Machine Learning Principles

Class10: Oct 6

Linear Classification: Data Imbalance and Metrics

Instructor: Diana Kim

Today's Lecture

1. Data imbalance and its effect on classification and its performance

2. Classification Metrics
- accuracy
- precision & recall
- F1 score
- ROC curve (Receiver Operating Characteristic)

3. Strategies for Data Imbalance in Training

Why do we care about data imbalance?

- when a class is minority
  a regular training method might not learn the class well.

- data imbalance can lead to deceptively high performance
  for a biased classifier.

- Data Imbalance in ML

  : class populations are often imbalanced

[1] Data Imbalance in ML (example1)

- data imbalance is a natural occurrence

  (1) medical diagnosis: breast cancer vs. normal (13% vs. 87% in 2025)
  diabetes vs. normal (11.6 vs. 88.4% in 2021)

  (2) card fraudulent transaction: fraud vs. normal (0.17 vs. 99.83%)

  (3) expected imbalance but not: spam vs. ham (46% vs. 54%)

[2] Data Imbalance in ML (example2)

- fine art paintings & styles (WikiArt)

https://www.kaggle.com/datasets/steubk/wikiart

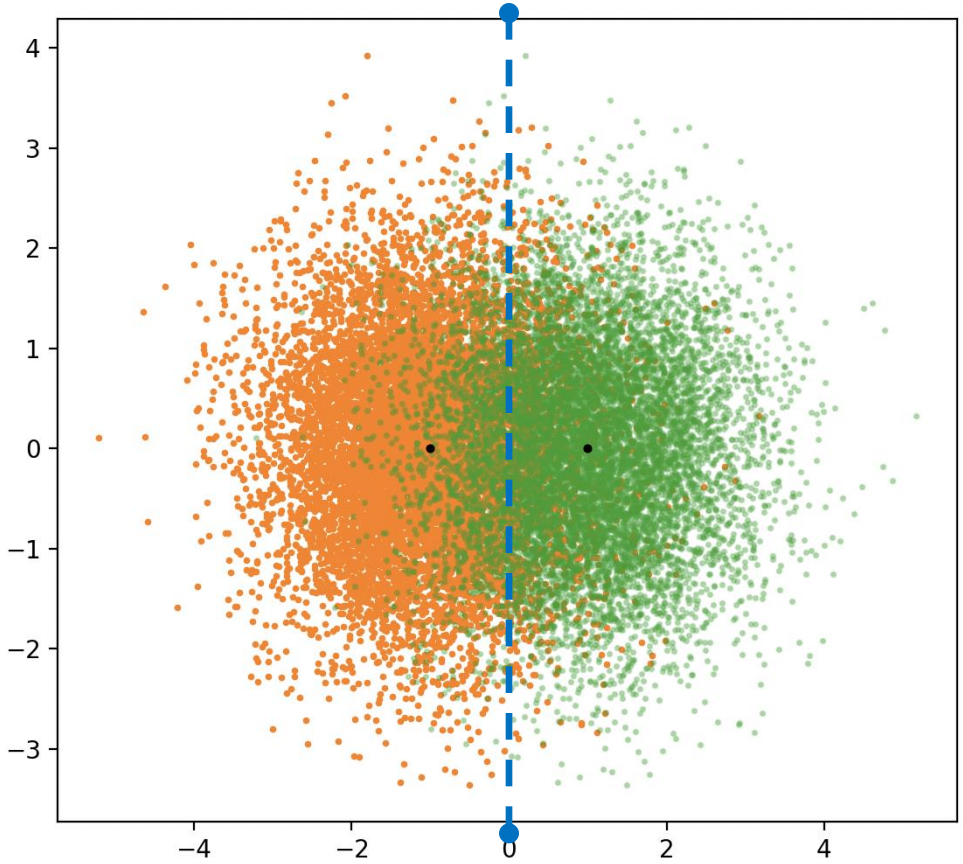
- action painting

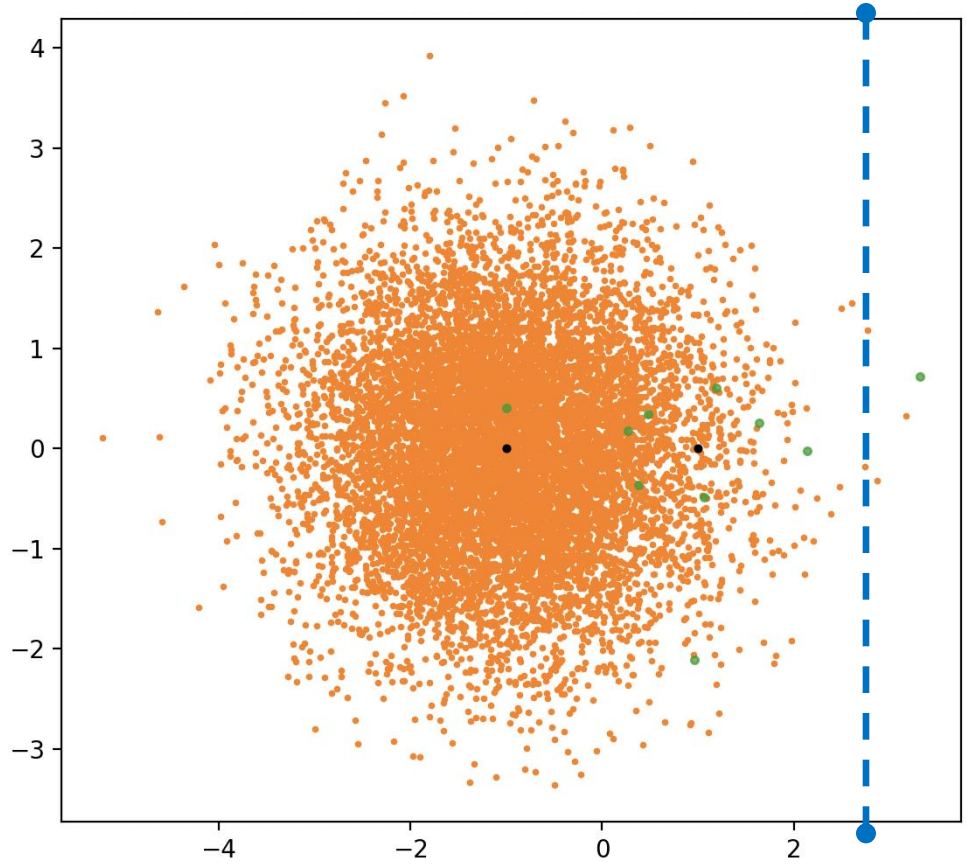
- impressionist

- Overfitting by Data Imbalance

  : a trained model is biased by data imbalance.

[1] Overfitting by Data Imbalance (bias toward majority class)





Q: when we trained a logistic regression to classify orange vs. green dots what would the trained decision boundary look like?

[2] Overfitting by Data Imbalance (weighting effect)

- the biased population leads to weighting more on majority class.

$$J(\vec{w}) = -\ln P(t_1, ..., t_N | w) = \sum_{n \in g} -t_n \ln \sigma(w^t x_n) - (1 - t_n) \ln (1 - \sigma(w^t x_n))$$
$$+ \sum_{n \in o} -t_n \ln \sigma(w^t x_n) - (1 - t_n) \ln (1 - \sigma(w^t x_n))$$

- when overlapping occurs, classifying as the major class minimizes the loss more effectively. this results in a biased classifier working well on majority but poor on minority.

[3] Overfitting by Data Imbalance (weighting effect)

- The poor performance on minority is critical
  because general classification tasks focus on recognizing these minor classes.

[4] Overfitting by Data Imbalance (overfitting)

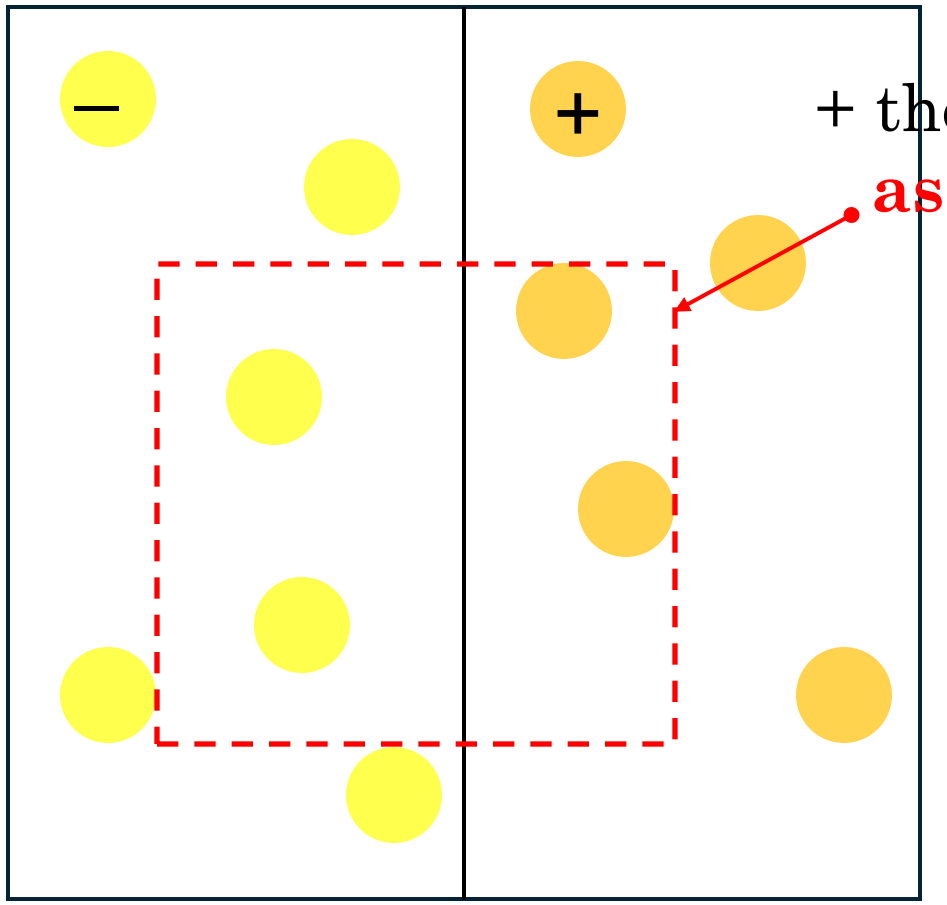
- "overfitting" in the context of data imbalance
  differ from the concept of overfitting we previously learned.

	overfitting	overfitting
causes	high model complexity relative to # data points	data imbalance (pos>>neg / pos<<neg)
symptoms	high performance on training but low on test	good performance overall classes but poor on minority class
main issues	poor generalization to new data	poor generalization to minority class
solutions	regularization/ more data	resampling / reweights

