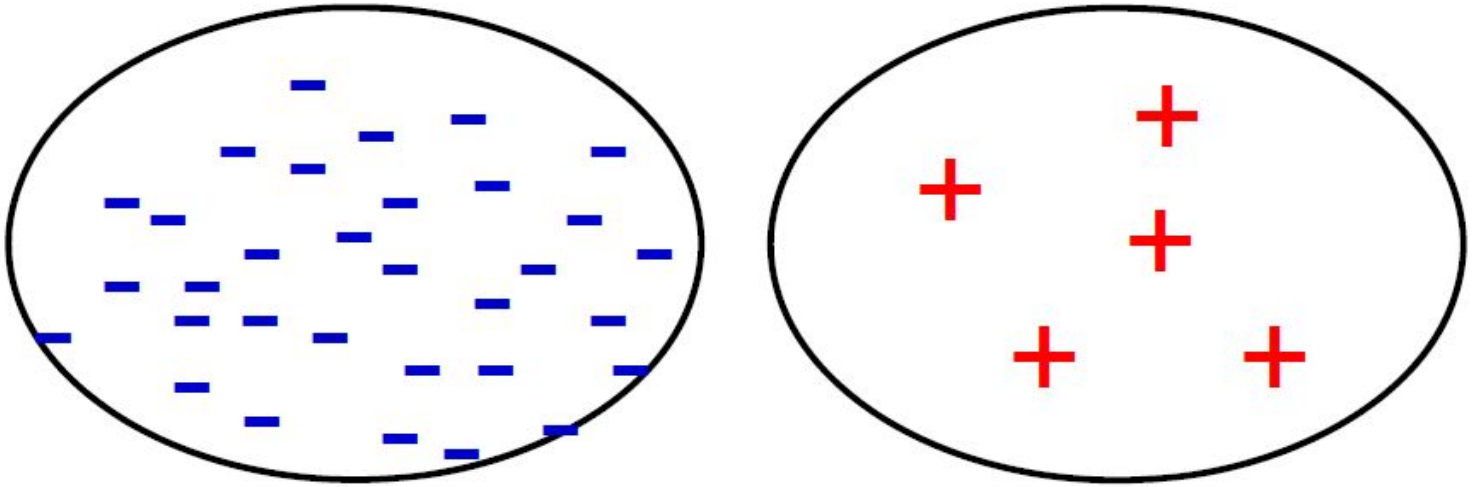
Классификция

Задача **классифика́ции** — задача, в которой имеется множество, разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Проблема несбалансированности

- Данные **несбалансированы** когда представители классов представлены не в приблизительном равном количестве (далее все рассматриваем



В чем проблема?

- Многие стандартные классификаторы пытаются увеличить точность и не изменить распределение обучающей выборки, поэтому они игнорируют маленькие классы. Если данные не сбалансированы, то предсказание большего класса для любого объекта приводит к точности порядка 90% (в зависимости от соотношения классов)

Цель классификации - детектирование

Стоимость ошибки неправильно классифицировать ненормальный образец данных как нормальный много выше чем наоборот.

Пример - поиск раковых клеток среди здоровых

Примеры несбалансированных данных:

- 1) из 100 000 тысяч подавших заявку, только 2% проходят в гарвард на стажировку
- 2) автоматизированная машина проверяющая на дефект произведенные на конвейере продукты намного чаще выбирает продукт без дефекта
- 3) тест на проверку заболевания раком получает в результатах много больше здоровых людей чем больных
- 4) в отслеживании воровства кредитных карт законных переводов много больше чем незаконных
- 5)мошеннические телефонные звонки
- 6)обнаружение нефтяных пятен по изображениям со спутника
- 7)оценка рисков

Техники работы с несбалансированными данными

I. Работа с данными :

1) SMOTE
2) Random Undersampling
3) Random Oversampling

II. Чувствительность к стоимости ошибки

III. Выбор характеристик

Метрики качества

Пусть есть два класса — отрицательный и положительный (меньший)

