«Машинное обучение»

Специальные задачи машинного обучения

Александр Дьяконов

План

Многоклассовые задачи

Несбалансированные данные – Imbalanced Data

Активное обучение – Active Learning

Ковариальный сдвиг – Covariate Shift

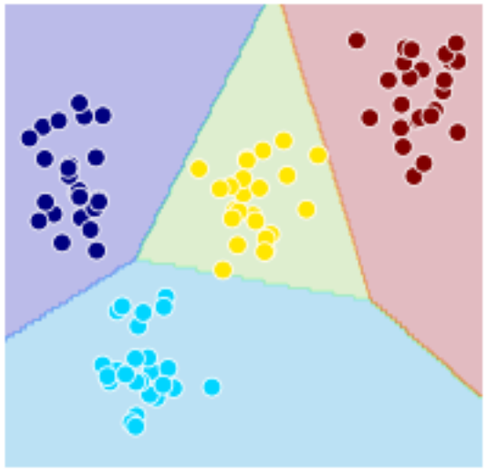
Если алгоритм показывает плохое качество

Многоклассовые задачи: если умеем решать задачу бинарной классификации

1) One versus All (the rest) – один против всех – отделяем класс от остальных l раз

дисбаланс классов, линейная сложность от l, усложение задачи

смотрим на вероятности принадлежности к классам



	target	y1	y2	y3
0	1	1	0	0
1	1	1	0	0
2	2	0	1	0
3	2	0	1	0
4	3	0	0	1
5	3	0	0	1

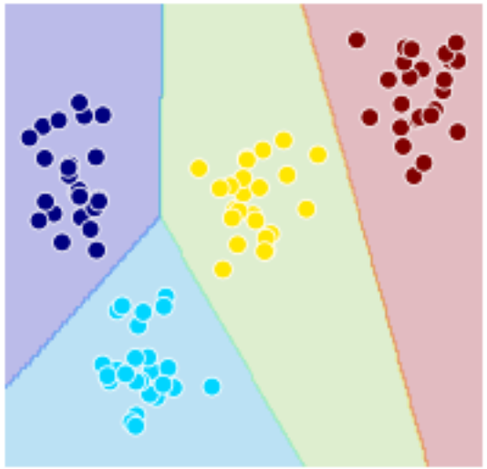
```
from sklearn.multiclass import OneVsRestClassifier
model = OneVsRestClassifier(LogisticRegression())
model.fit(X, y)
```

$$\{a_1,\ldots,a_l\}$$

$$y = \arg\max_t (a_t)$$

Многоклассовые задачи: если умеем решать задачу бинарной классификации

2) One versus One – один против одного – отделяем классы попарно, относим в тот, в который чаще выбирается (квадратичная сложность, разнообразие в ответах)



	target	y12	y23	y13
0	1	1	-	1
1	1	1	-	1
2	2	0	1	-
3	2	0	1	-
4	3	-	0	0
5	3	-	0	0

```
from sklearn.multiclass import OneVsOneClassifier
model = OneVsOneClassifier(LogisticRegression())
model.fit(X, y)
```

$$\{a_{12}(a_{21}=1-a_{12}), a_{23}(a_{32}=1-a_{23}), a_{13}(a_{31}=1-a_{13})\}$$

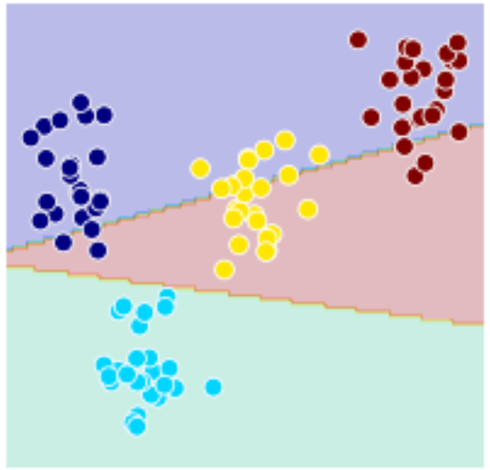
$$y = \arg\max\left(\sum_{\bullet\in\{1,2,\ldots,l\}\setminus\{t\}} a_{\bullet t}\right)$$

Многоклассовые задачи: если умеем решать задачу бинарной классификации

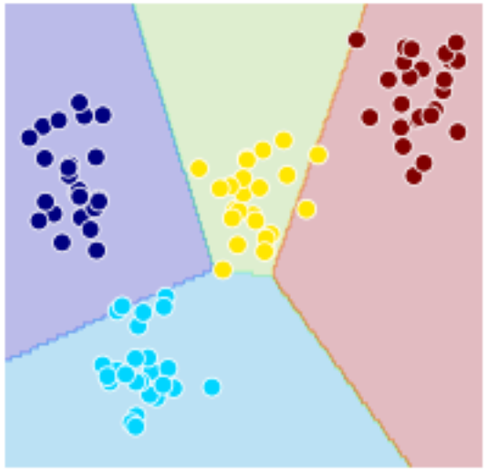
3) ECOC – использование бинарных кодовых векторов для классов

уже проходили в ансамблировании

`code_size=0.5`



`code_size=2.0`



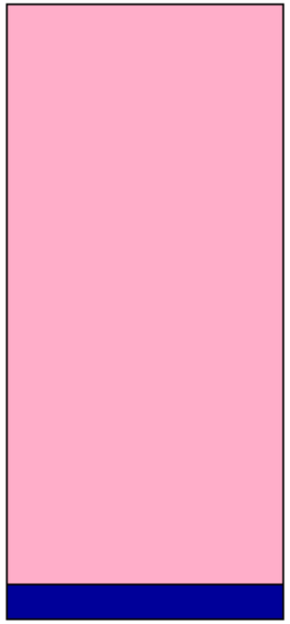
	tagret	y1	y2
0	1	1	1
1	2	1	0
2	3	0	1
3	4	0	0

`code_size` – **регулирует избыточность кода (длина =** `code_size` $\times$ **число классов)**

```
from sklearn.multiclass import OutputCodeClassifier
model = OutputCodeClassifier(LogisticRegression(), code_size=1.5)
model.fit(X, y)
```

Несбалансированные данные (Imbalanced Data)

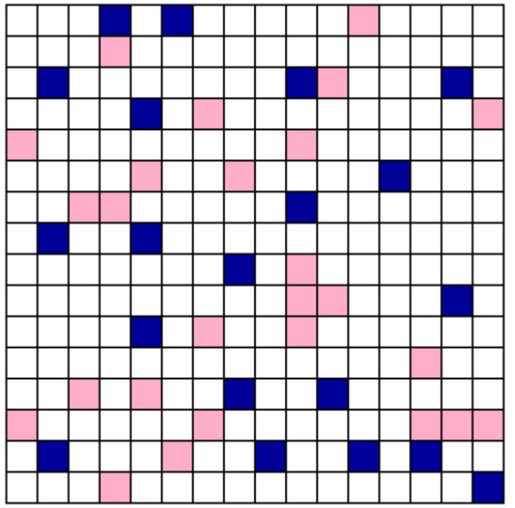
Несбалансированные



Пример:

- дефолт
- поломки
- мошенничество
- редкие заболевания

Разреженные



Что важно:

- природа задачи
- функция ошибки
- понимание отсутствия проблемы

Несбалансированные данные (Imbalanced Data)

Почему важна природа задачи?

Причины дисбаланса – может дубликаты и достаточно их устранить

Вид дисбаланса – сколько классов, какое распределение по ним, может «маленькие» классы не важны

Если выбросов <<1% и это поломки – возможно, лучше детектировать аномалии

Если дисбаланс меняется – логично угадывать пропорции классов

Если дисбаланс из-за недостатка разметки – есть варианты синтетических данных

Если 2-10% и он стабилен – **сейчас рассматриваем + 2 класса**

Почему важна функция ошибки / функционал качества?

LogLoss – дисбаланс даже благо, если пропорция классов не меняется

ROC AUC – дисбаланс не влияет на значение

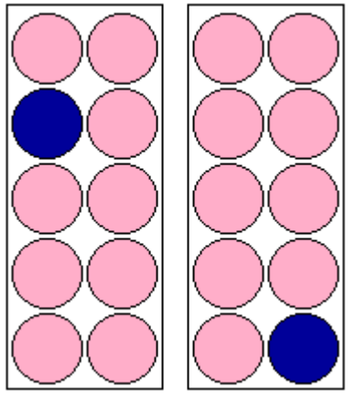
F1 – а вот тут интересно….

Почему нет проблемы дисбаланса?

Есть проблема выбора функционала качества, стоимость ошибок 1/2 рода, необходимость стратифицированного контроля

Несбалансированные данные (Imbalanced Data)

Небольшое пояснение, стоит ли гоняться за повышением, например F1-меры



Пусть есть 2 корзины, выбираем одну,
затем делаем прогноз, какой шар из неё извлечём

«шар белый»

«шар из первой корзины белый,
а из второй – чёрный»

«шар из первой корзины чёрный,
а из второй – белый»

$$Acc = 0.5 \times 0.9 + 0.5 \times 0.9 = 0.9,$$
$$F1 = 0$$

$$Acc = 0.5 \times 0.9 + 0.5 \times 0.1 = 0.5,$$
$$F1 = 2/(1/0.1 + 1/0.5)) = 0.16(6)$$

=*=

F1-мера растёт не из-за того, что прогноз становится адекватнее

Несбалансированные данные (Imbalanced Data)

- **Изменение выборки**
  - Недосэмплирование (Undersampling the majority class)
    - умные методы: NearMiss-1/2, Tomek links
  - Пересэмплирвоание (Oversampling the minority class)
    - умные методы: SMOTE, ADASYN
  - Перевзвешивание объектов (Reweighting the examples) (*)

    в DL есть аугментации со схожими идеями, например MixUp

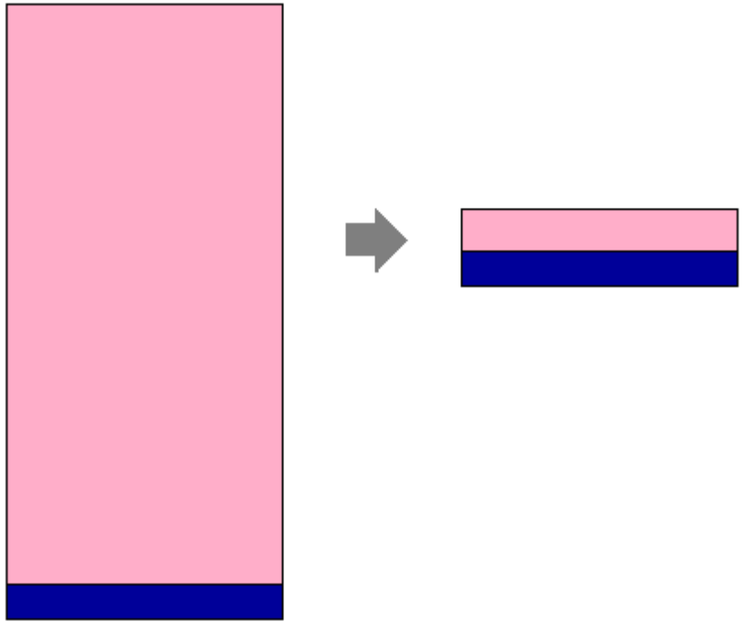
- **Изменение модели**
  - функция ошибки: учитывает дисбаланс (**)

- **Решающее правило**
  - подбор порога бинаризации

Часто (*) = (**)

Несбалансированные данные (Imbalanced Data)

Недосэмплирование
(Undersampling the majority class)