- Classification Evaluation & Metrics

  accuracy, precision, recall, F1 score

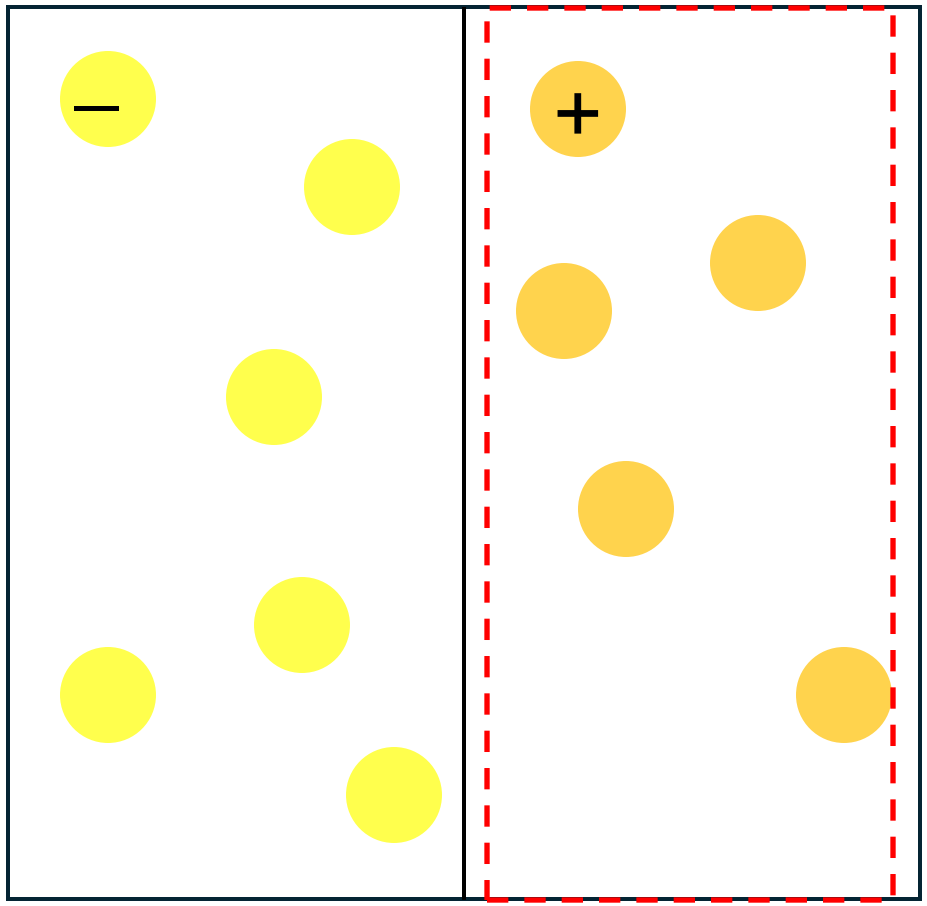
[1] Classification Evaluation (TP/FP/TN/FN)



+ the points inside of the rectangular is classified **as positive.**

- true positive TP: (P | P)
- false positive FP: (P | N)
- true negative TN: (N | N)
- false negative FN: (N | P)

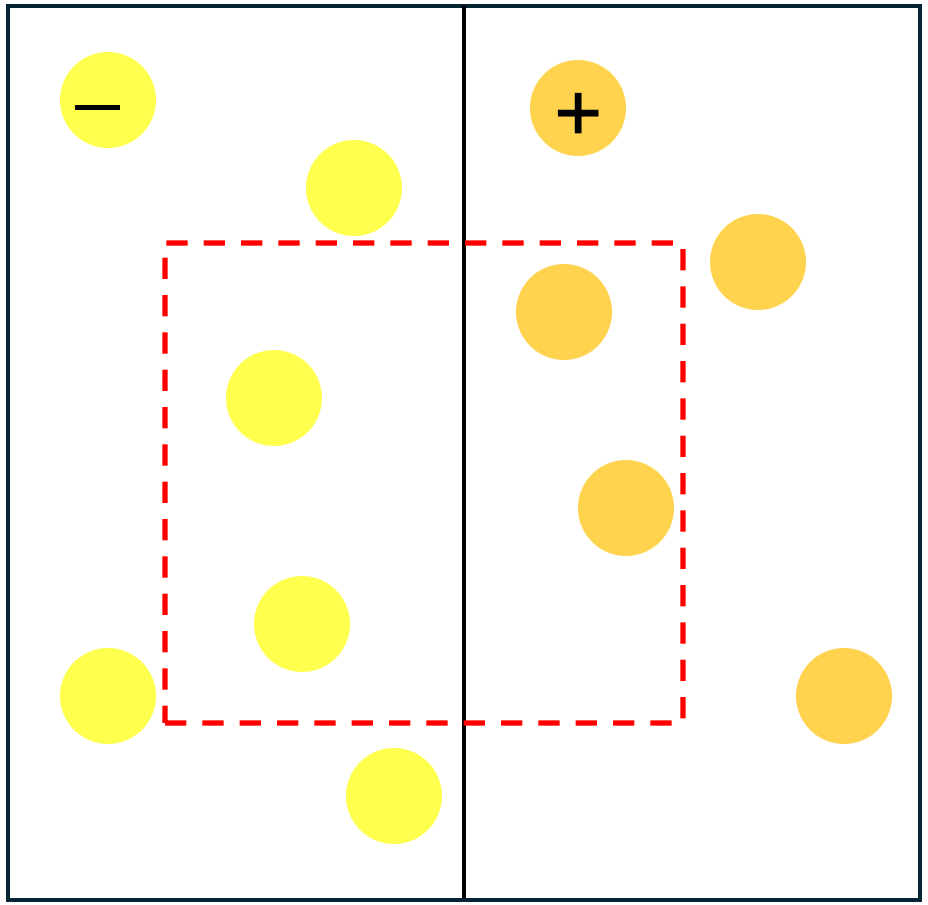
[2] Classification Evaluation (ideal case)



- true positive (P | P)
- false positive (P | N)
- true negative (N | N)
- false negative (N | P)

[the case of 100% accuracy]

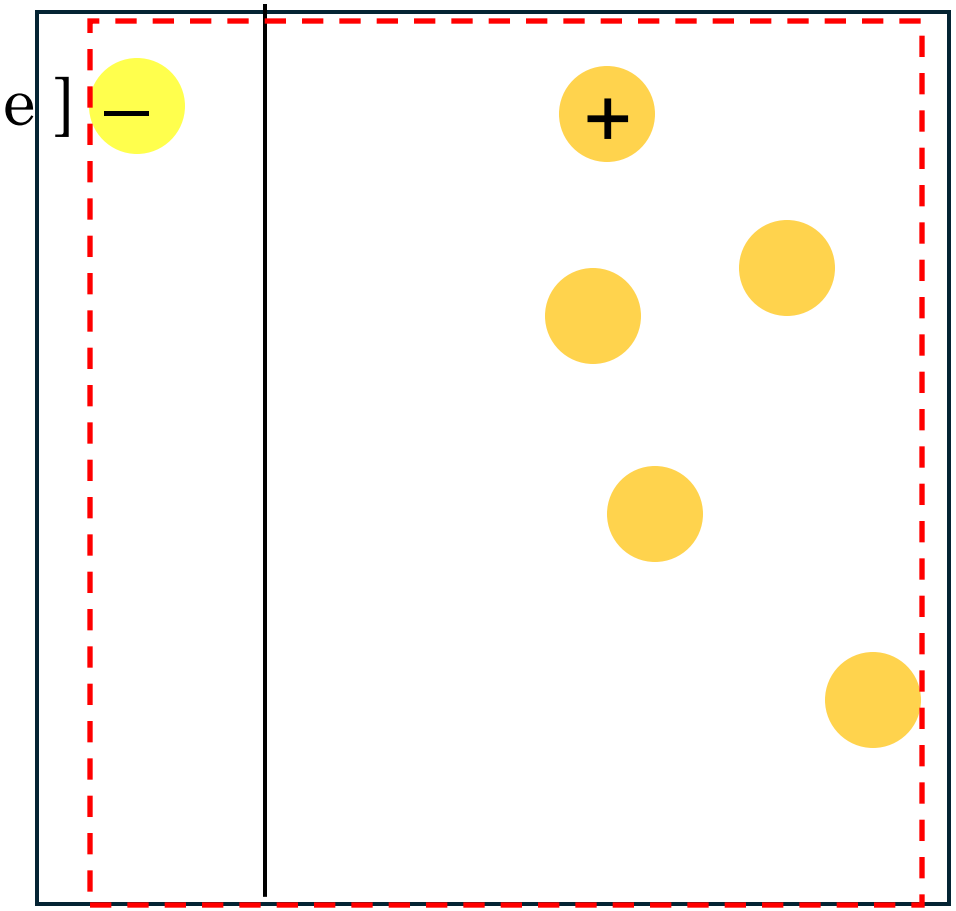
[3] Classification Evaluation (accuracy)



- Accuracy = $\frac{[TP] + [TN]}{\#\text{ all data points}}$

[4] Classification Evaluation (data imbalance pos>>neg)

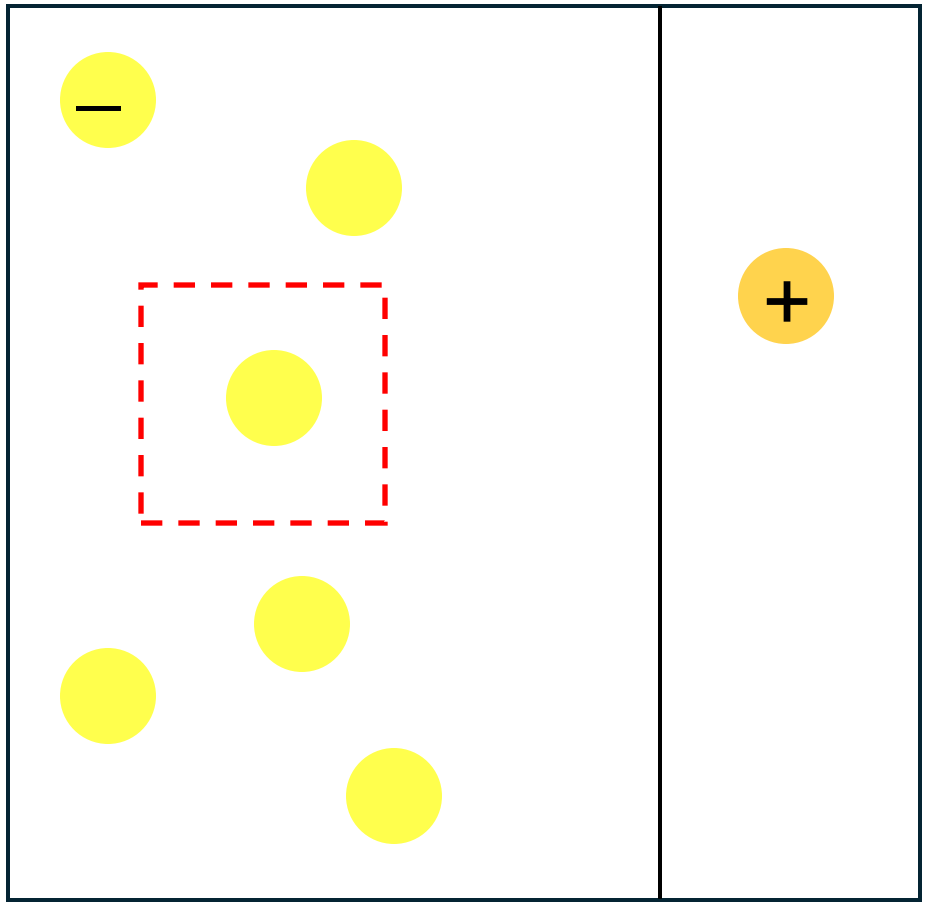
[positive majority case]