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Confusion Matrix

1) Accuracy – для сбалансированных данных

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Процент правильно классифицированных образцов от всего числа образцов

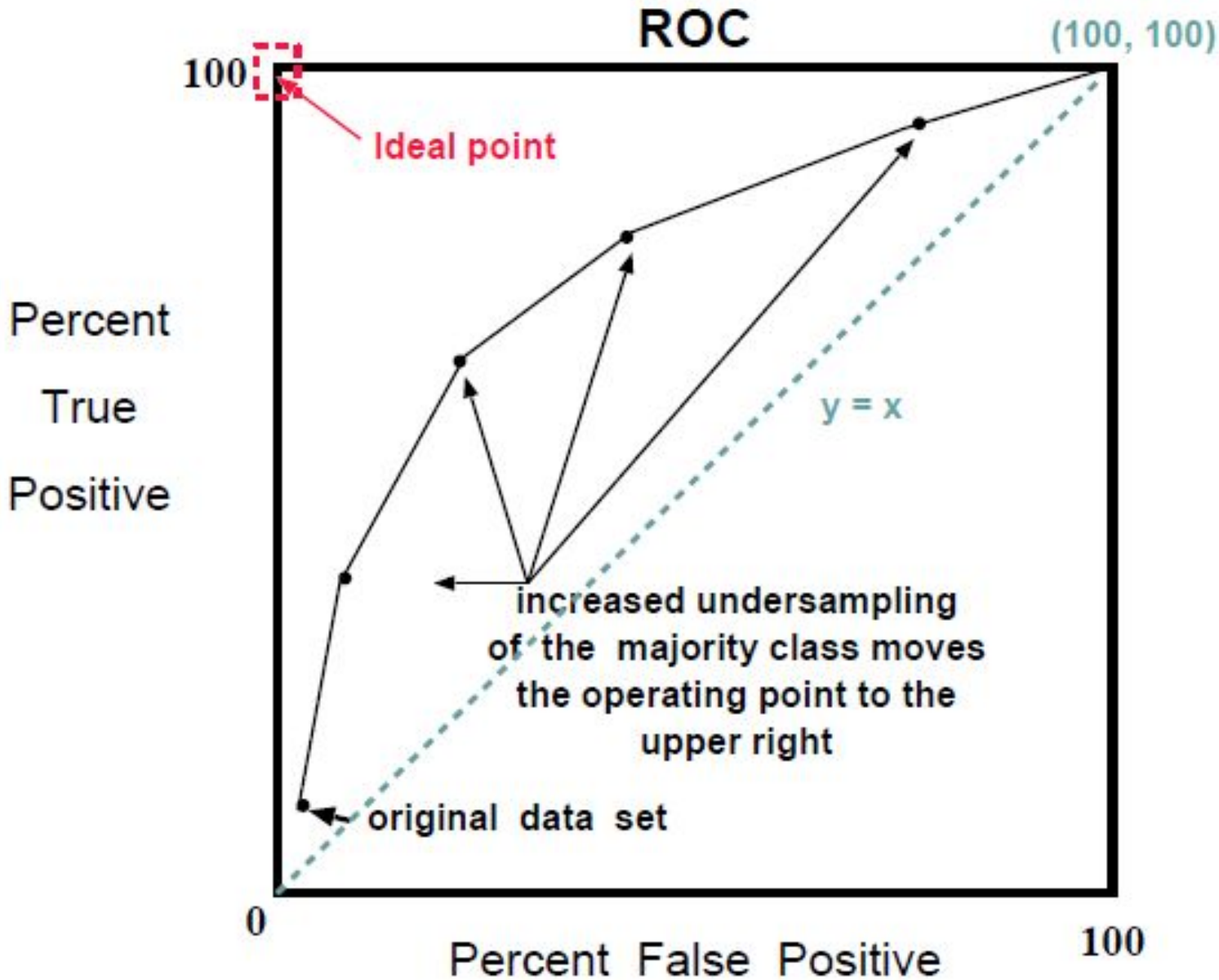
2) ROC кривая – для несбалансированных

- представляет границы лучших решений для относительных TP (по оси У) & FP(по оси X)

