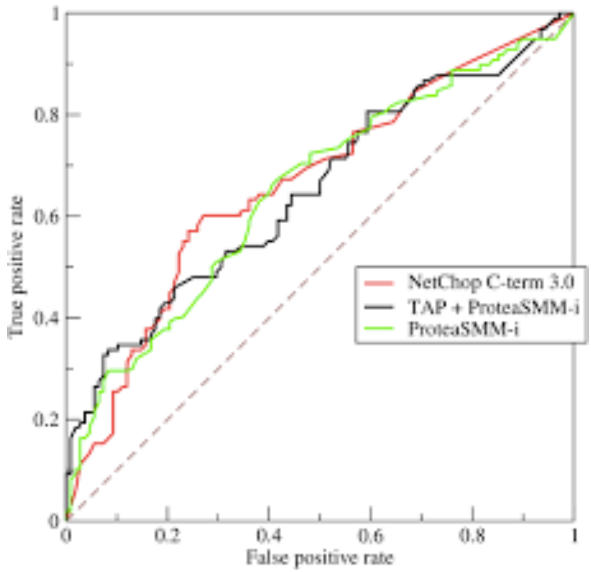
Example

		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		
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

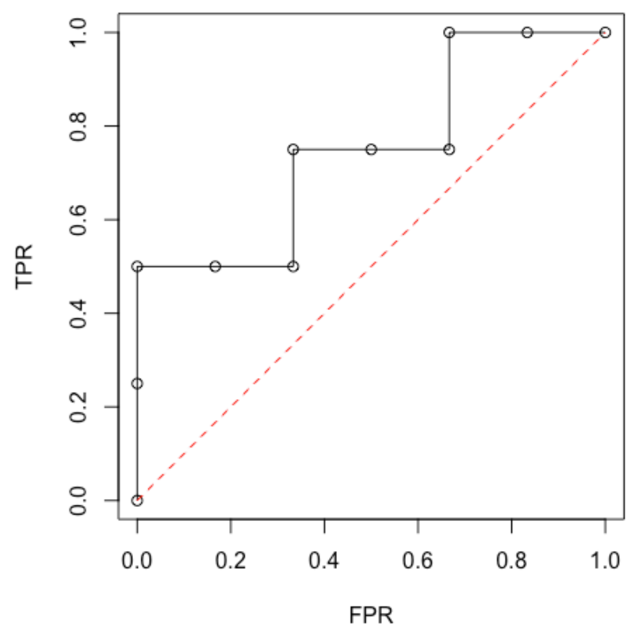
consider

Plotting the ROC curve

1. all cases
2. best AUC = 1

Seq		
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



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

der ROC curve

area under ROC curve

class 2

10%

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) ➔ (0,1) ➔ (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
  - $E[TP] = kpp'$
  - $E[FP] = kp\left(1 - p'\right)$
  - $E[TN] = k\left(1 - p\right)\left(1 - p'\right)$
  - $E[FN] = k\left(1 - p\right)p'$
  - $E[TPR] = kpp' / \left(kpp' + k\left(1 - p\right)p'\right) = p$
  - $E[FPR] = kp\left(1 - p'\right) / \left(kp\left(1 - p'\right) + k\left(1 - p\right)\left(1 - p'\right)\right) = p$

Quiz

- Can we apply logistic regressio on multi-class classification problem?

Using binary classifier for multi-class classification

- One-vs.-rest (aka one-vs-all)
- One-vs.-one

One-vs.-rest (aka: one-vs.-all)

- Train a single classifier for each class
- E.g.,
  - Target labels: “red”, “blue”, or “green”
  - Train three binary classifiers
    - f_1: ”Red” vs “not red”
    - f_2: “Blue” vs “not blue”
    - f_3: “Green” vs “not green”
  - $\hat{y}_i = \arg \max_{k \in \{1,2,3\}} f_k(\mathbf{x}_i)$

One-vs.-one

- Training: for a k-nary classification problem, one trains $C(k, 2)$ classifiers
- Test:
  - Feed the test instance to all $C(k, 2)$ classifiers
  - The class receiving the most "+1" predictions is the predicted class

Metrics for multiclass classification

- Cross entropy loss
- Accuracy
- Confusion matrix
- Precision
- Recall
- F1 score

Cross entropy loss of multiple-class classification

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

Accuracy

- $\text{Accuracy}\left(y,\ \hat{y}\right) = \frac{1}{n}\sum I(y_i = \hat{y}_i)$

Confusion matrix

- We want large values on the diagonal grids
- We want 0s on the other grids

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

Precision (1/2)