- Accuracy =

[5] Classification Evaluation (data imbalance neg>>pos)

[negative majority case]



- Accuracy =

[6] Classification Evaluation (data imbalance)

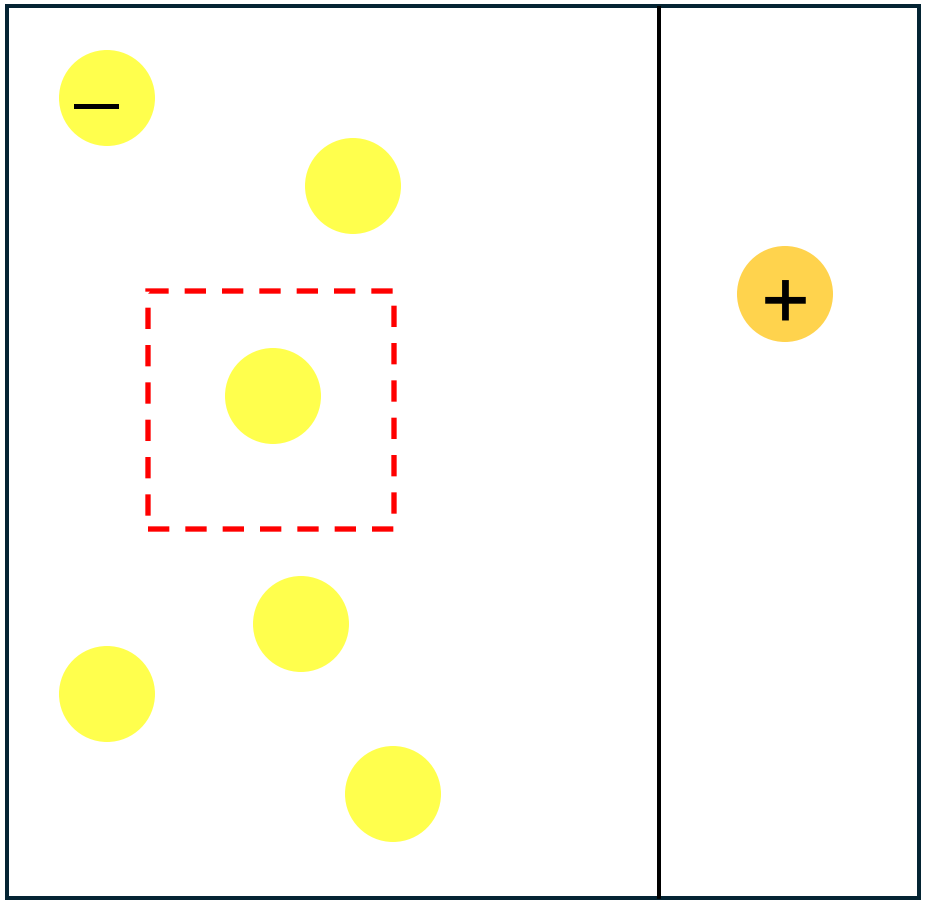
- minority is not classified correctly, but the overall accuracy can be high.

[7] Classification Evaluation (data imbalance)

The classifier cannot detect breast cancer
cannot detect diabetes
cannot detect a fraud card transaction from normal cases
is useless.

the accuracy for the biased classifier can be deceptively high.

[8] Classification Evaluation (data imbalance, right metric)



- the poor performance on minority is critical because general classification tasks focus on recognizing these minor classes.
- Don't miss positive cases!
- But be precise!

Q: which metric would be useful <u>to see if a classifier is truly good</u>?

[9] Classification Evaluation (Precision & Recall)

- Precision $= \dfrac{\#\ TP}{\#\ TP + FP \text{ (classified as positive)}}$

- Recall $= \dfrac{\#\ TP}{\#\ TP + FN \text{ (positive)}}$

❖ A good classification system should be high on both of precision and recall.

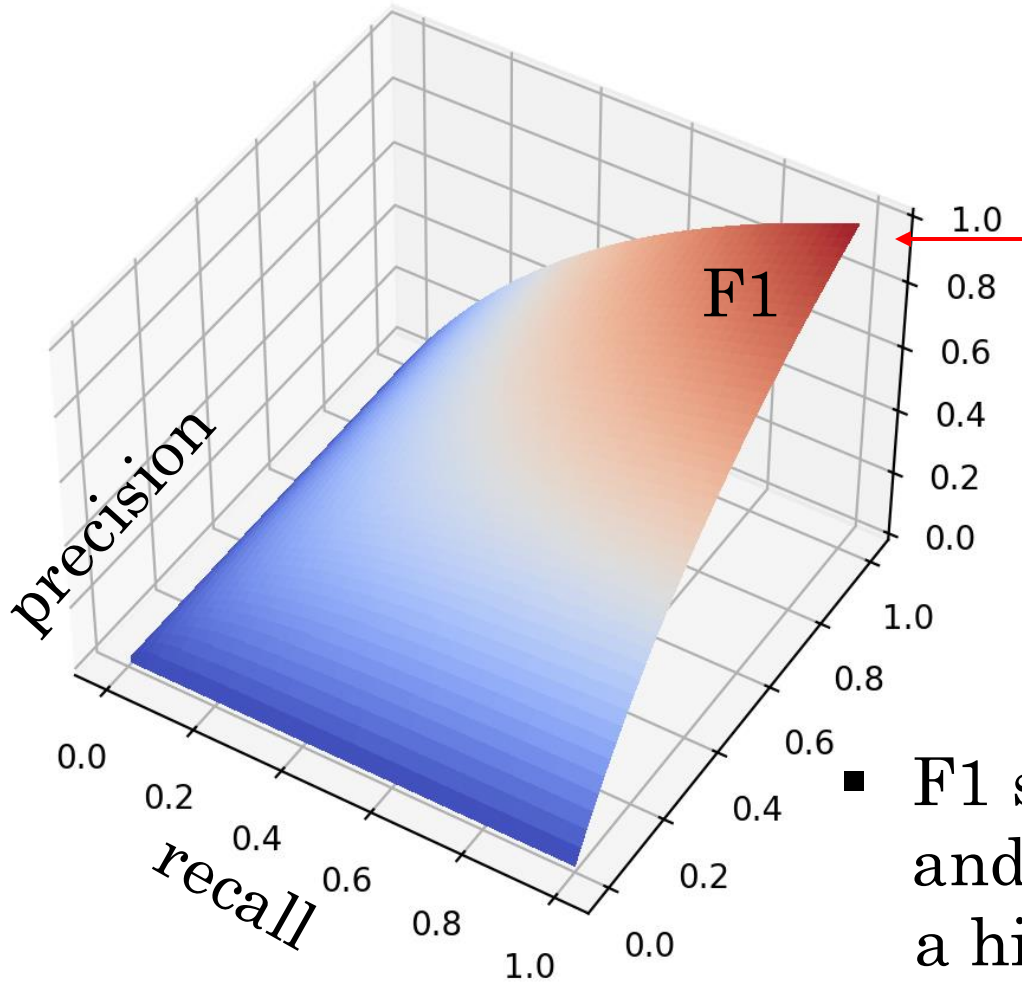
[10] Classification Evaluation (F1 score, harmonic mean)

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$= \frac{2}{1/\text{precision} + 1/\text{recall}}$$

Q: why not arithmetic mean?
½ (precision + recall) ?

[11] Classification Evaluation (F1 score, harmonic mean)



F1 is maximum when both recall and precision are one! if either of ones are zero then F1 is zero.

- F1 score is harmonic mean balancing precision and recall. it ensures that
a high precision / recall deceptively generates the high scores.

- The classifiers biased by data imbalance (Experiment Results)

** Logistic Regression Experiments Setup

- # data points (train / test): 10,000 (majority) vs. 100 (minority)
- data points ~ $N(\mu_k, \mathrm{I})$ for $k = 1$ (+) or 0 (−)
- gradient descents with $\eta = 0.1$ and 500 epochs
- scatter plots are based on test data

- Biased Classifiers by Data Imbalance (two classes are close)

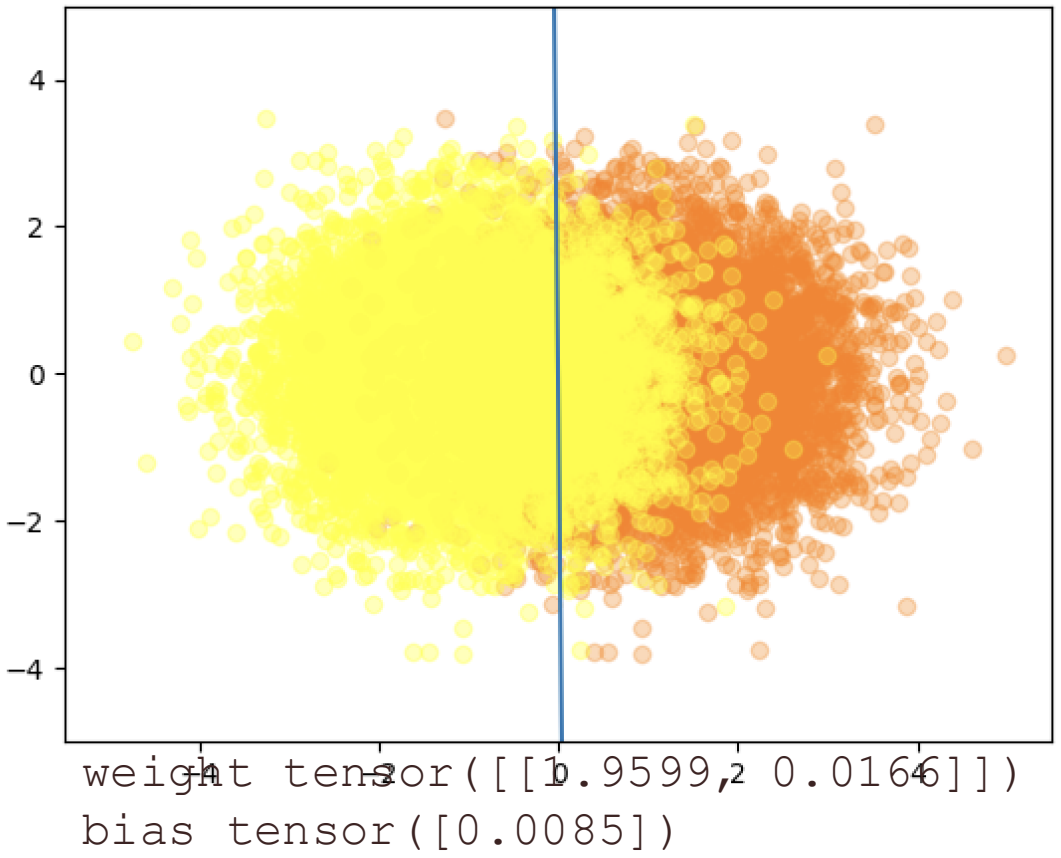
[1] Experiment Result (balanced + close classes)

- **accuracy: tensor(0.8425)**

- pos_precision: tensor([0.8378])
- pos_recall: tensor([0.8495])

- neg_precision: tensor([0.8474])
- neg_recall: tensor([0.8355])

[balanced & close]



- when the classes are balanced,
  the classifier separates the data points in the middle.