$$FP = \frac{FP}{FP + TN}$$

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

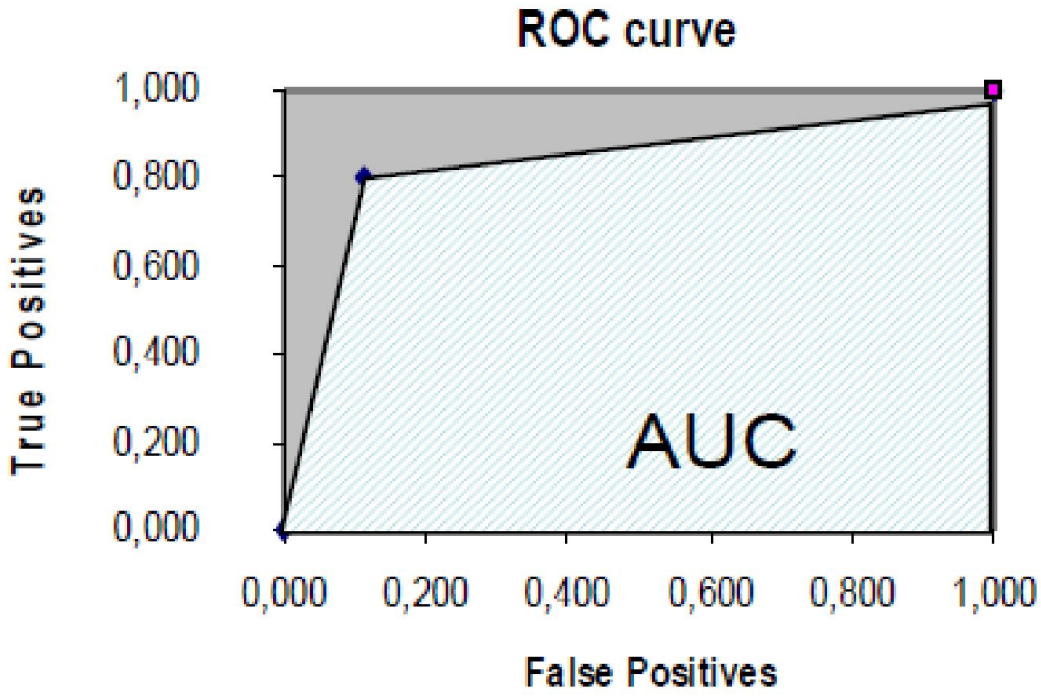
- каждая точ акими-то параметрами
- линия x=y — при произвольном выборе метки класса


ROC
(100, 100)
100
Ideal point
Percent
True
Positive
y = x
increased undersampling
of the majority class moves
the operating point to the
upper right
original data set
0
Percent False Positive
100

AUC - площадь под ROC кривой .

- Она эквивалентна вероятности того что классификатор ценит произвольно выбранный образец меньшего класса выше чем произвольно выбранный образец из большего класса. (она больше 0,5)
- Т.е. это численная характеристика для сравнения классификаторов

Для одной точки



$$\mathrm{AUC} = \frac{1 + \mathrm{TP}_{\mathrm{rate}} - \mathrm{FP}_{\mathrm{rate}}}{2}$$

Преимущества ROC

- Когда алгоритм изучает больше образцов одного (-) класса он будет ошибочно классифицировать больше образцов другого класса (+). т.о. ROC изображает согласование между долей правильных и долей ложных предсказаний классификатора.
- ROC показывает в каком диапазоне (в нашем случае соотношений обьемов классов) классификатор лучше других
- ROC кривые нечувствительны к распределению по классам т. е. если соотношение между образцами из меньшего и большего класса изменится ROC кривая не изменится