- Macro-precision
  - Treat one class as positive; the others as negatives, compute precision
  - Repeat the above step for every class
  - Compute the average
- Example
  - $\text{Precision}(Apple) = \dfrac{7}{7+8+9} \approx 0.29$
  - $\text{Precision}(Orange) = \dfrac{2}{1+2+3} \approx 0.33$
  - $\text{Precision}(Mango) = \dfrac{1}{3+2+1} \approx 0.17$
  - $\text{MacroPrecision} = \dfrac{.29+.33+.17}{3} \approx 0.26$

- Micro-precision
  - Treat one class i as positive; the others as negatives, compute TP_i and FP_i
  - Repeat the above step for every class
  - $\text{MicroPrecision} = \frac{\sum_i TP}{\sum_j (TP_j + FP_j)}$
- Example
  - $TP_{Apple} = 7, FP_{Apple} = 8 + 9 = 17$
  - $TP_{Orange} = 2, FP_{Orange} = 1 + 3 = 4$
  - $TP_{Mango} = 1, FP_{Mango} = 3 + 2 = 5$
  - $\text{MicroPrecision} = \frac{7+2+1}{7+17+2+4+1+5} \approx 0.28$

Precision (2/2)

- Micro-precision
  - Treat one class i as positive; the others as negatives, compute TP_i and FP_i
  - Repeat the above step for every class
  - $$\text{MicroPrecision} = \frac{\sum_i TP}{\sum_j \left(TP_j + FP_j\right)}$$
- Example
  - $TP_{Apple} = 7,\ FP_{Apple} = 8 + 9 = 17$
  - $TP_{Orange} = 2,\ FP_{Orange} = 1 + 3 = 4$
  - $TP_{Mango} = 1,\ FP_{Mango} = 3 + 2 = 5$
  - $$\text{MicroPrecision} = \frac{7 + 2 + 1}{7 + 17 + 2 + 4 + 1 + 5} \approx 0.28$$

Recall (1/2)

Predicted Class \ True Class	Apple	Orange	Mango
Apple	7	8	9
Orange	1	2	3
Mango	3	2	1

- Macro-recall
  - Treat one class as positive; the others as negatives, compute recall
  - Repeat the above step for every class
  - Compute the average
- Example
  - $\text{Recall}\left(Apple\right) = \dfrac{7}{7+1+3} \approx 0.64$
  - $\text{Recall}\left(Orange\right) = \dfrac{2}{8+2+2} \approx 0.17$
  - $\text{Recall}\left(Mango\right) = \dfrac{1}{9+3+1} \approx 0.08$
  - $\text{MacroRecall} = \dfrac{.64+.17+.08}{3} \approx 0.30$

Recall (2/2)

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

- Micro-recall
  - Treat one class i as positive; the others as negatives, compute TP_i and FN_i
  - Repeat the above step for every class
  - $$\text{MicroPrecision} = \frac{\sum_i TP}{\sum_j \left(TP_j + FN_j\right)}$$
- Example
  - $TP_{Apple} = 7,\ FN_{Apple} = 1 + 3 = 4$
  - $TP_{Orange} = 2,\ FN_{Orange} = 8 + 2 = 10$
  - $TP_{Mango} = 1,\ FN_{Mango} = 9 + 3 = 12$
  - $$\text{MicroPrecision} = \frac{7 + 2 + 1}{7 + 4 + 2 + 10 + 1 + 12} \approx 0.28$$

F1 score

- MacroF1
- MicroF1
- WeightedF1
- SamplesF1

MacroF1 (1/2)

Predicted Class \ True Class	Apple	Orange	Mango
Apple	7	8	9
Orange	1	2	3
Mango	3	2	1

- Two different definitions
- Definition 1

$$\_ \text{ MacroPrecision} = \frac{.29 + .33 + .17}{3} \approx 0.26$$

$$\_ \text{ MacroRecall} = \frac{.64 + .17 + .08}{3} \approx 0.30$$

$$\_ \text{ MacroF1} = \frac{2 * \text{MacroPrecision} * \text{MacroRecall}}{\text{MacroPrecision} + \text{MacroRecall}} \approx 0.28$$

MacroF1 (2/2)

Predicted Class \ True Class	Apple	Orange	Mango
Apple	7	8	9
Orange	1	2	3
Mango	3	2	1

Class	Precision	Recall	F1
Apple	0.29	0.64	0.40
Orange	0.33	0.17	0.22
Mango	0.17	0.08	0.11