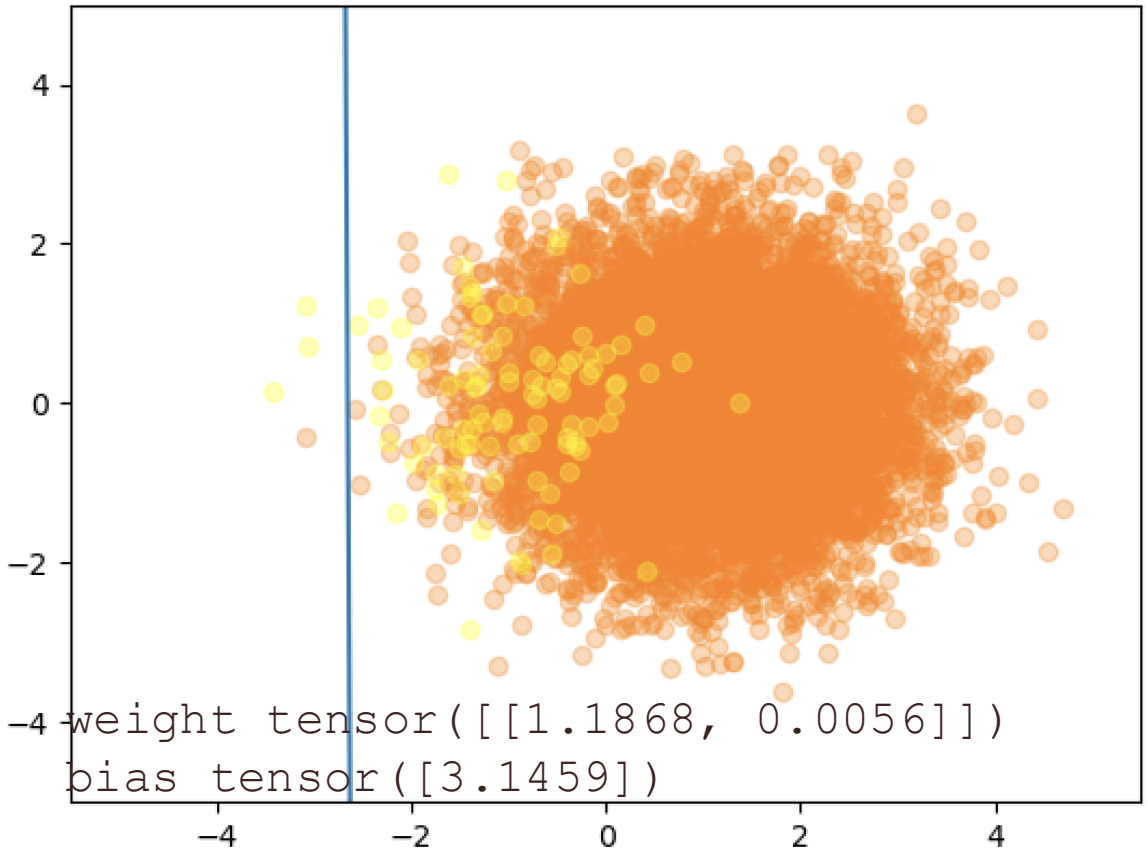
[2] Experiment Result (neg<<pos + close classes)

- **accuracy: tensor(0.9903)**

- pos tensor([9999.])
- pos_precision: tensor([0.9904])

- neg_precision: tensor([0.7500])
- neg_recall: tensor([0.0300])

[imbalanced (neg<<pos) & close]



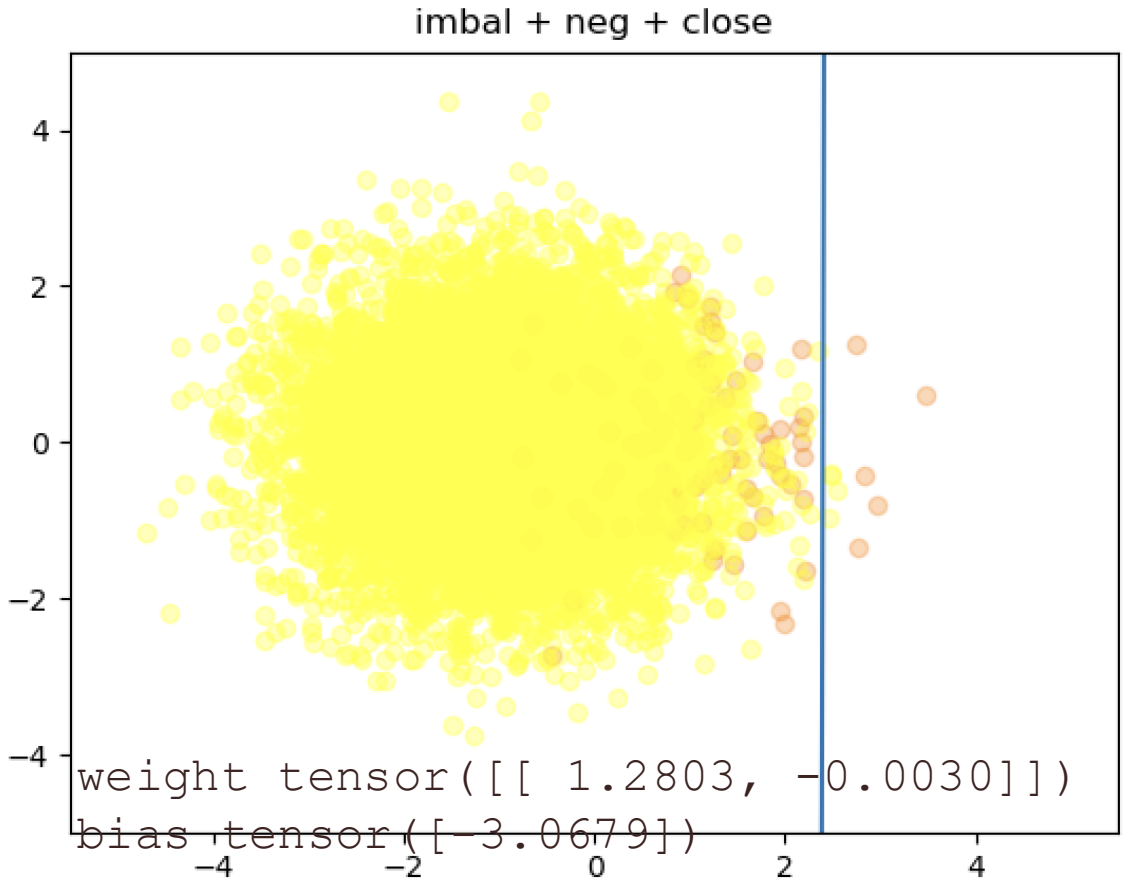
- when the classes are imbalanced (neg<<pos), the classifier is biased. accuracy is falsely high but the recall of minority are low.

[3] Experiment Result (neg>>pos + close classes)

- **accuracy: tensor(0.9902)**

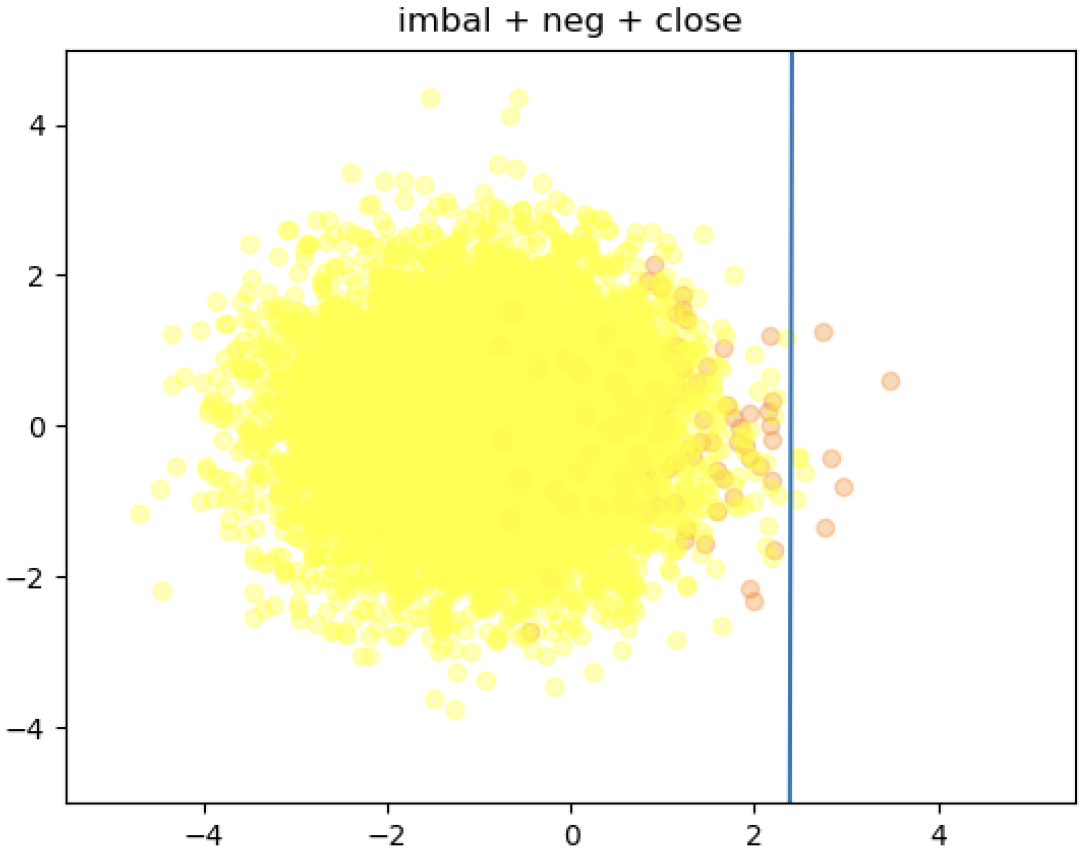
- pos_precision: tensor([0.5556])
- pos_recall: tensor([0.0500])

- **neg_precision: tensor([0.9906])**
- **neg_recall: tensor([0.9996])**



- when the classes are imbalanced (neg>>pos), the classifier is biased. accuracy is falsely high but the precision and recall of minority are low.