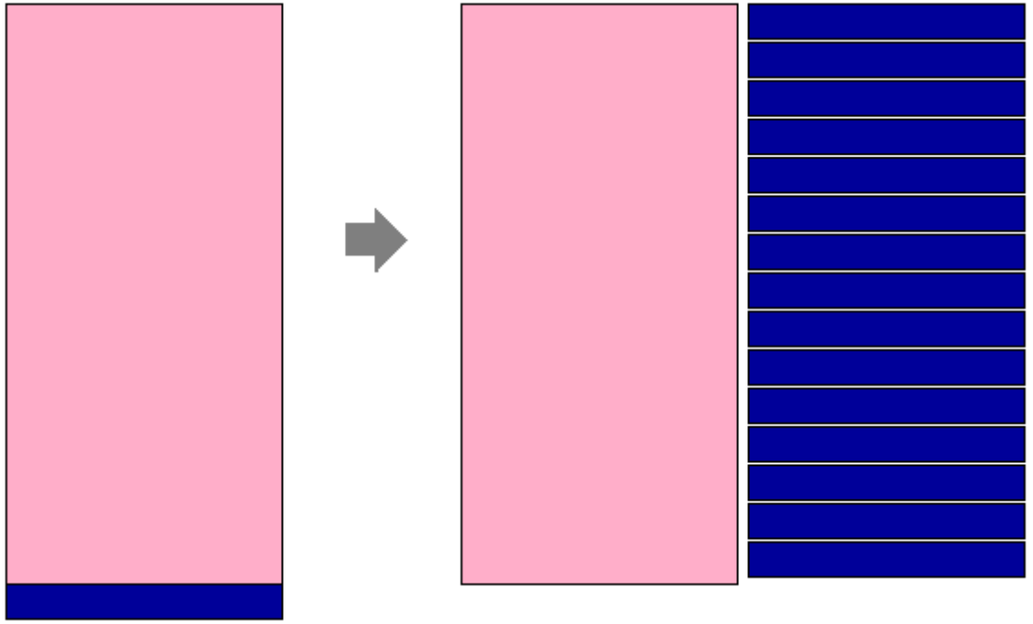
Создаём выборку:

все k позитивных объекта,
случайные k негативных

Модификация – умный отбор объектов

Near miss – отобрать объекты, нужные для разделения классов

Пересэмлирование
(Oversampling the minority class)



Создаём выборку:

позитивные объекты с повторами,
пока их не станет примерно столько,
сколько негативных

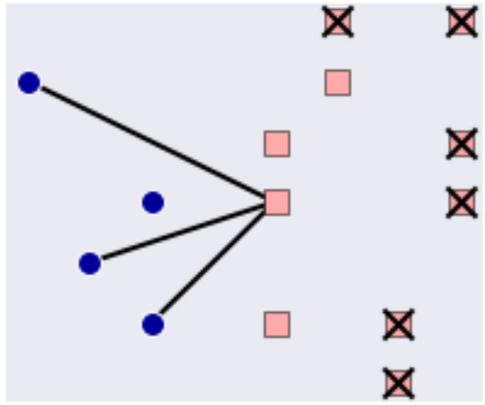
используется больше данных.
качество, как правило, выше

Умные способы недосэмплирования

NearMiss-1

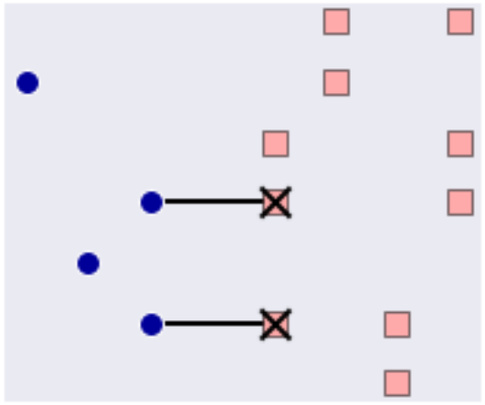
Из большего класса выбираем объекты: среднее расстояние до N ближайших малого класса наименьшее

NearMiss-2



Из большего класса выбираем объекты: среднее расстояние до N дальних малого класса наименьшее

Tomek links



удалить объекты большого класса, образующие связи Томека

Объекты двух разных классов образуют связь Томека, если нет объекта, который ближе к одному из них при этом являясь объектом другого класса

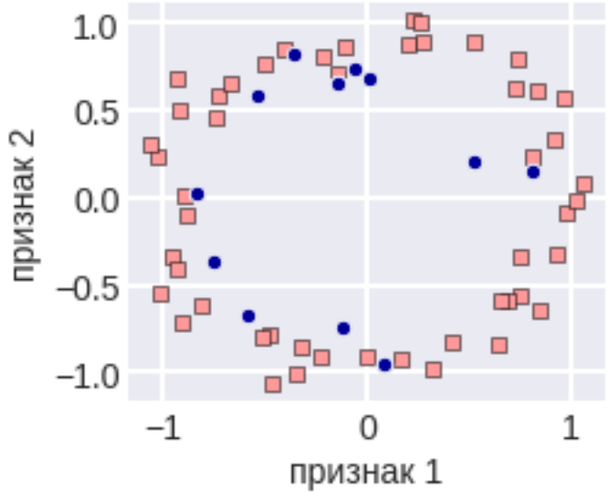
Edited nearest neighbors (ENN)

идея ~ на CV (LOO) посмотреть объекты, на которых ошибки и их удалить

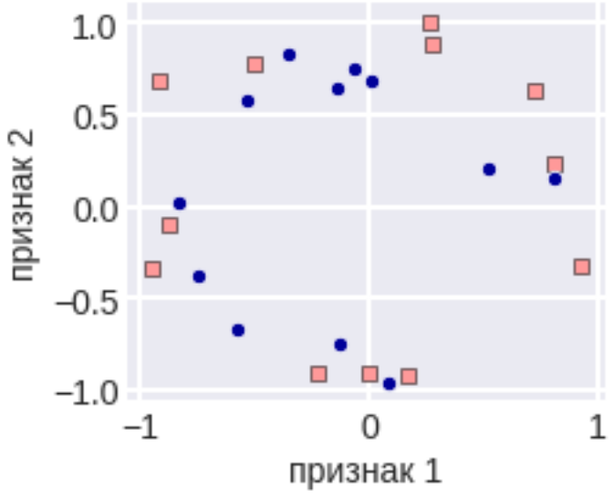
используется метод 1NN

но идею можно обобщить…

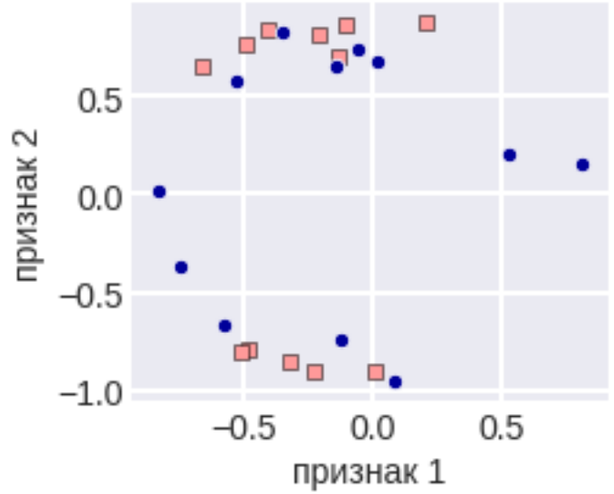
Пример применения недосэмплирования



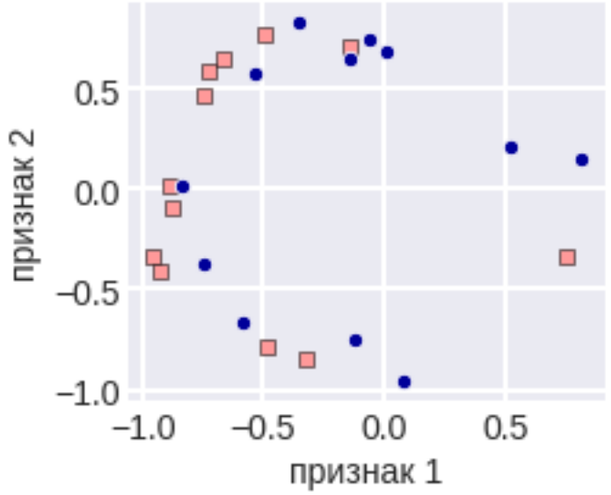
данные



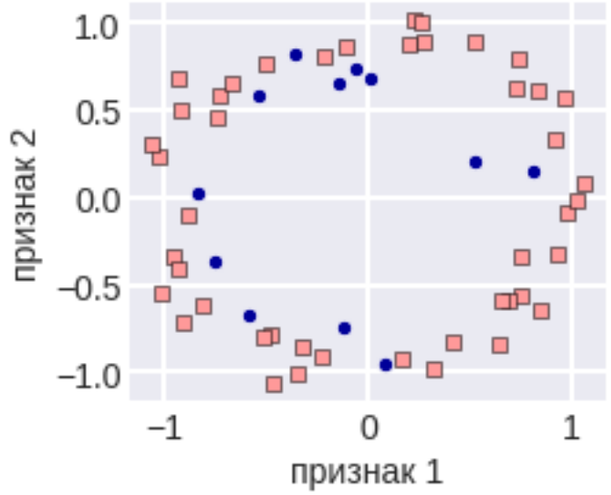
RandomUnderSampler



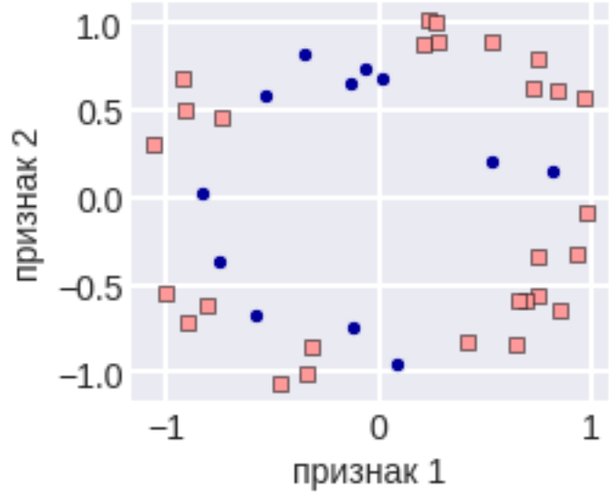
NearMiss(version=1)



NearMiss(version=2)



TomekLinks()



EditedNearestNeighbours()

SMOTE = Synthetic Minority Oversampling Techniques

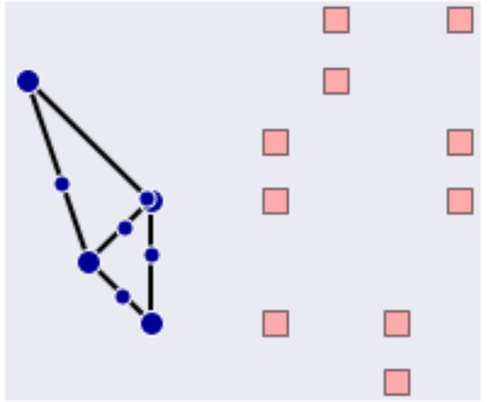
Идея: увеличить малый класс за счёт представителей выпуклых комбинаций пар

Для точки выбирается ≤k-й сосед и на отрезке между ними – новый объект

есть разные модификации, например, все точки делятся на группы, в зависимости от процента «чужих» в окрестности (от этого зависит вероятность порождения нового объекта)

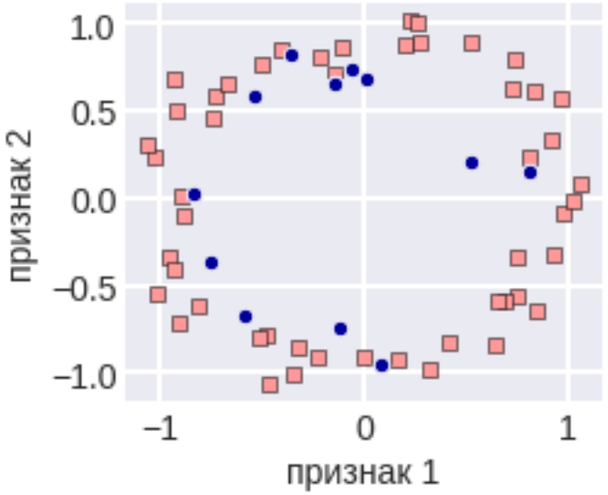
ADASYN = Adaptive Synthetic

Аналогично SMOTE, но число объектов, которые генерятся с помощью x, пропорционально числу чужаков рядом с x (т.е. большую роль играют выбросы)