- Definition 2
  - $\text{MacroF1} = \frac{1}{3}(0.40 + 0.22 + 0.11) = 0.24$
- Sklearn’s MacroF1 is defined by Definition 2

MicroF1

		True Class		
		Apple	Orange	Mango
Predicted Class	Apple	7	8	9
	Orange	1	2	3
	Mango	3	2	1

- $\text{MicroPrecision} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K}(TP_k + FP_k)} = \frac{7+2+1}{7+17+2+4+1+5} \approx 0.28$
- $\text{MicroRecall} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K}(TP_k + FN_k)} = \frac{7+2+1}{7+4+2+10+1+12} \approx 0.28$
- $\text{MicroF1} = \frac{2 * \text{MicroPrecision} * \text{MicroRecall}}{\text{MicroPrecision} + \text{MicroRecall}} \approx 0.28$
- For a multi-class classification problem, $\text{MicroPrecision} = \text{MacroRecall}$
  - This is because FP of one class must be the FN of another class
  - E.g., $FP_{Apple} = 8 + 9$
    - 8 is part of FN_{Orange}
    - 9 is part of FN_{Mango}

Weighted F1

Predicted Class \ True Class	Apple	Orange	Mango
Apple	7	8	9
Orange	1	2	3
Mango	3	2	1

Class	Precision	Recall	F1
Apple	0.29	0.64	0.40
Orange	0.33	0.17	0.22
Mango	0.17	0.08	0.11

- 11 Apple, 12 Orange, 13 Mango
- $\text{WeightedF1} = \dfrac{11 * 0.4 + 12 * 0.22 + 13 * 0.11}{11 + 12 + 13} = 0.24$

Multi-label classification

- Predicting zero or more class labels for each instance
- Example: possible labels include ‘A’, ‘B’, and ‘C’

#	Truth	Prediction
1	B	B, C
2	B, C	B, C
3	A, C	B
4	C	empty

Samples F1 score

- Compute precision, recall, and F1 for “each instance”
- Compute the average over the instances

Example of samples F1 score (1/3)

#	Truth	Pred
1	B	B, C
2	B, C	B, C
3	A, C	B
4	C	''
5	''	A

- Sample 1 ($S1$):
  - $Prec(S1) = \frac{\left|\text{Pred} \cap \text{Truth}\right|}{\left|\text{Pred}\right|} = \frac{1}{2}$
  - $Rec(S1) = \frac{\left|\text{Pred} \cap \text{Truth}\right|}{\left|\text{Truth}\right|} = 1$
  - $F1(S1) = \frac{2 \times \frac{1}{2} \times 1}{\frac{1}{2} + 1} = \frac{2}{3}$
- Sample 2 ($S2$):
  - $Prec(S2) = \frac{\left|\text{Pred} \cap \text{Truth}\right|}{\left|\text{Pred}\right|} =$
  - $Rec(S2) = \frac{\left|\text{Pred} \cap \text{Truth}\right|}{\left|\text{Truth}\right|} = 1$
  - $F1(S2) = \frac{2 \times 1 \times 1}{1 + 1} = 1$

Example of samples F1 score (2/3)

#	Truth	Pred
1	B	B, C
2	B, C	B, C
3	A, C	B
4	C	“
5	“	A

- Sample 3 ($S3$):
  - $Prec(S3) = \frac{|\text{Pred} \cap \text{Truth}|}{|\text{Pred}|} = 0$
  - $Rec(S3) = \frac{|\text{Pred} \cap \text{Truth}|}{|\text{Truth}|} = 0$
  - $F1(S3) = 0$
- Sample 4 ($S4$):
  - $Prec(S4) = \frac{|\text{Pred} \cap \text{Truth}|}{|\text{Pred}|} = 0$
  - $Rec(S4) = \frac{|\text{Pred} \cap \text{Truth}|}{|\text{Truth}|} = 0$
  - $F1(S4) = 0$
- Sample 5 ($S5$):
  - $Prec\left(S5\right) = \frac{|\text{Pred} \cap \text{Truth}|}{|\text{Pred}|} = 0$
  - $Rec\left(S5\right) = \frac{|\text{Pred} \cap \text{Truth}|}{|\text{Truth}|} = 0$
  - $F1\left(S5\right) = 0$

Example of samples F1 score (3/3)