[4] Experiment Result (neg>>pos + close classes)




```
weight tensor([[ 1.2803, -0.0030]])
bias tensor([-3.0679])
```

- **accuracy: tensor(0.9902)**

- pos_precision: tensor([0.5556])
- pos_recall: tensor([0.0500])

- neg_precision: tensor([0.9906])
- neg_recall: tensor([0.9996])

- the poor performance on minority becomes critical
  because general classification tasks focus on recognizing these minor classes.

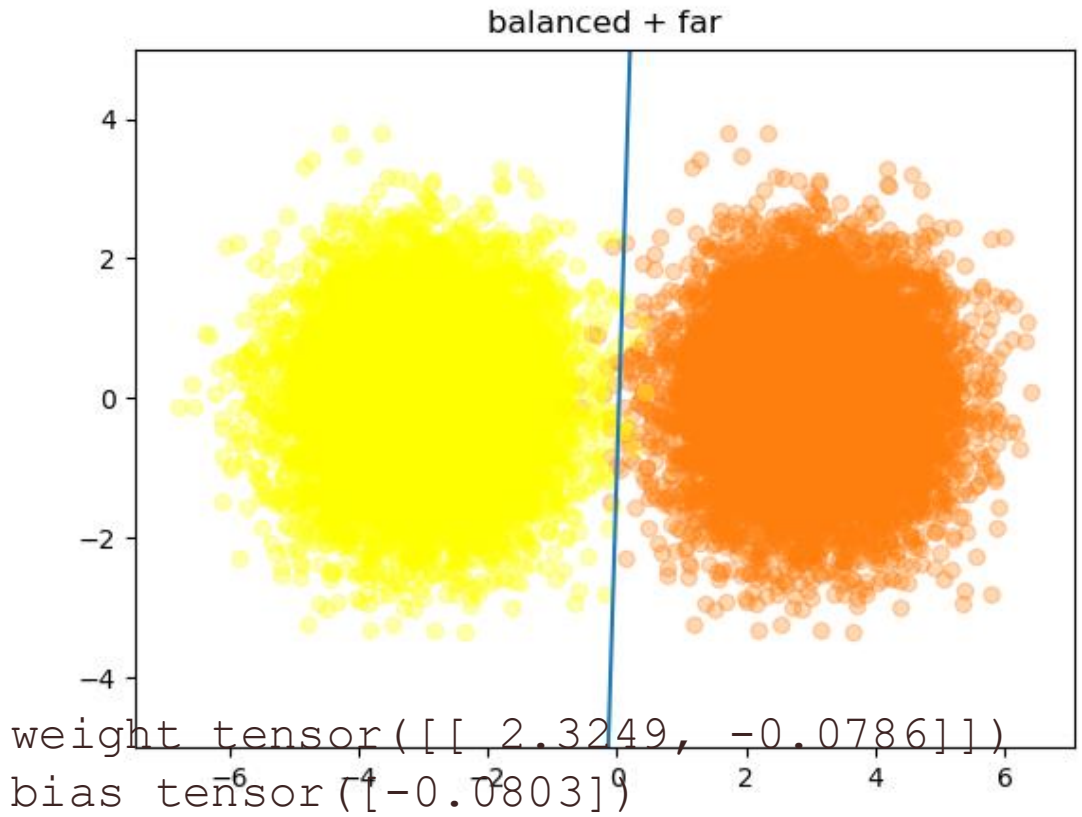
- Biased Classifiers by Data Imbalance (two classes are far)

data imbalance has less impact when the two classes are well separated

[1] Experiment Result (balanced + far classes)

- **accuracy: tensor(0.9987)**

- pos_precision: tensor([0.9991])
- pos_recall: tensor([0.9984])
- neg_precision: tensor([0.9984])
- neg_recall: tensor([0.9991])

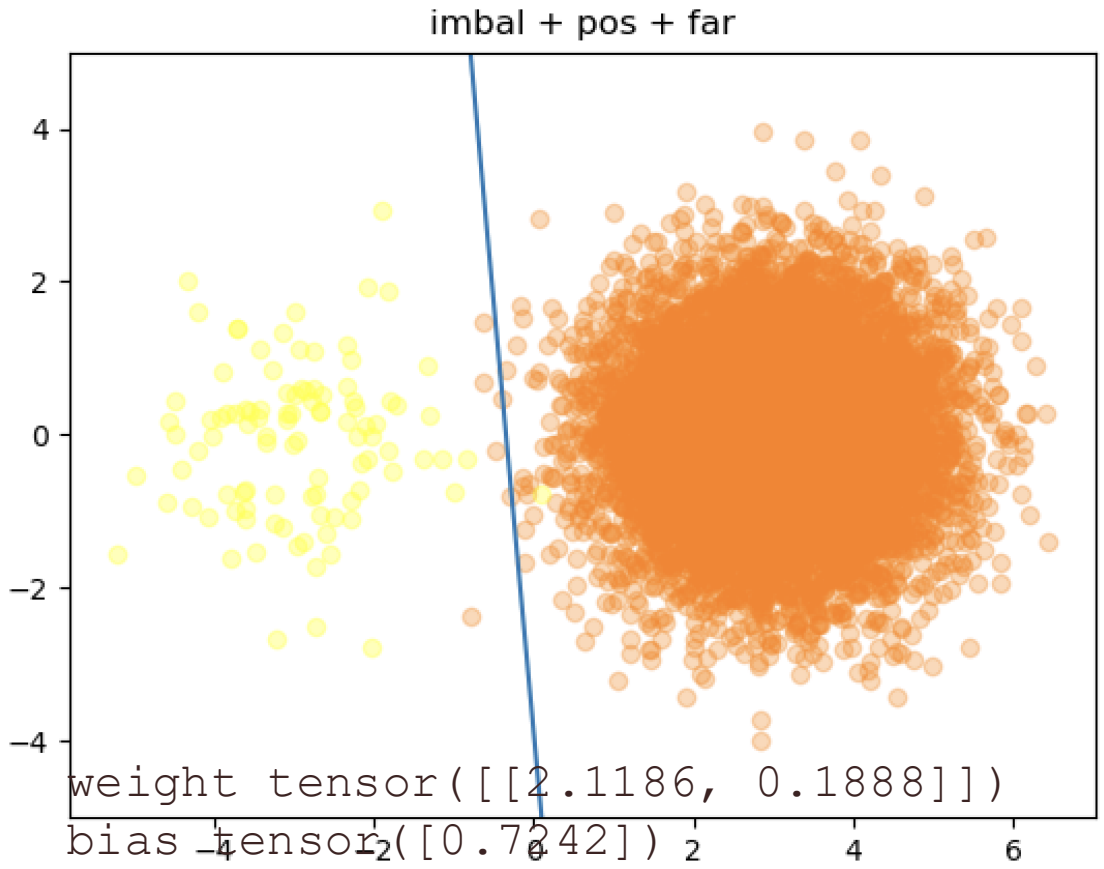


  - when the classes are balanced,
    the classifier separates the data points in the middle.

[2] Experiment Result (neg<<pos + far classes)

- **accuracy: tensor(0.9993)**

- pos_precision: tensor([0.9999])
- pos_recall: tensor([0.9994])
- neg_precision: tensor([0.9429])
- neg_recall: tensor([0.9900])

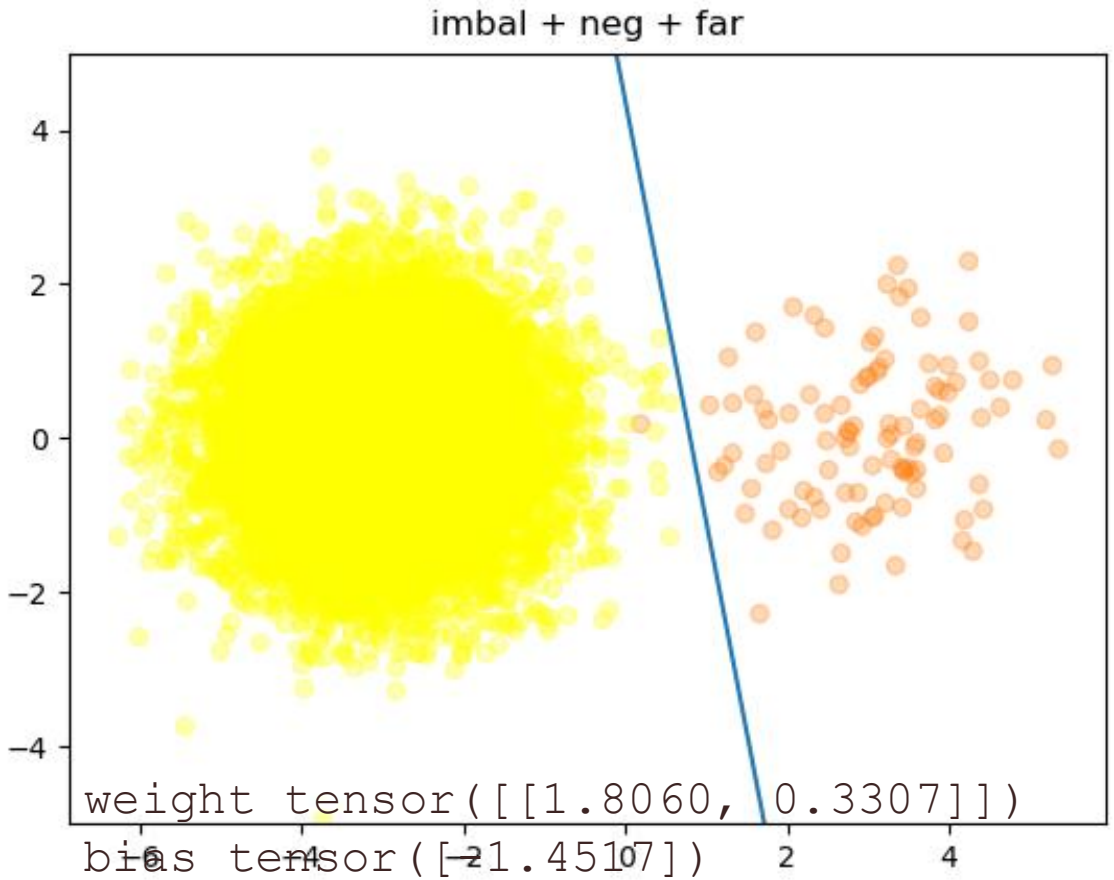


- even when the classes are imbalanced (neg<<pos), the classifier is not biased if the distance between the two classes is sufficiently large.