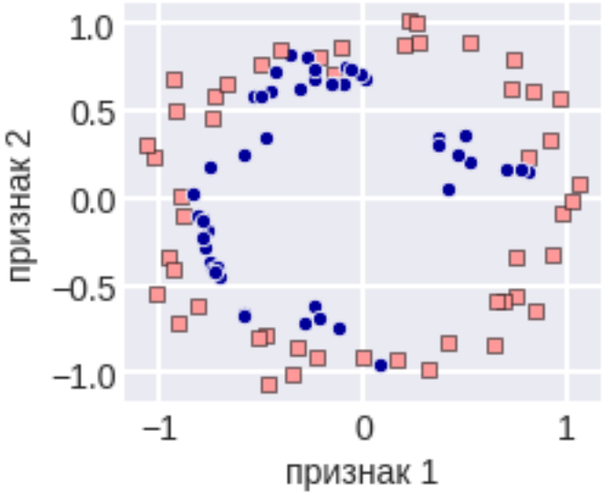


в качестве k ближайших соседей рассматриваются только объекты малого класса

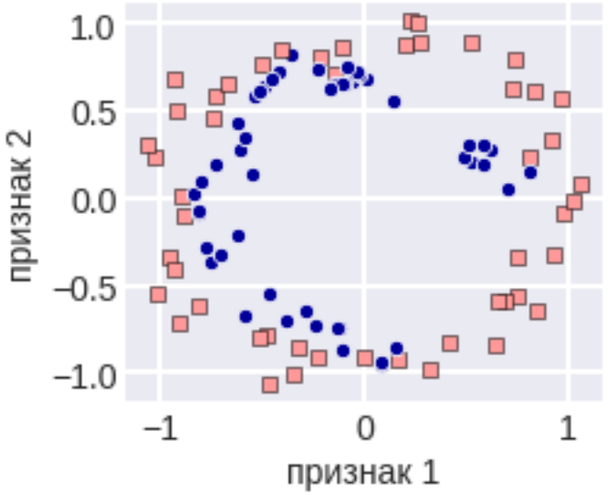
Пример применения пересэмплирования



Данные / RandomOverSampler



SMOTE



ADASYN

Весовые схемы:

негативные примеры – вес =1

позитивные – вес = $m_0 \,/\, m_1$

как правило, самый эффективный приём

в sklearn – `class_weight='balanced'`

Функции ошибки при дисбалансе

Обычная функция будет «затачиваться» под больший класс:

$$\sum_i L(y_i, a(x_i))$$

Взвешивание классов:

$$\sum_i C_{y_i} L(y_i, a(x_i))$$

попарные функции ошибки (~ ROC AUC)

$$L(a(x_i), a(x_j)) = \begin{cases} 0, & y_i < y_j, \\ 1, & y_i > y_j \end{cases}$$

$$\text{AUC}(a) = \frac{1}{m_0 m_1} \sum_{i:y_i=0} \sum_{j:y_j=1} I[a(x_i) < a(x_j)]$$

легко обобщать функции на пары объектов ($w^{\text{T}} x \to w^{\text{T}}(x_j - x_i)$)

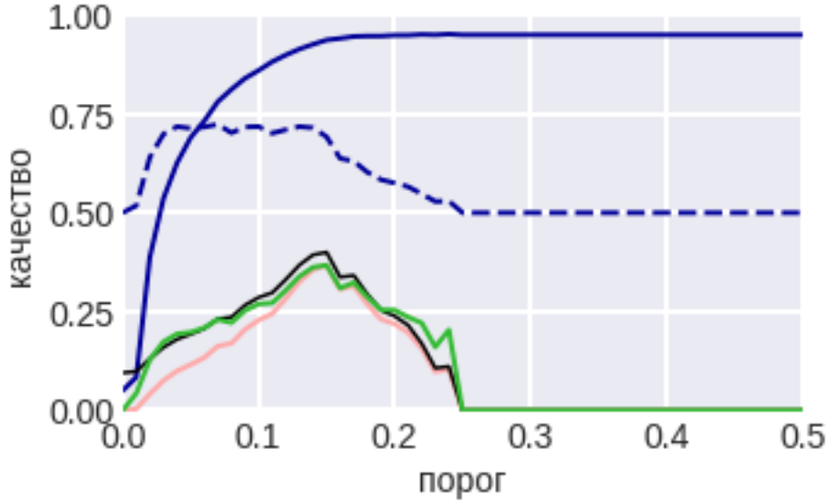
Решающее правило: выбор порога

Обычно модель получает некоторые оценки принадлежности к классам

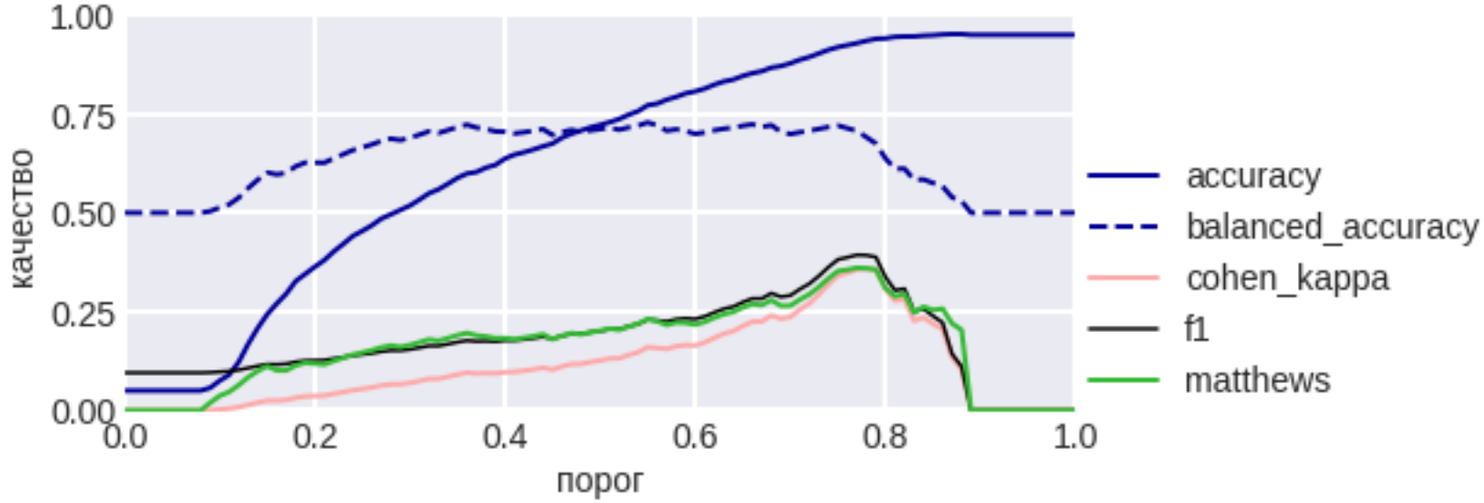
Классификация – результат бинаризации (по умолчанию порог = 0.5)

Но порог можно подбирать – ниже графики качества от порога на CV-контроле по 10 фолдам

обычная модель

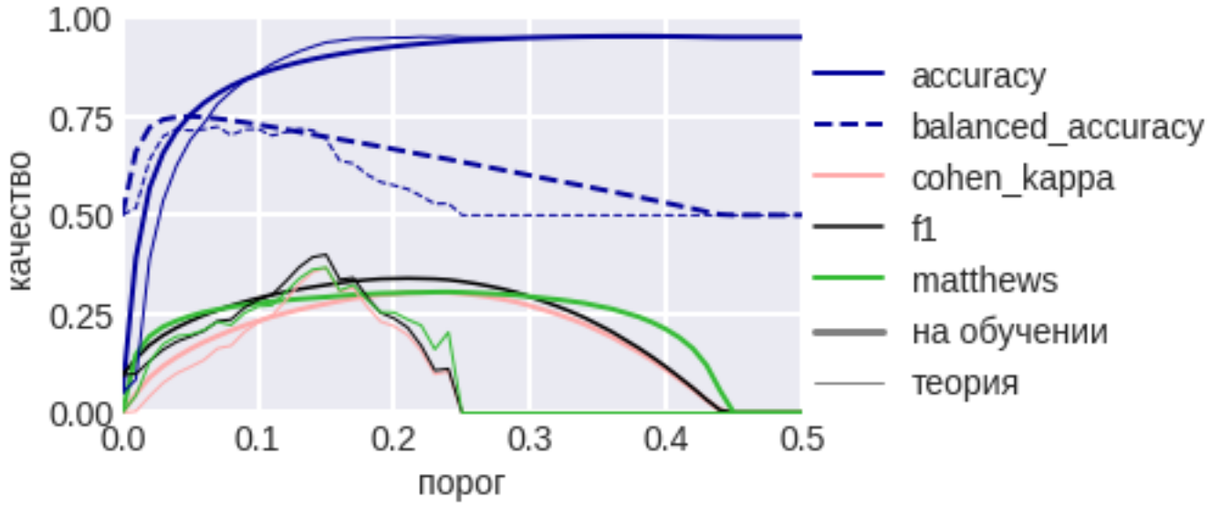


модель с параметром `class_weight='balanced'`



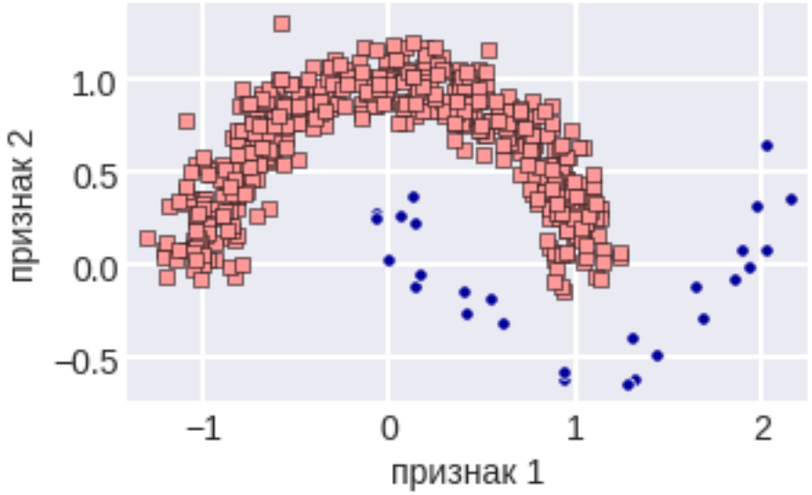
Решающее правило: выбор порога

сравнение наблюдаемых графиков для выбора порогов и теоретических

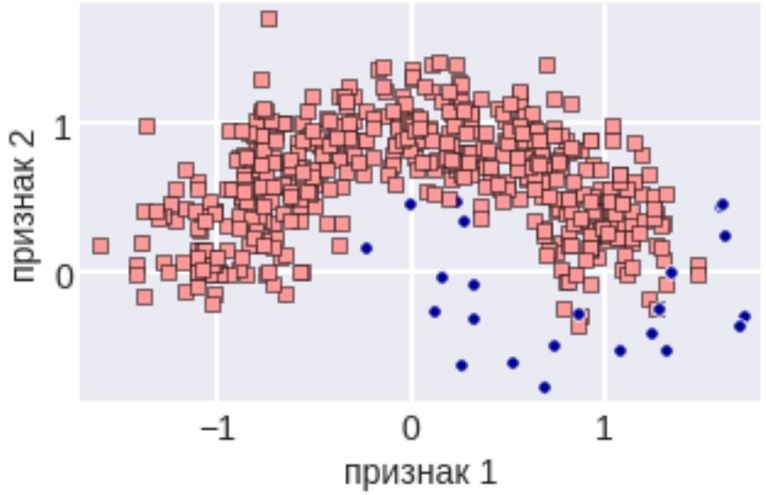


тут неверная легенда

Эксперименты: два полумесяца



задача 1



задача 2

Эксперименты: логистическая регрессия (верх – задача 1, низ – задача 2)

	None	Weights	Th-d	Th-d + W	RandOS	SMOTE	ADASYN	RandUS	NM1	NM2	TLinks	ENNs
accuracy_score	0.970	0.866	0.975	0.974	0.869	0.867	0.939	0.831	0.950	0.895	0.970	0.970
balanced_accuracy_score	0.731	0.881	0.861	0.864	0.880	0.882	0.863	0.857	0.854	0.824	0.731	0.731
cohen_kappa_score	0.619	0.377	0.715	0.714	0.382	0.380	0.558	0.311	0.603	0.397	0.619	0.619
f1_score	0.633	0.431	0.728	0.727	0.435	0.433	0.589	0.372	0.629	0.445	0.633	0.633
matthews_corrcoef	0.669	0.459	0.738	0.732	0.461	0.461	0.578	0.403	0.612	0.441	0.669	0.669
−log_loss	-0.100	-0.291	-0.100	-0.291	-0.283	-0.285	-0.135	-0.361	-0.336	-0.352	-0.100	-0.100
roc_auc_score	0.960	0.962	0.960	0.962	0.962	0.962	0.961	0.956	0.962	0.933	0.960	0.960
average_precision_score	0.788	0.789	0.788	0.789	0.789	0.788	0.784	0.783	0.788	0.704	0.788	0.788