#	Truth	Pred
1	B	B, C
2	B, C	B, C
3	A, C	B
4	C	‘’
5	‘’	A

#	Precision	Recall	F1 score
1	1/2	1	2/3
2	1	1	1
3	0	0	0
4	0	0	0
5	0	0	0
Samples avg	(1/2 + 1) / 5 = 0.3	(1 + 1) / 5 = 0.4	(2/3 + 1) / 5 = 0.333

Problematic case

#	Truth	Pred
1	B	B, C
2	B, C	B, C
3	A, C	B
4	C	“
5	“	A
6	“	“

如果用sklib
会使Pr、Rc、F1
低

- If truth is “, prediction is “, this should be a correct prediction
- However, adding this case decreases the samples precision/recall/F1 scores

#	Precision	Recall	F1 score
1	1/2	1	2/3
2	1	1	1
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
Samples avg	(1/2 + 1) / 6 = 0.25	(1 + 1) / 6 = 0.333	(2/3 + 1) / 6 = 0.278

Using sklearn to check the above case

```
from sklearn.metrics import classification_report
y_true = np.array([[0,1,0],
                   [0,1,1],
                   [1,0,1],
                   [0,0,1],
                   [0,0,0],
                   [0,0,0]])
y_pred = np.array([[0,1,1],
                   [0,1,1],
                   [0,1,0],
                   [0,0,0],
                   [1,0,0],
                   [0,0,0]])
print(classification_report(y_true, y_pred, digits=3))
```

	precision	recall	f1-score	support
0	0.000	0.000	0.000	1
1	0.667	1.000	0.800	2
2	0.500	0.333	0.400	3
micro avg	0.500	0.500	0.500	6
macro avg	0.389	0.444	0.400	6
weighted avg	0.472	0.500	0.467	6
samples avg	0.250	0.333	0.278	6

Any suggested improvement?

Example of improved samples F1 score

- Key: treat empty as a new symbol (e.g., χ)

#	Truth	Pred
1	B	B, C
2	B, C	B, C
3	A, C	B
4	C	″
5	″	A
6	″	″

#	Truth	Pred
1	B	B, C
2	B, C	B, C
3	A, C	B
4	C	
5		A
6		

- Sample 4 ($S4$):
  - $Prec(S4) = \dfrac{|\text{Pred} \cap \text{Truth}|}{|\text{Pred}|} = \dfrac{0}{1}$
  - $Rec(S1) = \dfrac{|\text{Pred} \cap \text{Truth}|}{|\text{Truth}|} = \dfrac{0}{1}$
  - $F1(S1) = 0$
- Sample 5 ($S5$):
  - $Prec(S5) = \dfrac{|\text{Pred} \cap \text{Truth}|}{|\text{Pred}|} = \dfrac{0}{1}$
  - $Rec(S5) = \dfrac{|\text{Pred} \cap \text{Truth}|}{|\text{Truth}|} = \dfrac{0}{1}$
  - $F1(S5) = 0$
- Sample 6 ($S6$):
  - $Prec(S6) = \dfrac{|\text{Pred} \cap \text{Truth}|}{|\text{Pred}|} = \dfrac{1}{1}$
  - $Rec(S6) = \dfrac{|\text{Pred} \cap \text{Truth}|}{|\text{Truth}|} = \dfrac{1}{1}$
  - $F1(S6) = 1$

I don’t see anyone discuss this issue in any literature so far

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

Quiz

- What is accuracy?
- What is precision?
- What is recall?
- What are advantages and disadvantages of ROC curve and AUROC?
- How to apply logistic regression on multi-class classification problems?
- Explain the differences between “multi-class classification” and “multi-label classification”

Quiz

- Show the MacroF1, MicroF1, and WeightedF1 of the following experimental results

Predicted

True labels	A	B	C	
A	2	2	0	4
B	1	2	0	3
C	0	0	3	3
	3	4	3	Total

A+: $P_A = \frac{2}{3}$, $R_A = \frac{2}{4}$, $F_A = \frac{2P_A \cdot R_A}{P_A + R_A}$

B+: $P_B = \frac{2}{4}$ $R_B = \frac{2}{3}$ $F_B = \frac{2P_B \cdot R_B}{P_B + R_B}$

C+: $P_C = \frac{3}{3}$, $R_C = \frac{3}{3}$, $F_C = 1$

$$\text{Macro } F_1 = \frac{1}{3}(F_A + F_B + F_C)$$

有很多版本
sklearn用此
版本

$$\text{Weighte } F_1 = \frac{4}{10}F_A + \frac{3}{10}F_B + \frac{3}{10}F_C$$