[3] Experiment Result (neg>>pos + far classes)

- **accuracy: tensor(0.9999)**

- pos_precision: tensor([1.])
- pos_recall: tensor([0.9900])
- neg_precision: tensor([0.9999])
- neg_recall: tensor([1.])



- even when the classes are imbalanced (neg>>pos), the classifier is not biased if the distance between the two classes is sufficiently large.

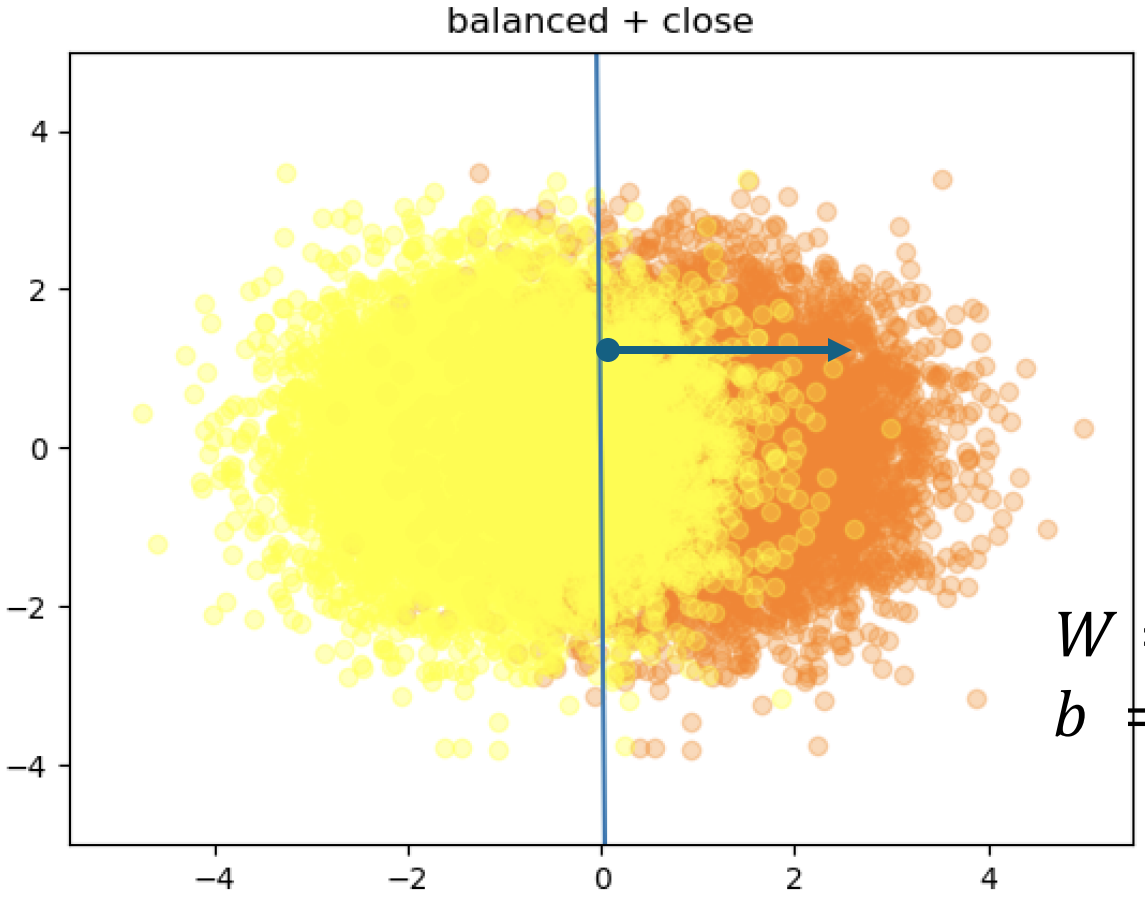
[4] Experiment Result (data imbalance, far vs. close classes)

- The data imbalance becomes a problem,
  when the data sample from the binary classes are close to each other or sharing overlapping geometrical regions.

- Classification Performance with Different Thresholds

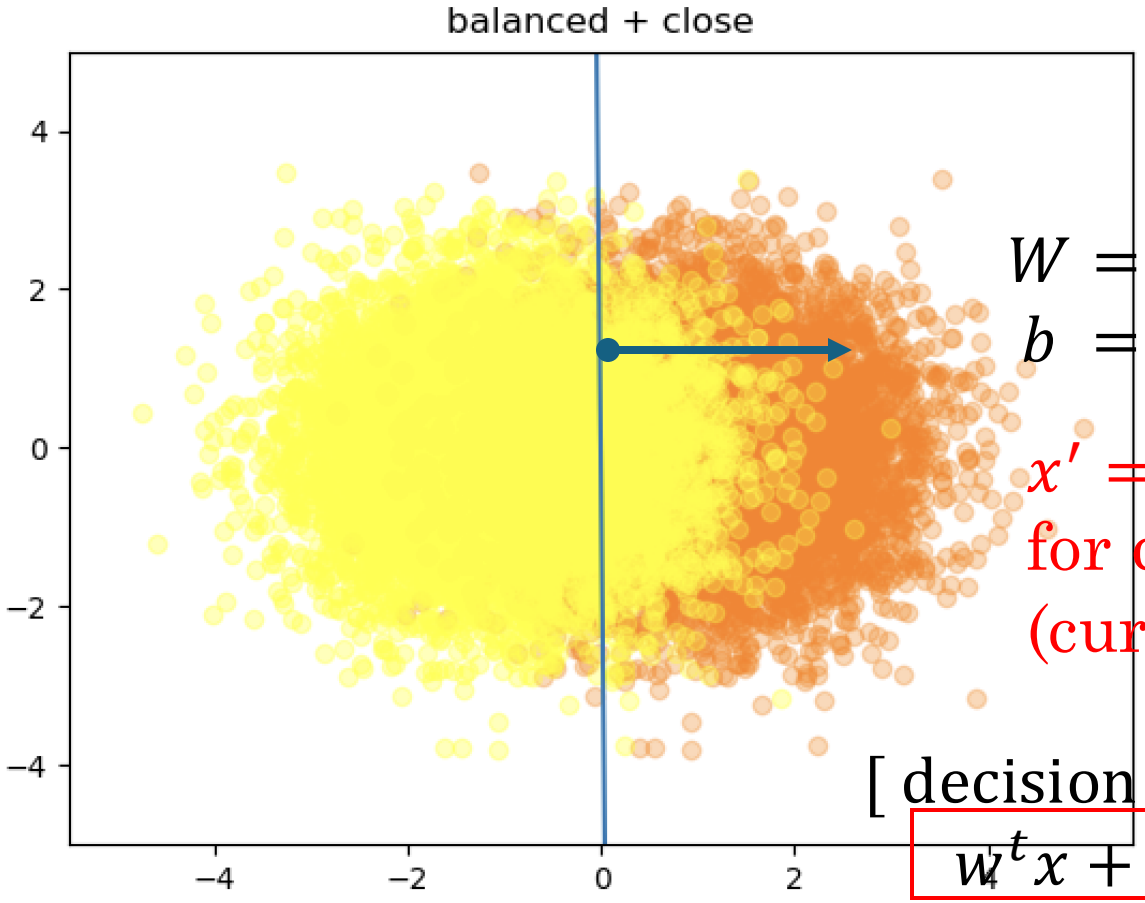
  -ROC curve / PR curve

[1] Classification Performance ($\eta = 0$)



$$W = [1.96, 0.0166]$$
$$b = 0.0085$$

[2] Classification Performance ($\eta = 0$)



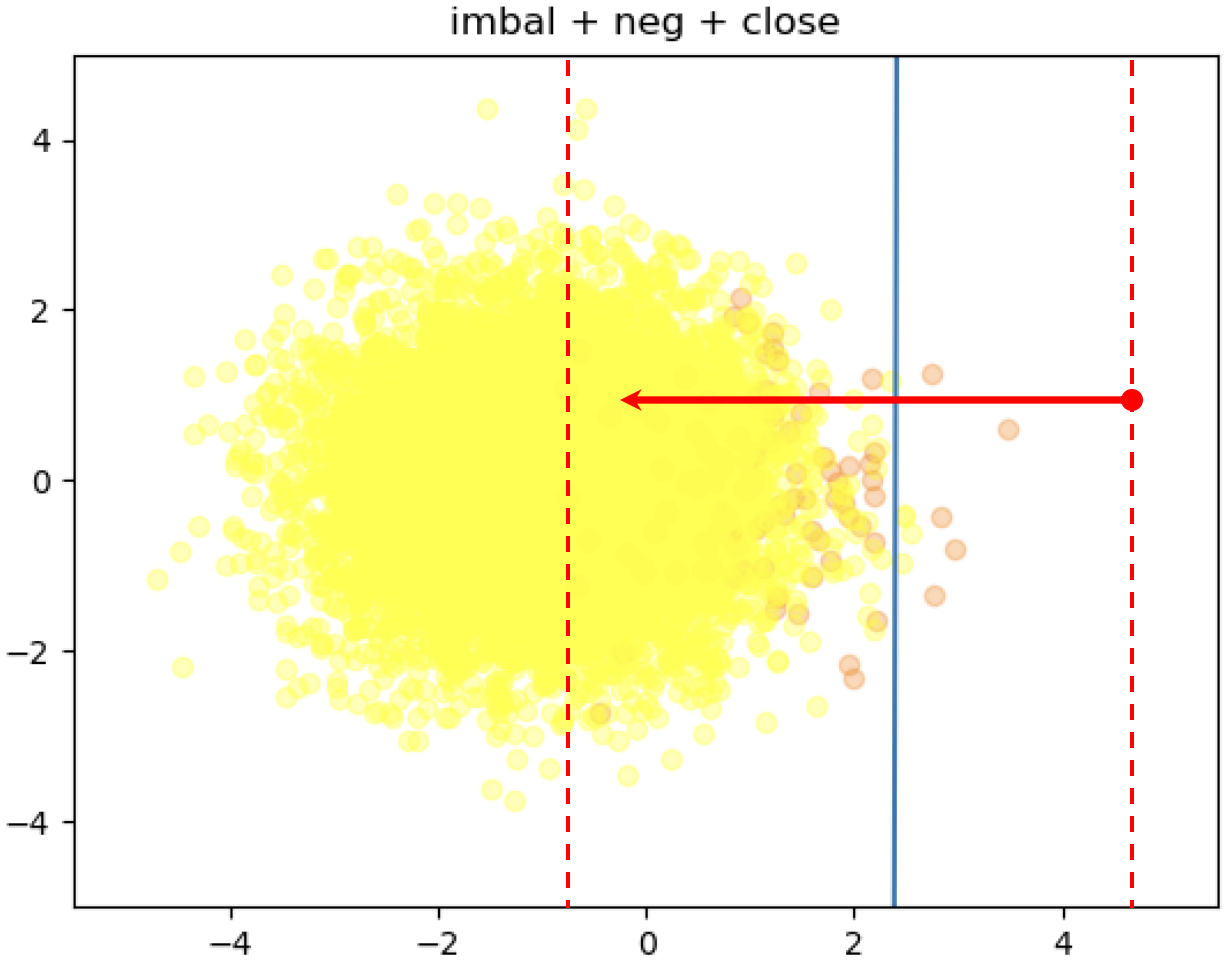
$$W = [1.96, 0.0166]$$
$$b = 0.0085$$

x' =the projection of data points x by $(\vec{w}x + b)$.
for different b, the threshold will be changed.
(currently zero)