	None	Weights	Th-d	Th-d + W	RandOS	SMOTE	ADASYN	RandUS	NM1	NM2	TLinks	ENNs
accuracy_score	0.961	0.857	0.963	0.962	0.858	0.867	0.843	0.848	0.904	0.882	0.963	0.966
balanced_accuracy_score	0.665	0.872	0.870	0.871	0.872	0.871	0.870	0.859	0.812	0.770	0.684	0.723
cohen_kappa_score	0.477	0.358	0.623	0.612	0.359	0.374	0.336	0.335	0.411	0.329	0.516	0.583
f1_score	0.492	0.414	0.640	0.648	0.415	0.427	0.395	0.394	0.457	0.382	0.531	0.598
matthews_corrcoef	0.551	0.441	0.577	0.569	0.442	0.451	0.427	0.420	0.446	0.366	0.578	0.621
−log_loss	-0.110	-0.316	-0.110	-0.316	-0.313	-0.293	-0.345	-0.356	-0.392	-0.423	-0.108	-0.107
roc_auc_score	0.952	0.952	0.952	0.952	0.952	0.951	0.951	0.948	0.919	0.878	0.952	0.952
average_precision_score	0.722	0.722	0.722	0.722	0.722	0.718	0.720	0.722	0.609	0.439	0.722	0.722

тут достаточно подобрать порог

Эксперименты: градиентный бустинг (верх – задача 1, низ – задача 2)

	None	Weights	Th-d	Th-d + W	RandOS	SMOTE	ADASYN	RandUS	NM1	NM2	TLinks	ENNs
accuracy_score	0.992	0.992	0.992	0.992	0.987	0.991	0.991	0.845	0.966	0.917	0.992	0.992
balanced_accuracy_score	0.972	0.970	0.972	0.970	0.941	0.965	0.963	0.840	0.843	0.859	0.972	0.972
cohen_kappa_score	0.929	0.927	0.929	0.927	0.879	0.921	0.920	0.321	0.682	0.482	0.929	0.929
f1_score	0.934	0.932	0.934	0.932	0.886	0.926	0.925	0.380	0.700	0.522	0.934	0.934
matthews_corrcoef	0.930	0.928	0.930	0.928	0.879	0.922	0.920	0.399	0.682	0.519	0.930	0.930
−log_loss	-0.065	-0.051	-0.065	-0.051	-0.130	-0.088	-0.088	-0.468	-0.442	-0.636	-0.065	-0.065
roc_auc_score	0.988	0.995	0.988	0.995	0.967	0.964	0.983	0.908	0.909	0.845	0.988	0.988
average_precision_score	0.970	0.972	0.970	0.972	0.857	0.889	0.878	0.393	0.695	0.440	0.970	0.970

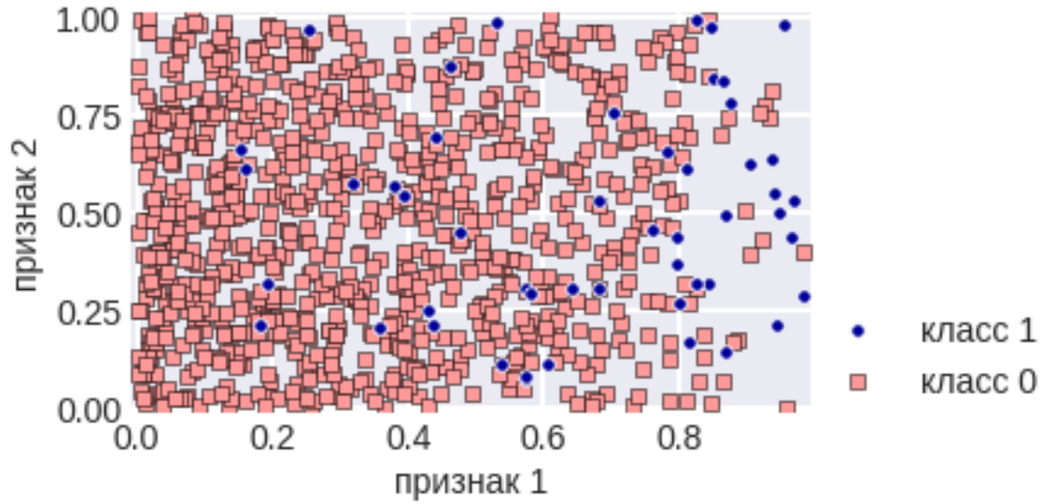
	None	Weights	Th-d	Th-d + W	RandOS	SMOTE	ADASYN	RandUS	NM1	NM2	TLinks	ENNs
accuracy_score	0.983	0.983	0.983	0.983	0.973	0.981	0.980	0.846	0.927	0.264	0.982	0.963
balanced_accuracy_score	0.892	0.902	0.926	0.930	0.843	0.923	0.916	0.840	0.784	0.482	0.901	0.935
cohen_kappa_score	0.832	0.837	0.826	0.832	0.728	0.825	0.817	0.320	0.453	-0.005	0.828	0.715
f1_score	0.841	0.846	0.835	0.841	0.742	0.835	0.828	0.380	0.491	0.101	0.837	0.734
matthews_corrcoef	0.834	0.838	0.829	0.834	0.730	0.825	0.818	0.399	0.465	-0.019	0.828	0.730
−log_loss	-0.103	-0.102	-0.103	-0.102	-0.191	-0.130	-0.140	-0.384	-0.397	-0.699	-0.111	-0.255
roc_auc_score	0.976	0.964	0.976	0.964	0.968	0.962	0.960	0.909	0.885	0.738	0.975	0.976
average_precision_score	0.884	0.874	0.884	0.874	0.837	0.872	0.843	0.449	0.464	0.403	0.878	0.758

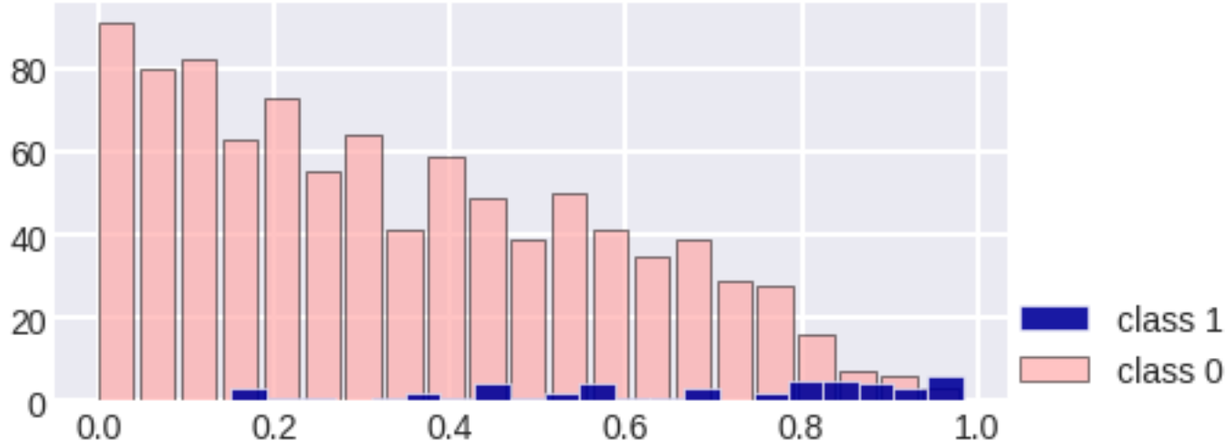
бустинг хорош по умолчанию или с весами

Эксперименты: дизайн

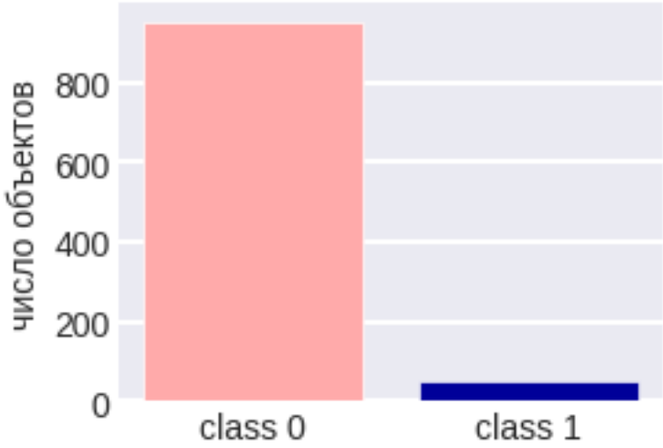
Наша «любимая» задача

объекты





пропорции



– гистограммы классов

Эксперименты: пересэмплирование

	RandomOverSampler		SMOTE		ADASYN		
	train	test	train	test	train	test	record
accuracy_score	0.721	0.752	0.725	0.755	0.710	0.736	0.953
balanced_accuracy_score	0.721	0.749	0.713	0.748	0.715	0.749	0.749
cohen_kappa_score	0.131	0.160	0.128	0.161	0.123	0.152	0.302
f1_score	0.205	0.231	0.203	0.232	0.199	0.225	0.340
matthews_corrcoef	0.210	0.244	0.204	0.244	0.202	0.239	0.303
log_loss	0.522	0.506	0.518	0.499	0.547	0.530	0.161
roc_auc_score	0.820	0.833	0.820	0.832	0.820	0.833	0.833
average_precision_score	0.331	0.309	0.305	0.295	0.330	0.307	0.310

```
from imblearn.over_sampling import RandomOverSampler, SMOTE, ADASYN
techniques = [RandomOverSampler(),
              SMOTE(),
              ADASYN()]
for sampler in techniques:
    X_resampled, y_resampled = sampler.fit_resample(X, y)
    model.fit(X_resampled, y_resampled)
```

Эксперименты: недосэмплирование

	RandomUS		NearMiss(ver=1)		NearMiss(ver=2)		TomekLinks		EditedNNs		
	train	test	train	test	train	test	train	test	train	test	record
accuracy_score	0.724	0.754	0.641	0.613	0.729	0.749	0.950	0.950	0.950	0.950	0.953
balanced_accuracy_score	0.722	0.749	0.603	0.622	0.687	0.668	0.500	0.500	0.500	0.500	0.749
cohen_kappa_score	0.133	0.162	0.052	0.057	0.116	0.113	0.000	0.000	0.000	0.000	0.302
f1_score	0.207	0.232	0.135	0.141	0.191	0.188	0.000	0.000	0.000	0.000	0.340
matthews_corrcoef	0.212	0.245	0.093	0.109	0.180	0.168	0.000	0.000	0.000	0.000	0.303
log_loss	0.543	0.532	0.635	0.643	0.593	0.582	0.163	0.160	0.163	0.160	0.161
roc_auc_score	0.820	0.833	0.690	0.691	0.742	0.740	0.820	0.833	0.820	0.833	0.833
average_precision_score	0.331	0.307	0.150	0.134	0.142	0.165	0.331	0.310	0.330	0.306	0.310

```
from imblearn.under_sampling import RandomUnderSampler, NearMiss, TomekLinks,
                                    EditedNearestNeighbours
techniques = [RandomUnderSampler(), NearMiss(version=1), NearMiss(version=2),
              TomekLinks(), EditedNearestNeighbours()]
for sampler in techniques:
    X_resampled, y_resampled = sampler.fit_resample(X, y)
model.fit(X_resampled, y_resampled)
```

Эксперименты: подбор порога и взвешивание

	train	test	cv	opt_test	theta
accuracy_score	0.950	0.950	0.952	0.953	0.24
balanced_accuracy_score	0.500	0.500	0.724	0.745	0.07
cohen_kappa_score	0.000	0.000	0.367	0.301	0.15
f1_score	0.000	0.000	0.400	0.340	0.15
matthews_corrcoef	0.000	0.000	0.367	0.303	0.15
log_loss	0.163	0.161	0.166	NaN	NaN
roc_auc_score	0.820	0.833	0.808	NaN	NaN
average_precision_score	0.331	0.310	0.311	NaN	NaN

```
LogisticRegression()
```

	train	test	cv	opt_test	theta
accuracy_score	0.722	0.753	0.952	0.952	0.87
balanced_accuracy_score	0.721	0.749	0.728	0.747	0.55
cohen_kappa_score	0.131	0.161	0.358	0.302	0.78
f1_score	0.206	0.232	0.393	0.339	0.77
matthews_corrcoef	0.210	0.244	0.361	0.302	0.77
log_loss	0.521	0.505	0.523	NaN	NaN
roc_auc_score	0.820	0.833	0.804	NaN	NaN
average_precision_score	0.332	0.308	0.306	NaN	NaN

```
LogisticRegression(class_weight='balanced')
```