Алгоритм SMOTE

1) Считываем число образцов меньшего класса T
2) Процент генерируемых образцов N
3) Число ближайших соседей k
4) Для каждого образца (i) (вектора из атрибутов) из T(меньшего класса) находим k ближайших соседей и генерируем [N/100] исскуственных образцов, повторяя на каждом шаге:
5) Из найденных соседей произвольно выбираем одного (nn), прибавляем к каждому из атрибутов i разницу между соответсвующими атрибутами i и nn, умноженную на произвольное число из отрезка [0,1] – получили новый вектор атрибутов – это новый искуственный образец меньшего класса

(атрибуты здесь – непрерывные величины,
т.е. числа)

SMOTE

Algorithm *SMOTE*(T, N, k)

Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k

Output: $(N/100) * T$ synthetic minority class samples

1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. k = Number of nearest neighbors
9. $numattrs$ = Number of attributes
10. $Sample[\,][\,]$: array for original minority class samples
11. $newindex$: keeps a count of number of synthetic samples generated, initialized to 0
12. $Synthetic[\,][\,]$: array for synthetic samples

```
Synthetic[ ][ ]: array for synthetic samples
(* Compute k nearest neighbors for each minority class sample only. *)
for i ← 1 to T
        Compute k nearest neighbors for i, and save the indices in the nnarray
        Populate(N, i, nnarray)
endfor

Populate(N, i, nnarray) (* Function to generate the synthetic samples. *)
while N ≠ 0
        Choose a random number between 1 and k, call it nn. This step chooses one of
        the k nearest neighbors of i.
        for attr ← 1 to numattrs
                Compute: dif = Sample[nnarray[nn]][attr] − Sample[i][attr]
                Compute: gap = random number between 0 and 1
                Synthetic[newindex][attr] = Sample[i][attr] + gap * dif
        endfor
        newindex++
         N = N − 1
endwhile
return (* End of Populate. *)
End of Pseudo-Code.
```

Преимущества SMOTE

- Этот способ увеличения меньшего класса не приводит к переобучению (в отличие от random oversampling), т. е. алгоритм одинаково хорошо работает и на новых данных.
- Множественные примеры с различным распределением данных и соотношением представителей классов показывают, что SMOTE работает лучше
- Не требует инициализации каких-либо величин, что сильно влияло бы на результат классификации

Недостатки SMOTE

Данный алгоритм не выходит за рамки существующих образцов меньшего класса, т.е. не будут созданы образцы с существенно отличными атрибутами, что вполне возможно в настоящих данных

Модификации SMOTE для дискретных атрибутов образцов

SMOTE-NC

F1 = 1 2 3 A B C [Let this be the sample for which we are computing nearest neighbors]

F2 = 4 6 5 A D E

F3 = 3 5 6 A B K

So, Euclidean Distance between F2 and F1 would be:

Eucl = sqrt[$(4\text{-}1)^2 + (6\text{-}2)^2 + (5\text{-}3)^2 + \text{Med}^2 + \text{Med}^2$]

Med is the median of the standard deviations of continuous features of the minority class.

The median term is included twice for feature numbers 5: B→D and 6: C→E, which differ for the two feature vectors: F1 and F2.

При вычислении атрибутов генерируемого образца для номинальных атрибутов значением будут самые частые соответсвующие номинальные атрибуты среди k ближайших соседей и рассматриваемого образца

SMOTE-N

Let F1 = A B C D E be the feature vector under consideration and let its 2 nearest neighbors be

F2 = A F C G N

F3 = H B C D N

The application of SMOTE-N would create the following feature vector:

FS = A B C D N

Расстояние между двумя значениями атрибутов V1 и V2(где C1 – число всех значений V1 по всем образцам, а C1i - то же число, но в классе i, обычно k=1

$$\delta(V_1, V_2) = \sum_{i=1}^{n} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k$$

Расстояние между двумя образцами X и У (где N – число атрибутов, r = 2 для Евклидовой нормы)

$$\Delta(X, Y) = \sum_{i=1}^{N} \delta(x_i, y_i)^r$$