[decision rule]

$$w^t x + b \geq 0 \ \ then \ \ P[C_+|x] \geq P[C_-|x] \rightarrow x \in C_+$$
$$w^t x + b < 0 \ \ then \ P[C_+|x] < P[C_-|x] \rightarrow x \in C_-$$

[3] Classification with different thresholds



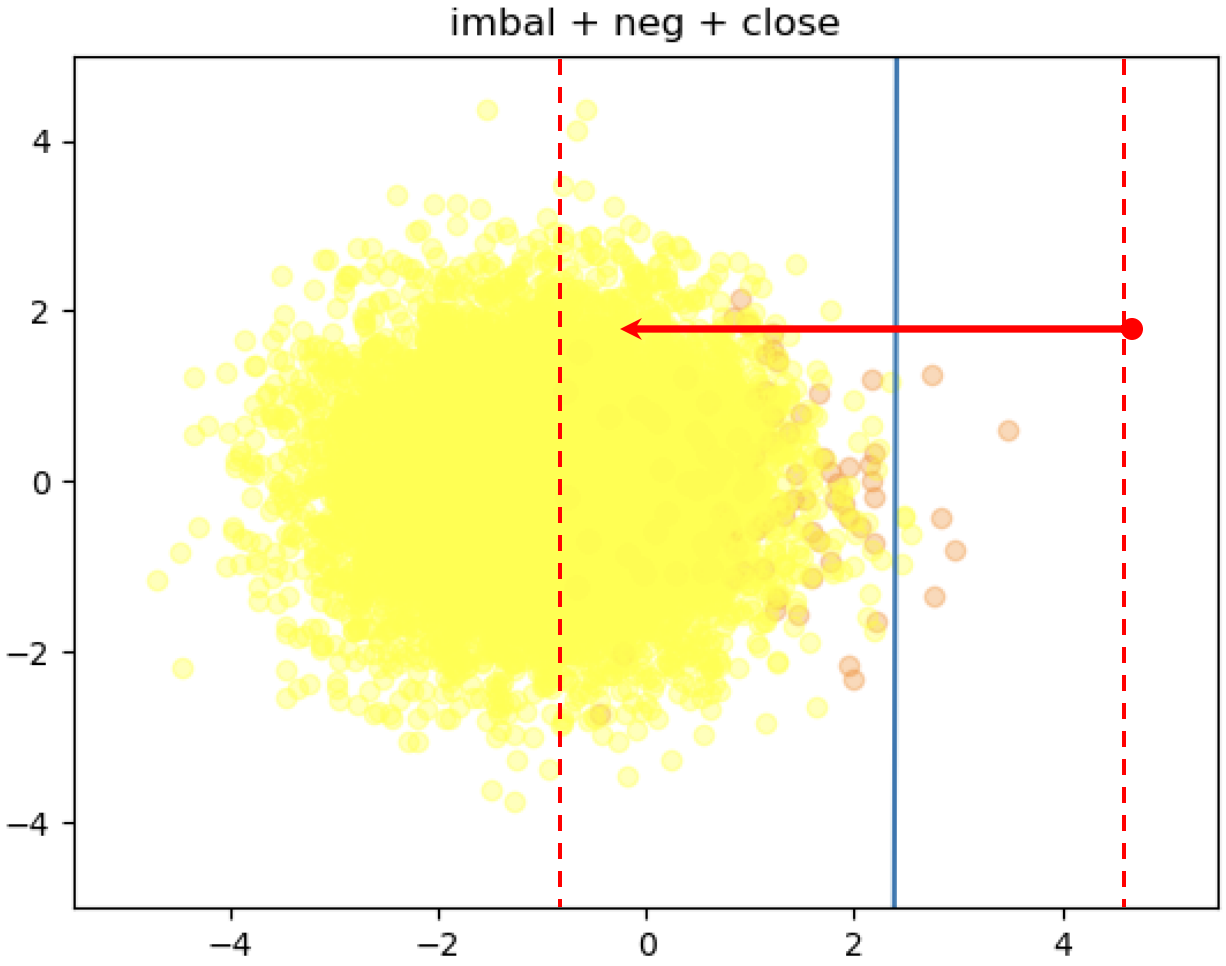
$W = [1.28, -0.003]$

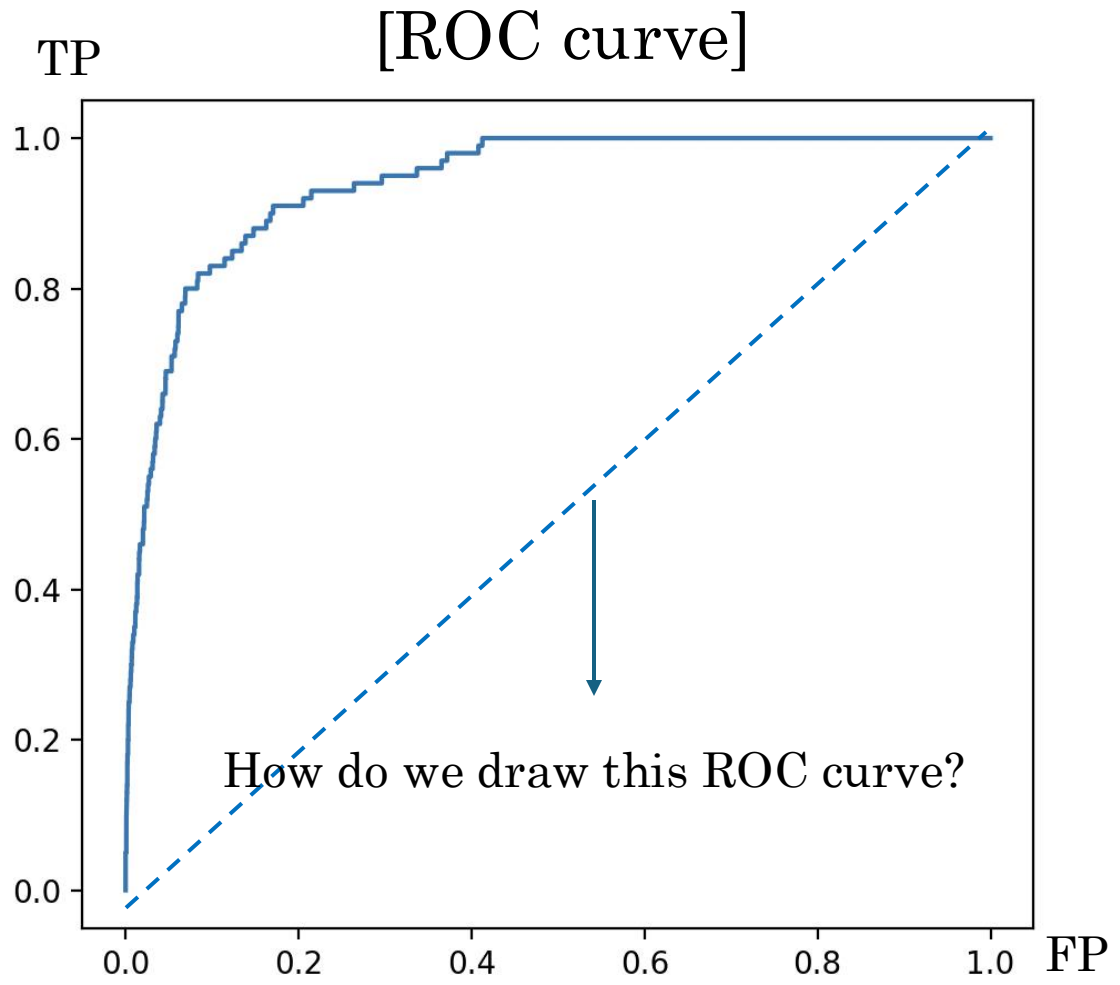
$b = -3.0679$

Q: how the TPR / FPR change as moving the threshold <u>from right to left</u>?

- TPR = $\dfrac{\#\ \text{TP}}{\#\ \text{TP} + \text{FN}}$
- FPR = $\dfrac{\#\ \text{FP}}{\#\ \text{TP} + \text{FN}}$

[4]ROC curve (Receiver Operator Characteristic)

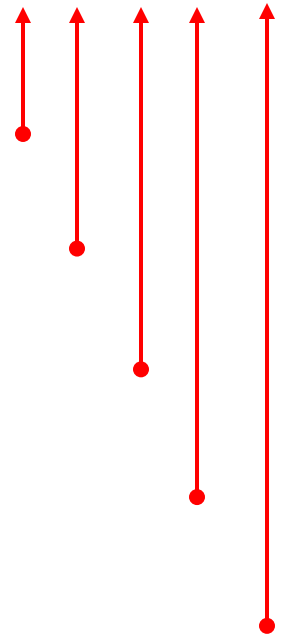




**drawing ROC curve (empirical computation)

- in the previous experiment, we had neg- : 10,000 pos + : 100 train / test samples
- we learned a hyperplane $y = w^t\overrightarrow{x} + b = [1.28, -0.003]^t\overrightarrow{x} - 3.06$
- based on the hyperplane, we can sort the data as descending order.
- for example,

$w^t\overrightarrow{x_{77}} + b$	$y_{77} = 1$
$w^t\overrightarrow{x_3} + b$	$y_3 = 0$
$w^t\overrightarrow{x_{16}} + b$	$y_{16} = 1$
$w^t\overrightarrow{x_1} + b$	$y_1 = 1$
…	…