именно здесь поставлены эти рекорды,
с которыми мы сравнивались

	train	test	cv	opt_test	theta	record
accuracy_score	1.000	0.943	0.952	0.950	0.75	0.953
balanced_accuracy_score	1.000	0.576	0.730	0.662	0.09	0.749
cohen_kappa_score	1.000	0.203	0.378	0.237	0.31	0.302
f1_score	1.000	0.229	0.408	0.275	0.31	0.340
matthews_corrcoef	1.000	0.218	0.378	0.237	0.31	0.303
log_loss	0.033	0.474	0.413	NaN	NaN	0.161
roc_auc_score	1.000	0.726	0.767	NaN	NaN	0.833
average_precision_score	1.000	0.192	0.274	NaN	NaN	0.310

```
RandomForestClassifier(n_estimators=500, max_features=1)
```

	train	test	cv	opt_test	theta	record
accuracy_score	1.000	0.941	0.952	0.950	0.85	0.953
balanced_accuracy_score	1.000	0.578	0.721	0.656	0.10	0.749
cohen_kappa_score	1.000	0.202	0.306	0.210	0.24	0.302
f1_score	1.000	0.229	0.347	0.257	0.24	0.340
matthews_corrcoef	1.000	0.213	0.312	0.215	0.24	0.303
log_loss	0.033	0.533	0.542	NaN	NaN	0.161
roc_auc_score	1.000	0.728	0.743	NaN	NaN	0.833
average_precision_score	1.000	0.183	0.254	NaN	NaN	0.310

```
RandomForestClassifier(n_estimators=500, max_features=2)
```

	train	test	cv	opt_test	theta	record
accuracy_score	0.999	0.934	0.950	0.943	0.84	0.953
balanced_accuracy_score	0.999	0.585	0.707	0.644	0.01	0.749
cohen_kappa_score	0.990	0.196	0.338	0.182	0.06	0.302
f1_score	0.990	0.230	0.375	0.230	0.06	0.340
matthews_corrcoef	0.990	0.199	0.341	0.185	0.06	0.303
log_loss	0.006	0.335	0.286	NaN	NaN	0.161
roc_auc_score	1.000	0.737	0.779	NaN	NaN	0.833
average_precision_score	1.000	0.163	0.246	NaN	NaN	0.310

```
model = LGBMClassifier()
```

	train	test	cv	opt_test	theta	record
accuracy_score	0.956	0.952	0.955	0.951	0.53	0.953
balanced_accuracy_score	0.588	0.555	0.724	0.714	0.05	0.749
cohen_kappa_score	0.276	0.179	0.426	0.296	0.18	0.302
f1_score	0.290	0.192	0.455	0.333	0.18	0.340
matthews_corrcoef	0.354	0.243	0.427	0.296	0.18	0.303
log_loss	0.138	0.165	0.156	NaN	NaN	0.161
roc_auc_score	0.880	0.798	0.791	NaN	NaN	0.833
average_precision_score	0.434	0.254	0.334	NaN	NaN	0.310

```
model = LGBMClassifier(num_leaves=2)
```

Итог

если умеете решать задачу...

если Вы понимаете геометрию задачи и правильно подбираете модель,
если она хорошо обучена (и откалибрована),
то достаточно подбирать порог

Устранение дисбаланса в DL

Есть широкий класс практически важных задач с дисбалансом:
детектирование объектов,
близость объектов

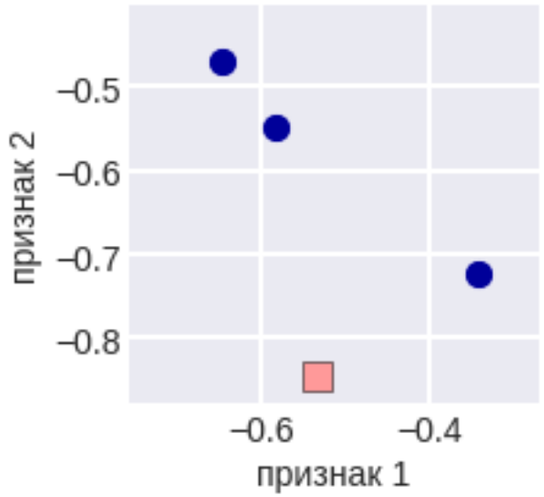
- **организация батчей**

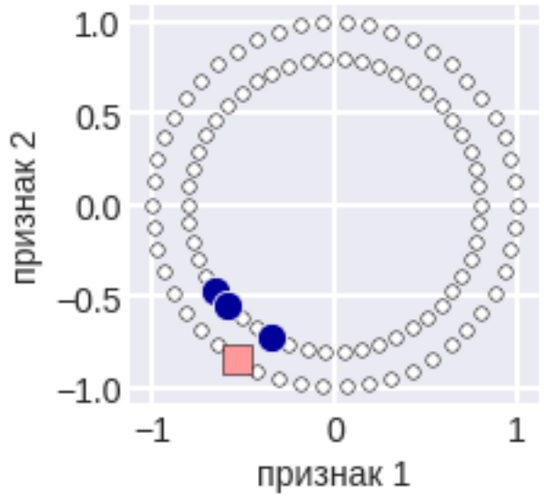
сэмплирование на уровне батчей, Negative Sampling

- **специальные функции ошибки**

Пример: focal loss

Обучение с частичной разметкой: Semi-supervised Learning (SSL)





В обучении размеченные данные $X_{\text{s}} = (x_i, y_i)_{i=1}^{m}$ **и неразмеченные данные** $X_{\text{u}} = \{x_i\}_{i=m+1}^{m+u}$

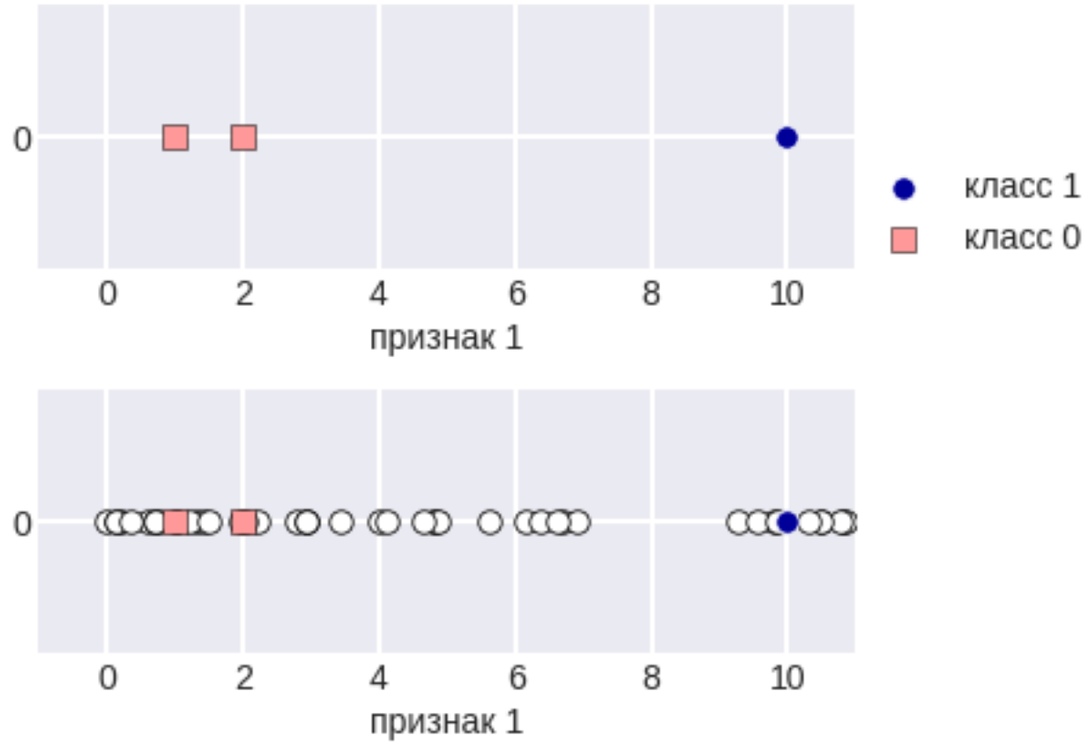
обычно $u >> m$

Если вторая часть является тестовой выборкой, то это Transductive Learning

Алгоритм надо построить используя всю имеющуюся информацию

Почему? – Проставление меток обычно очень дорого (пример: работа ассесоров, бурение новых скважин, ожидание поломок и т.д.)

Обучение с частичной разметкой: Semi-supervised Learning (SSL)



Идеи, которые дальше будут встречаться:

- соседние точки имеют одинаковые метки
- точки из одного кластера имеют одинаковые метки
- граница классов там, где плотность точек маленькая

SSL с помощью кластеризации

«Дешёвый способ» – сделать кластеризацию, пополнить признаковое пространство характеристическими признаками кластеров

Cluster-then-Label

- **сделать кластеризацию**
- **для каждого кластера**
  - **если в кластере есть размеченные объекты, обучить на них классификатор**
  - **если нет – обучить на всех данных**
  - **с помощью классификатора разметить неразмеченные объекты кластера**

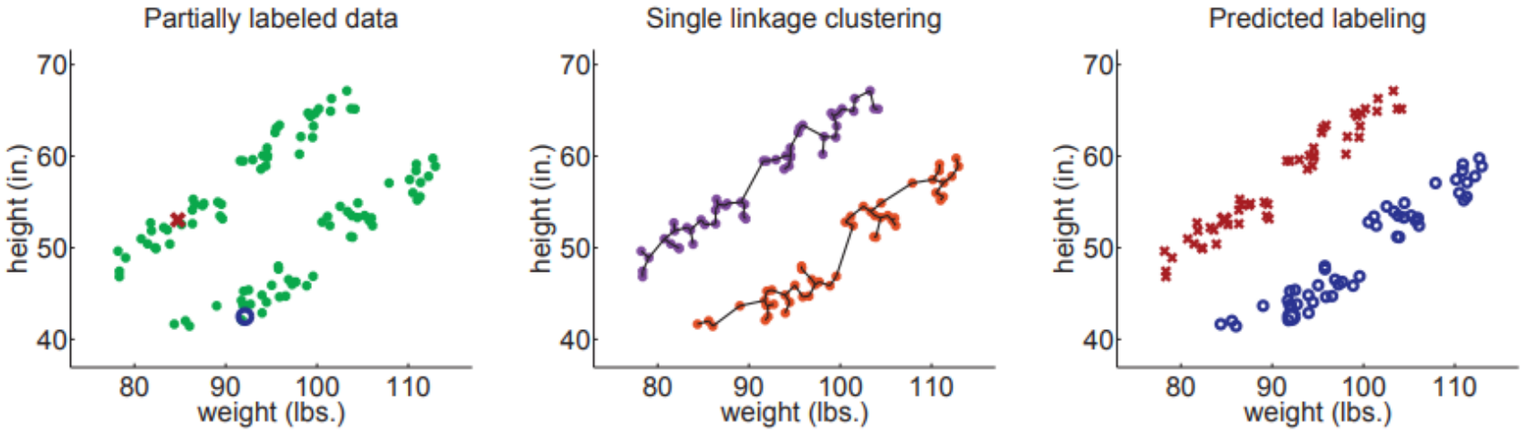


Figure 3.5: Cluster-then-label results using single linkage hierarchical agglomerative clustering ($\mathcal{A}$) and majority vote ($\mathcal{L}$).

SSL с помощью кластеризации: модификация k-means

Есть «переделки» алгоритмов кластеризации для SSL:

Вход: $X_{\text{s}} = (x_i, y_i)_{i=1}^{m}$ **+** $X_{\text{u}} = \{x_i\}_{i=m+1}^{m+u}$

Инициализация: $k = l$, $C_t = \{x_i \in X_{\text{s}} \mid y_i = t\}$

Итерация:

1. Пересчитать центры кластеров: $\mu_t = \frac{1}{|C_t|} \sum_{i \in C_t} x_i$

2. Каждый неразмеченный объект приписать к тому кластеру, к центру которого он ближе: $C_t = \{x_i \in X_{\text{u}} \mid \| x_i - \mu_t \| = \min_s \| x_i - \mu_s \|\}$

Self-Training (Yarowsky Algorithm)

идея: по кусочкам размечаем данные

1. Инициализация обучающей выборки (размеченные данные)

$$X_{\text{t}} \leftarrow X_{\text{s}} = (x_i, y_i)_{i=1}^{m}$$

2. Обучение (на размеченных данных)

$$a = \text{fit}(X_{\text{t}})$$

3. Разметка всех данных

$$(t, p) = a(X_{\text{u}})$$

здесь p **– уверенности в ответах**

4. Пополнение выборки объектами с уверенными ответами

$$X_{\text{t}} \leftarrow X_{\text{t}} \cup \{(x, a(x)) \,|\, x \in X_{\text{u}}, p(x) > \theta\}$$

$$X_{\text{u}} \leftarrow X_{\text{u}} \setminus X_{\text{t}}$$

порог θ можно подбирать так, чтобы какой-то процент объектов переходил в X_{t}

5. Если не вся выборка размечена или мало шагов – переход к п. 2

под уверенностью можно много чего понимать, **см. активное обучение**

Частный случай Self-Training: Propagating 1-Nearest-Neighbor

Заметим, что Self-Training зависит от применяемой модели

Пока есть неразмеченные объекты

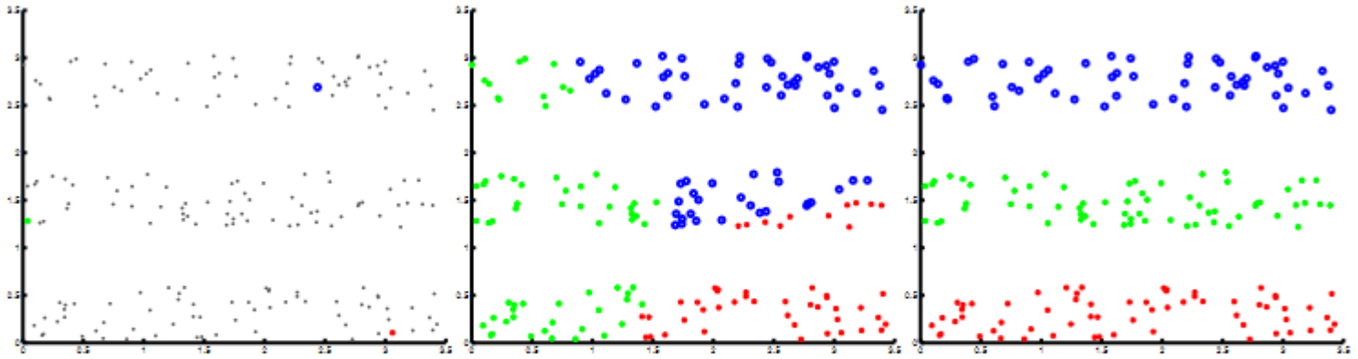
- **найти ближайший к размеченным неразмеченный объект**
- **пометить его меткой ближайшего размеченного соседа**

Есть обобщения с весовыми схемами,

подобная реализация есть в sklearn

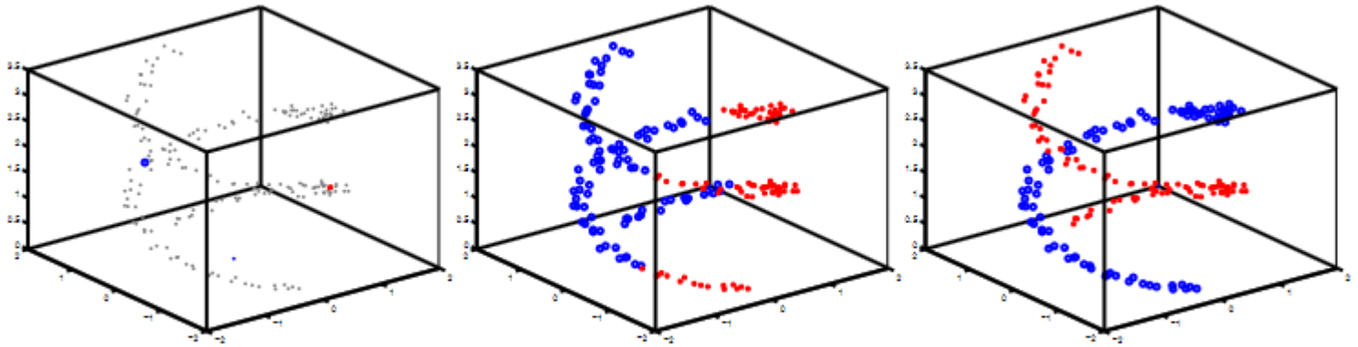
```
sklearn.semi_supervised.LabelPropagation

kernel='rbf' - ядро knn или rbf
gamma=20 - параметр ядра
n_neighbors=7 - число соседей
max_iter=1000 - число итераций
```

Частный случай Self-Training: Propagating 1-Nearest-Neighbor



(a) The data (b) kNN (c) Label propagation

Figure 1: The 3 Bands dataset. Labeled data are color symbols and unlabeled data are dots in (a). kNN ignores unlabeled data structure, while label propagation uses it.



(a) The data (b) kNN (c) Label propagation

Figure 2: The Springs dataset.

Xiaojin Zhu and Zoubin Ghahramani «Learning from labeled and unlabeled data with label propagation» // http://pages.cs.wisc.edu/~jerryzhu/pub/CMU-CALD-02-107.pdf

SSL с графовой регуляризацией (Graph-based Regularization)

строится граф, например k-соседства

идея: соседи на графе имеют схожие метки

$$\sum_i L(y_i, a(x_i)) + \lambda R(a) + \gamma \sum_{(i,j)\in E} w_{ij}(a(x_i) - a(x_j))^2 \to \min$$

функция ошибки + регуляризатор + ошибка на графе

первая сумма – по размеченным объектам

последняя – по всем

Это общий подход в SSL:

- **штраф за классификацию**
- **штраф за кластеризацию**
- **регуляризатор**

Общий подход: LogReg + Entropy Regularization

$$\sum_{i=1}^{m} \log(1+\exp(-y_i a(x_i))) + \lambda R(a) + \gamma \sum_{i=m+1}^{m+u} H(1/(1+\exp(-a(x_i))))$$

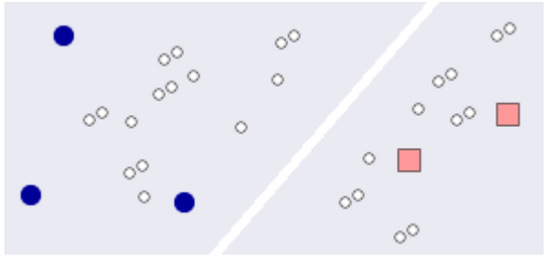
т.е. это логистическая регрессия, у которой на всех объектах уверенные ответы

+ очень простая идея

– когда классы хорошо разделимы

Transductive SVM / Semi-Supervised Support Vector Machines (S3VMs)

идея: разделяющая поверхность не лежит в регионах с большой плотностью



Классический метод SVM

$$\frac{\|w\|^2}{2} \to \min$$

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1,\ i \in \{1,2,\ldots,m\}$$

Трансдуктивный метод SVM

$$\frac{\|w\|^2}{2} \to \min_{w,\{\overline{y}_i\}_{i=m+1}^{m+u}}$$

$$y_i(w^{\mathrm{T}}x_i + b) \geq 1,\ i \in \{1,2,\ldots,m\}$$

$$\overline{y}_i(w^{\mathrm{T}}x_i + b) \geq 1,\ i \in \{m+1,\ldots,m+u\}$$

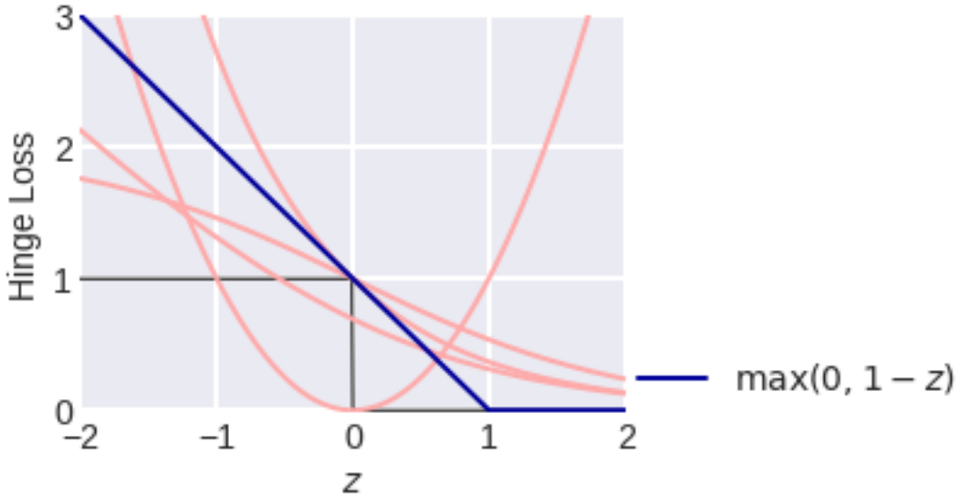
$$\overline{y}_i \in \{\pm 1\},\ i \in \{m+1,\ldots,m+u\}$$

Transductive SVM / Semi-Supervised Support Vector Machines (S3VMs)

что получится при решении…

Классический метод SVM

$$\xi_i = \max[1 - y_i(w^{\mathrm{T}} x_i + b), 0]$$

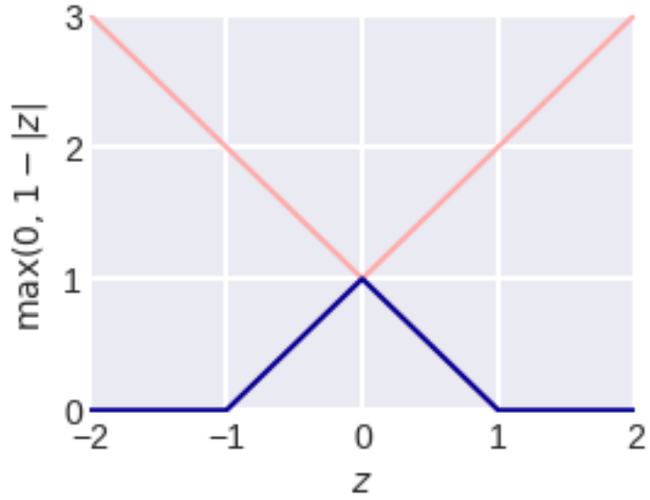


Трансдуктивный метод SVM

$$\xi_i = \max[1 - y_i(w^{\mathrm{T}} x_i + b), 0],\ i \in \{1, 2, \ldots, m\}$$

$$\xi_i = \max[1 - \bar{y}_i \cdot (w^{\mathrm{T}} x_i + b), 0],\ i \in \{m+1, \ldots, m+u\}$$

$$\bar{y}_i \in \{\pm 1\},\ i \in \{m+1, \ldots, m+u\}$$



функция $\max[1 - |\, w^{\mathrm{T}} x_i + b \,|, 0]$ **– хороший штраф за попадание в полосу – «the hat loss»**

Transductive SVM / Semi-Supervised Support Vector Machines (S3VMs)

$$\sum_{i=1}^{m} \max[1 - y_i(w^{\mathrm{T}} x_i + b), 0] + \frac{1}{2C} \| w \|^2 + \gamma \sum_{i=m+1}^{m+u} \max[1 - | y_i(w^{\mathrm{T}} x_i + b) |, 0] \to \min$$

получается как в стандартной схеме

+ можно использовать ядра

– сложная невыпуклая оптимизация

– неинтуитивные параметры

– когда классы хорошо разделимы

термины: Transductive vs. Inductive

Transductive – получить метки неразмеченных данных

Inductive – получить алгоритм разметки

EM для SSL

Обучение: $(x_i, y_i)_{i=1}^{m}$ **+** $\{x_i\}_{i=m+1}^{m+u}$ **(можно с весами)**

$$p(D \mid \theta) = \prod_{i=1}^{m} p(x_i, y_i \mid \theta) \prod_{i=m+1}^{m+u} p(x_i \mid \theta) = \prod_{i=1}^{m} p(y_i \mid \theta)\, p(x_i \mid y_i, \theta) \prod_{i=m+1}^{m+u} \sum_{y_j} p(x_i, y_j \mid \theta)$$

неизвестные: $\{y_i\}_{i=m+1}^{m+u}$ **(латентные переменные) и** θ **(параметры)**

EM

E-шаг: по известному (текущему) θ **оценить** $\{y_i\}_{i=m+1}^{m+u}$

M-шаг: MLE для оценки θ

Всё сильно зависит от гипотез о распределениях

[Dempster et al 1977]

SSL: co-training

Упрощённо это Self-Training с двумя алгоритмами

Algorithm 4.1. Co-Training.

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, *unlabeled data* $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, *a learning speed* k.

Each instance has two views $\mathbf{x}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}]$.