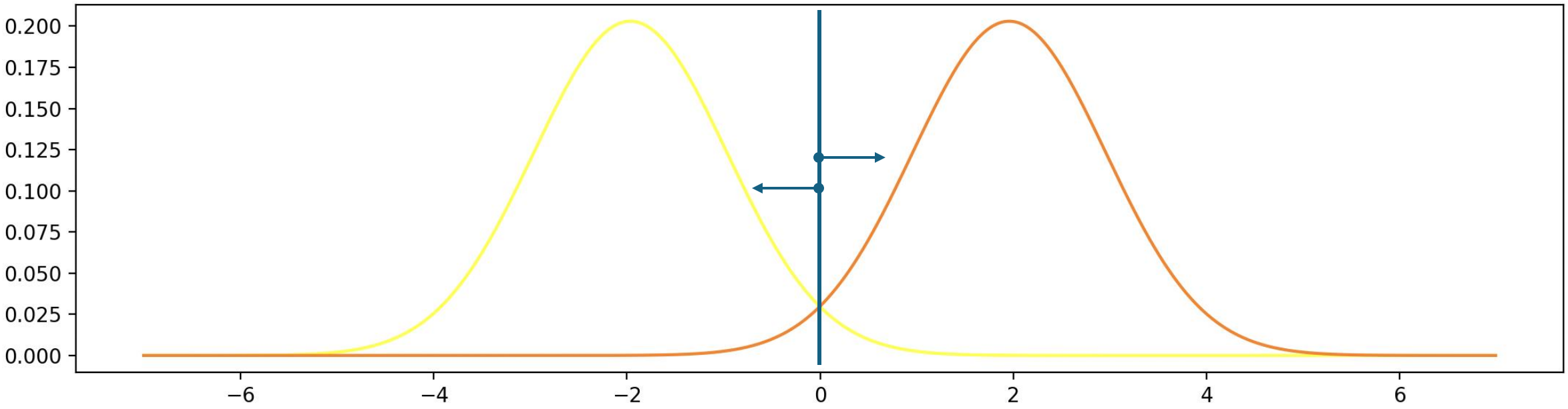


Classify as "1" if above η

- then compute TPR & FPR for the different η.

[5]ROC curve (theoretical computation)

Distribution of Projected Data to $\vec{w}$



depending on $\vec{w}$ (which hyperplane) and geometrical data configuration (far or close), the projected data distribution will be different.

[6] ROC curve

Q: why ROC curve?

1. one way to measure the quality of a classifier. (sensitivity over thresholds)

2. it integrates the performance over all possible thresholds,
   we have a unified /comprehensive view
   about models' performance.

3. especially, when data is not balanced,
   the performance sensitively varies with the choice of thresholds.
   ;not a good idea to take a threshold.

- Solutions for Data Imbalance

[1] Solutions (reweights, weighted cross entropy)

- PyTorch, "weighted cross entropy" function
  - Probabilities for each class; useful when labels beyond a single class per minibatch item are required, such as for blended labels, label smoothing, etc. The unreduced (i.e. with `reduction` set to `'none'`) loss for this case can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c}$$

where x is the input, y is the target, w is the weight, C is the number of classes, and N spans the minibatch dimension as well as $d_1, \dots, d_k$ for the *K*-dimensional case. If `reduction` is not `'none'` (default `'mean'`), then

$$\ell(x, y) = \begin{cases} \frac{\sum_{n=1}^{N} l_n}{N}, & \text{if reduction} = \text{'mean'}; \\ \sum_{n=1}^{N} l_n, & \text{if reduction} = \text{'sum'}. \end{cases}$$

[2] Solutions (resampling, SMOT)

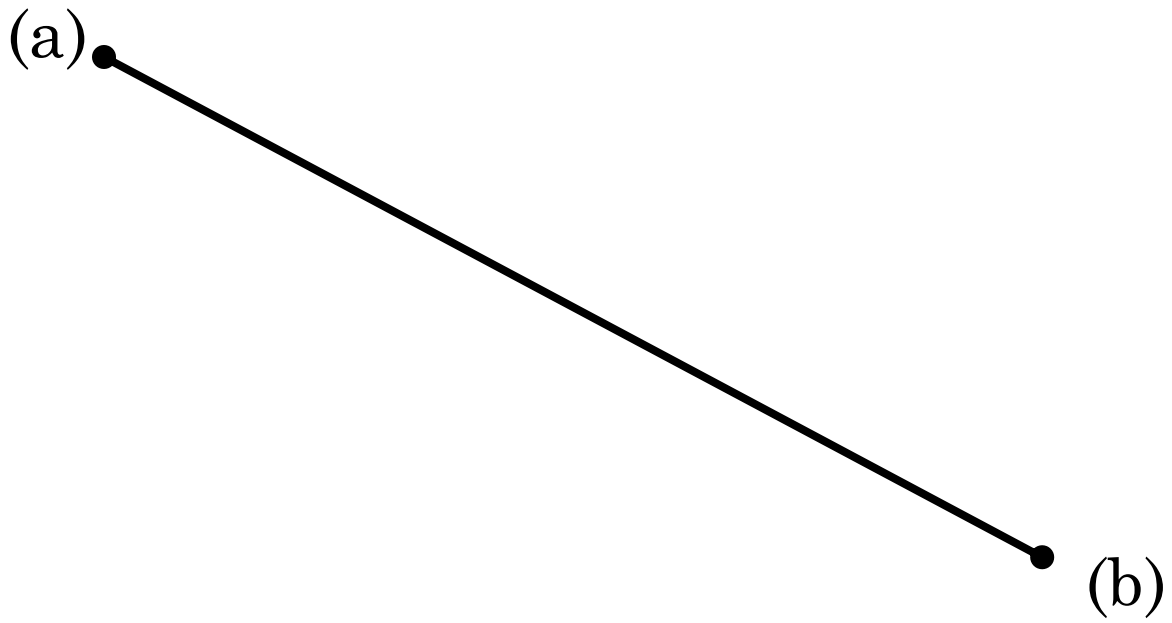
- **S**ynthetic **M**inority **O**versampling **T**echnique (SMOT)

  https://arxiv.org/abs/1106.1813/
  https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

  SMOT
  selects a minority class instance **a** at random and finds its *k* nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors **b** at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances **a** and **b**.

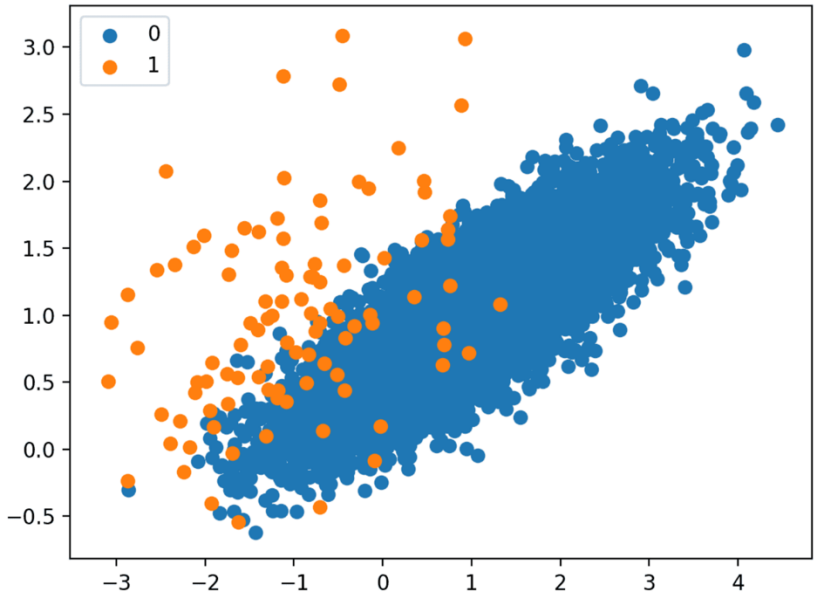
[3] Solutions (resampling, SMOT)

- **S**ynthetic **M**inority **O**versampling **T**echnique (SMOT)
  (perform on feature space)

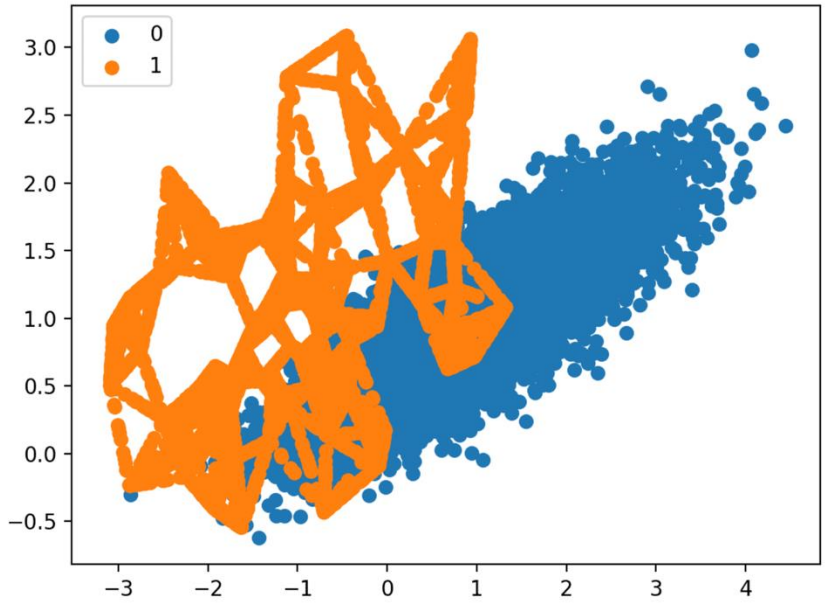


[4] Solutions (resampling, SMOT)

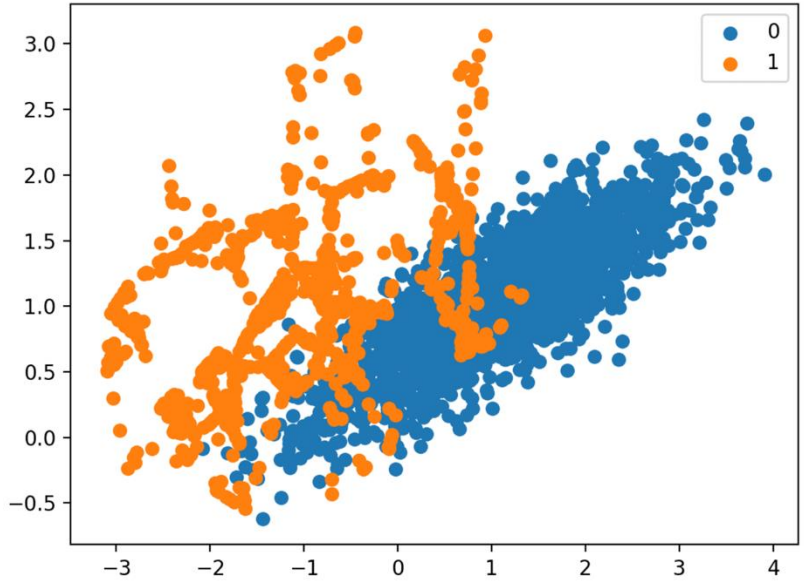
figure credit: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/



[data imbalance]



[sample generation for minority class]



[oversampling for minority undersampling for majority]