1. Initially let the training sample be $L_1 = L_2 = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$.

2. Repeat until unlabeled data is used up:

3. Train a view-1 classifier $f^{(1)}$ *from* L_1, *and a view-2 classifier* $f^{(2)}$ *from* L_2.

4. Classify the remaining unlabeled data with $f^{(1)}$ *and* $f^{(2)}$ *separately.*

5. Add $f^{(1)}$*'s top* k *most-confident predictions* $(\mathbf{x}, f^{(1)}(\mathbf{x}))$ *to* L_2.

Add $f^{(2)}$*'s top* k *most-confident predictions* $(\mathbf{x}, f^{(2)}(\mathbf{x}))$ *to* L_1.

Remove these from the unlabeled data.

Blum, Mitchell // https://www.cs.cmu.edu/~avrim/Papers/cotrain.pdf

Активное обучение – Active Learning

«Ухудшение» ситуации с Semi-Supervised Learning

Когда данные не размечены, разметка может быть выполнена, но для небольшого множества – есть возможность его сформировать

Разметка возможна, но дорога, не будет полноценного CV

Общая схема

- **выбирается подвыборка и размечается** (есть стратегия выбора)
- **строится модель / модели**
- **повторяется п.1**

Active Learning: стратегии выбора

- **Uncertainty Sampling / уточнение модели**

неуверенно классифицируемые примеры

- **Disagreement Sampling / комитетом алгоритмов**

примеры, по которым расходится мнения алгоритмов ансамбля

- **Expected model change / Expected error reduction**

анализируем влияние на модель

- **Density-Based Samling / на основе кластеризации**

используем структуру пространства

- **Maximal Diversity Sampling**

чтобы не размечать похожие

- **RL**

про это не будем

- **Acquisition functions (байесовская оптимизация)**

про это не будем

- **AL + SSL**

понятно как...

Active Learning: мотивация

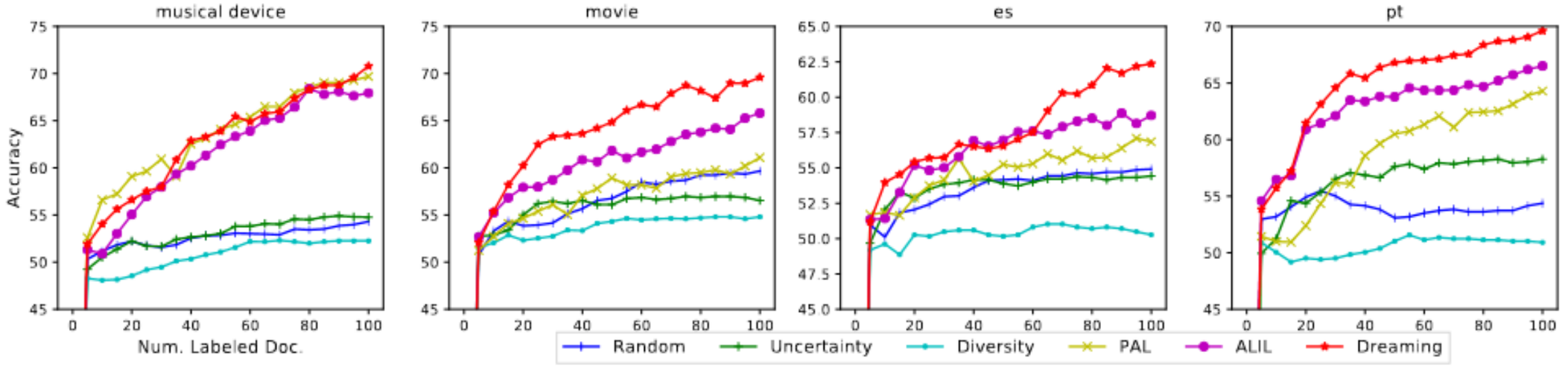


Figure 2: Accuracy of different active learning methods for cross domain sentiment classification (left two plots) and cross lingual authorship profiling (right two plots).

сравнение разных стратегий разметки

можно не размечать все примеры…

Thuy-Trang Vu «Learning How to Active Learn by Dreaming» //
https://www.aclweb.org/anthology/P19-1401.pdf

Active Learning: чем хорошо

два нормально распределённых класса: логистическая регрессия на случайном датасете из 30 точек и на специально выбранным для разметки

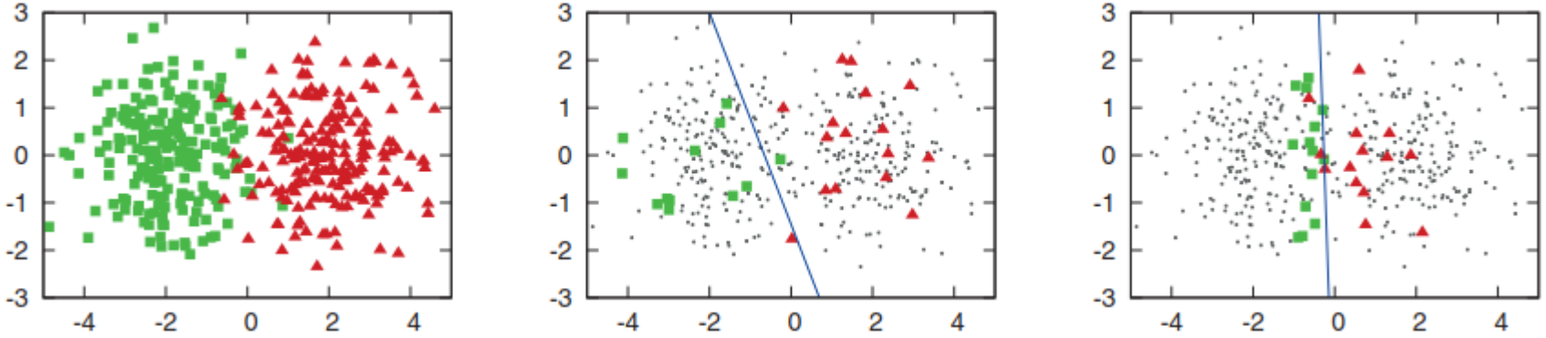


Figure 2.2: Uncertainty sampling with a toy data set. (a) 400 instances, evenly sampled from two class Gaussians. Instances are represented as points in a 2D input space. (b) A logistic regression model trained with 30 labeled instances randomly drawn from the problem domain. The line represents the decision boundary of the classifier. (c) A logistic regression model trained with 30 actively queried instances using uncertainty sampling.

экспоненциальное сокращение размеченных данных

пример: поиск разделения на прямой с помощью дихотомии

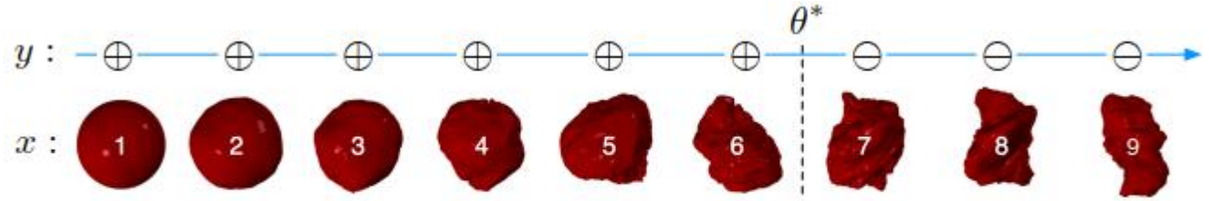


Figure 1.2: Supervised learning for the alien fruits example. Given a set of $\langle x, y \rangle$ instance-label pairs, we want to choose the threshold θ^* that classifies them most accurately.

Active Learning: проблемы

«Sampling bias»

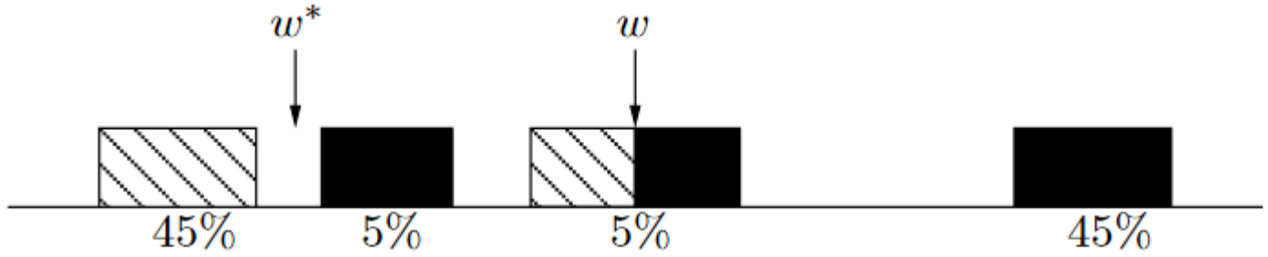


Figure 3: An illustration of sampling bias in active learning. The data lie in four groups on the line, and are (say) distributed uniformly within each group. The two extremal groups contain 90% of the distribution. Solids have a + label, while stripes have a − label.

Sanjoy Dasgupta «Two faces of active learning» //
http://cseweb.ucsd.edu/~dasgupta/papers/twoface.pdf

Уточнение модели для AL / Uncertainty Sampling

Иногда название – Model hypothesis space refinement – устранение неопределённостей

Надо выбирать из разметки те точки, для которых у нас максимальная неопределённость «uncertainty measure» согласно текущей модели

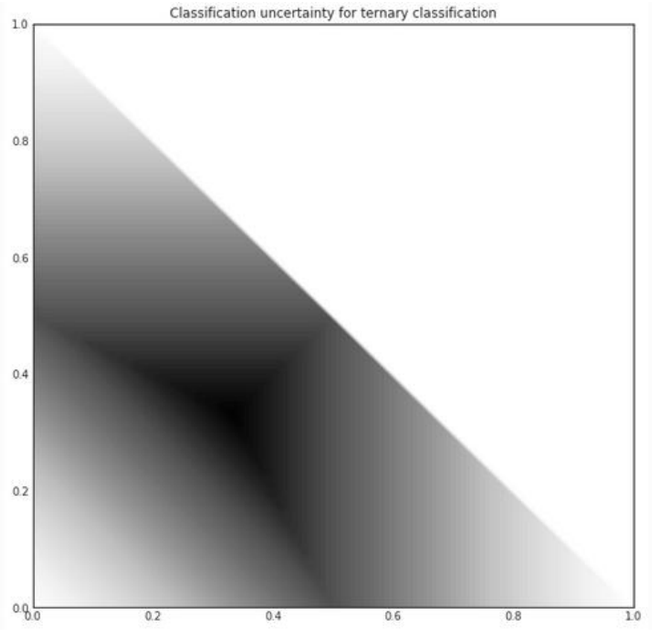
Виды мер неопределённости «uncertainty measures»

если алгоритм получает вероятности классов

$$a(x) = (b_1(x), \ldots, b_l(x))$$

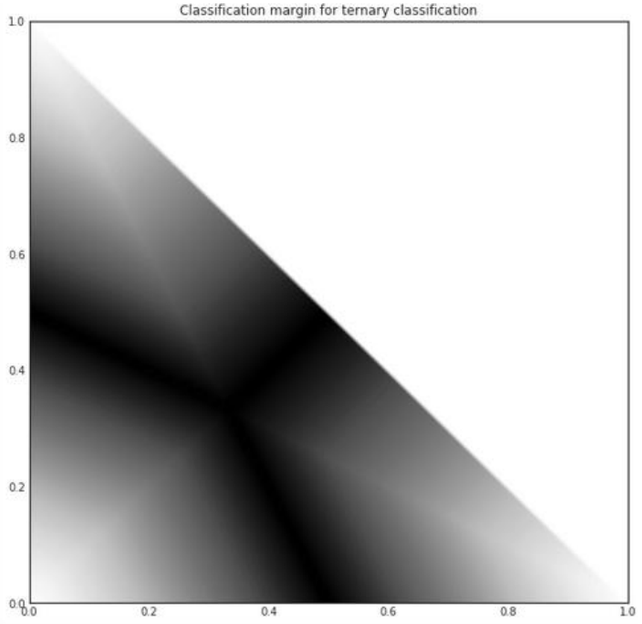
Classification uncertainty / Least Confident

$$U(x) = 1 - \max_i b_i(x)$$



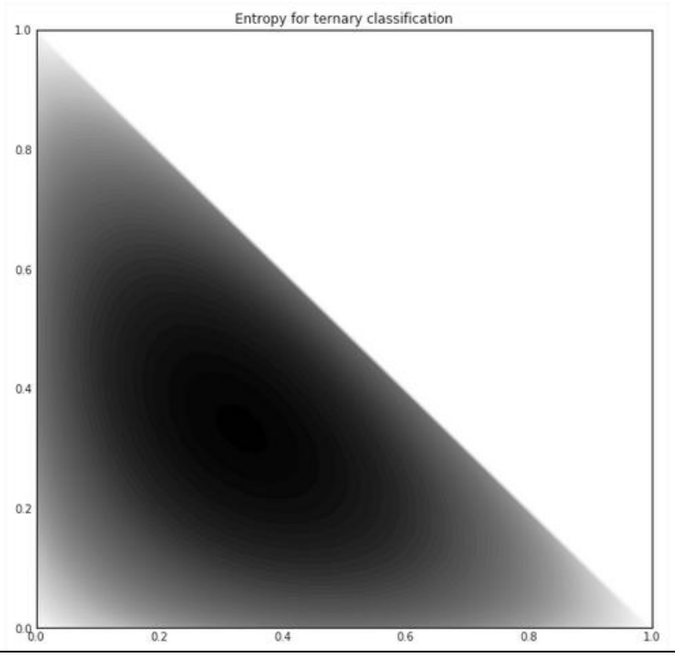
(Classification) margin

$$M(x) = b_{\text{top}(1)}(x) - b_{\text{top}(2)}(x)$$

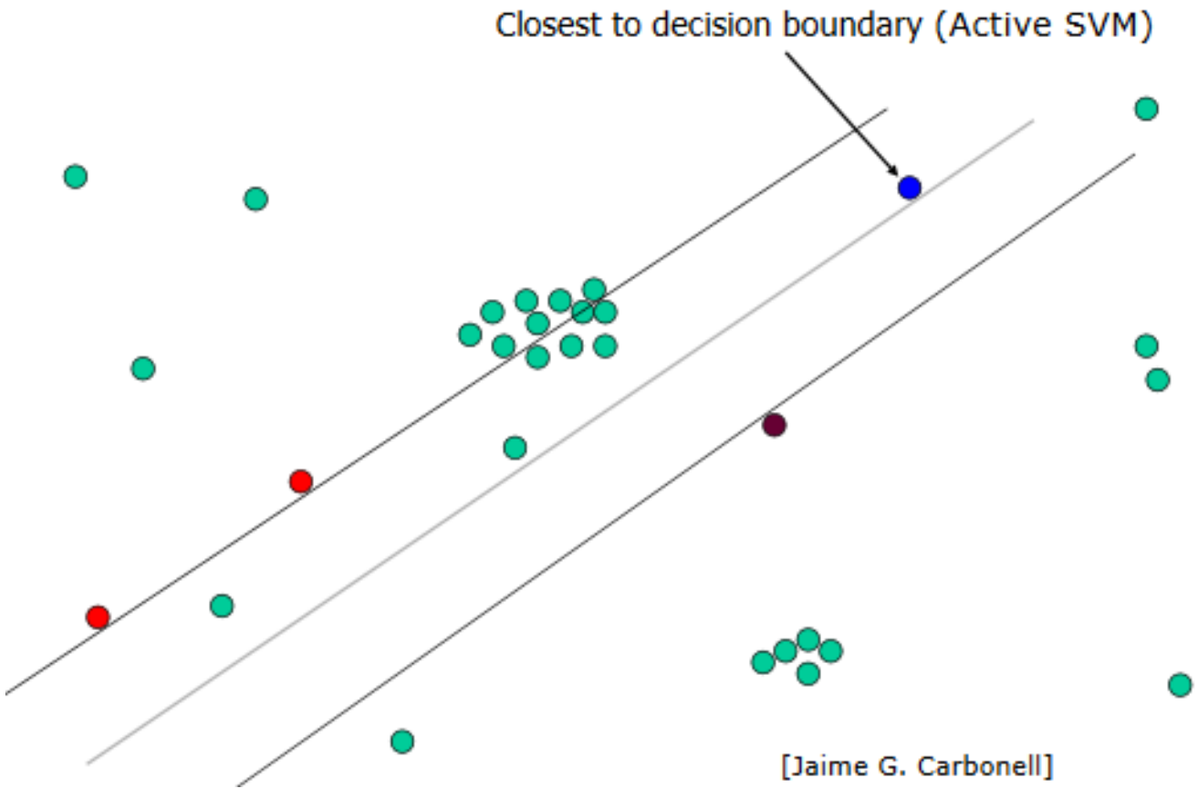


(Classification) entropy

$$H(x) = -\sum_i b_i(x) \log b_i(x)$$



Уточнение модели для AL / Uncertainty Sampling



Active SVM

- **вычисляем текущий разделитель**
- **выбираем точку ближайшую к разделителю**

Может привести к sampling bias.

+ заточено на определённую модель

– модель может быть выбрана неверно

[Tong & Koller, ICML 2000; Jain, Vijayanarasimhan& Grauman, NIPS 2010; SchohonCohn, ICML 2000]

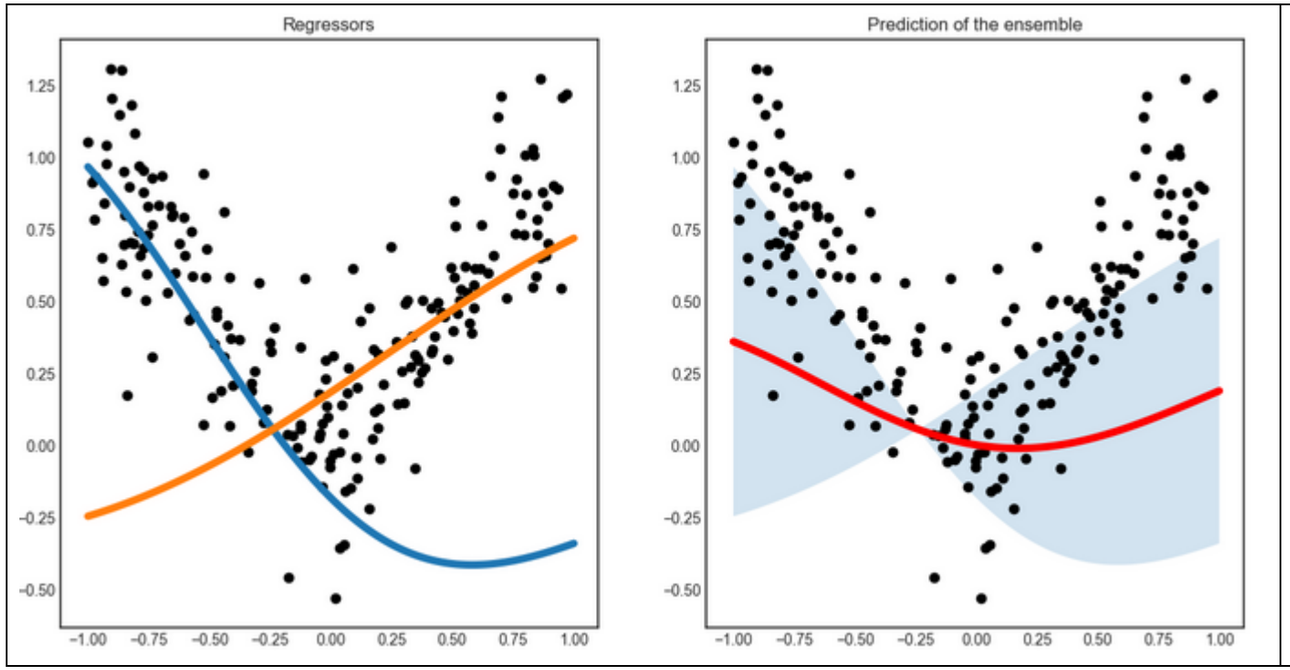
Disagreement Sampling / с помощью комитета алгоритомов

- строится ансамбль, точнее несколько разных алгоритмов ML

(м.б. трюки для разнообразия, например bagging)

- находим примеры, на которых согласие представителей ансамбля самое низкой

т.е. «мнения алгоритмов максимально расходятся»



+ одновременно исследуем разные подходы к решению задачи
+ можно применять в онлайн-режиме
– более долгое обучение

Формализация мер разногласия для классификации

$$b^t(x) = (b_1^t(x), \ldots, b_l^t(x)),\ a^t(x) = \arg\max_j b_j^t(x),\ t \in \{1, 2, \ldots, k\}$$

Vote entropy

$$H\left(\frac{1}{k}(|\,t : a^t(x) = 1\,|, \ldots, |\,t : a^t(x) = l\,|)\right)$$

энтропия на чётких ответов алгоритмов

Consensus entropy

$$H\left(\frac{1}{k}\sum_{t=1}^{k}(b_1^t(x), \ldots, b_l^t(x))\right)$$

энтропия на усреднённом векторе-ответов вероятностей классов

Max disagreement

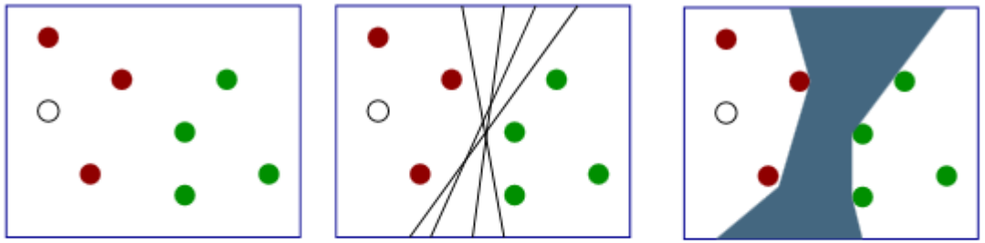
$$\mathrm{MD}(x)$$

отклонение от среденего

$$\mathrm{MD}(x) = \frac{1}{k}\sum_{i=1}^{k}\mathrm{KL}\left((b_1^i(x), \ldots, b_l^i(x)), \frac{1}{k}\sum_{t=1}^{k}(b_1^t(x), \ldots, b_l^t(x))\right)$$

для регрессии понятно, что надо использовать std

Disagreement Sampling / с помощью комитета алгоритмов



Онлайн режим

- **Поступает новая точка x**
- **проверяем согласованность ансамбля на ней**
- **если низкая – размечаем, переобучаем ансамбль**

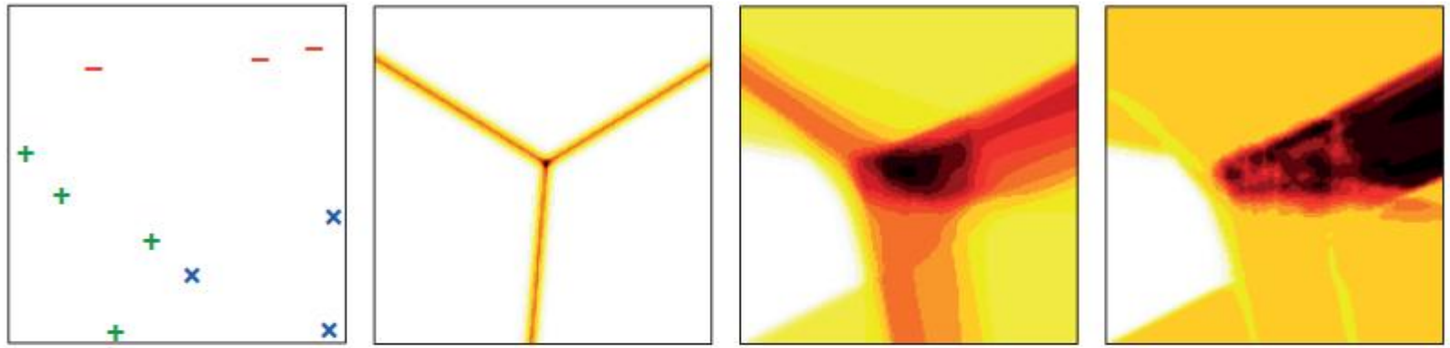


Figure 3.6: Visualization of different active learning heuristics. (a) A small training set with three classes in 2D. (b) A heatmap of the input space according to entropy-based uncertainty sampling. Darker regions are deemed more informative by the MAP-estimate logistic regression classifier. (c) Soft vote entropy using a committee of ten logistic regression classifiers obtained by bagging. (d) KL divergence using the exact same committee of ten classifiers.

Expected model change / Expected error reduction

Если можно оценить – разметка какой точки наиболее сильно повлияет на модель или уменьшит ошибку

например, если можно вычислить $\| \nabla L(y(x), a(x \mid w)) \|$

выбираем объект для разметки так:

$$\sum_y b_y(x) \| \nabla L(y, a(x \mid w)) \| \to \max$$

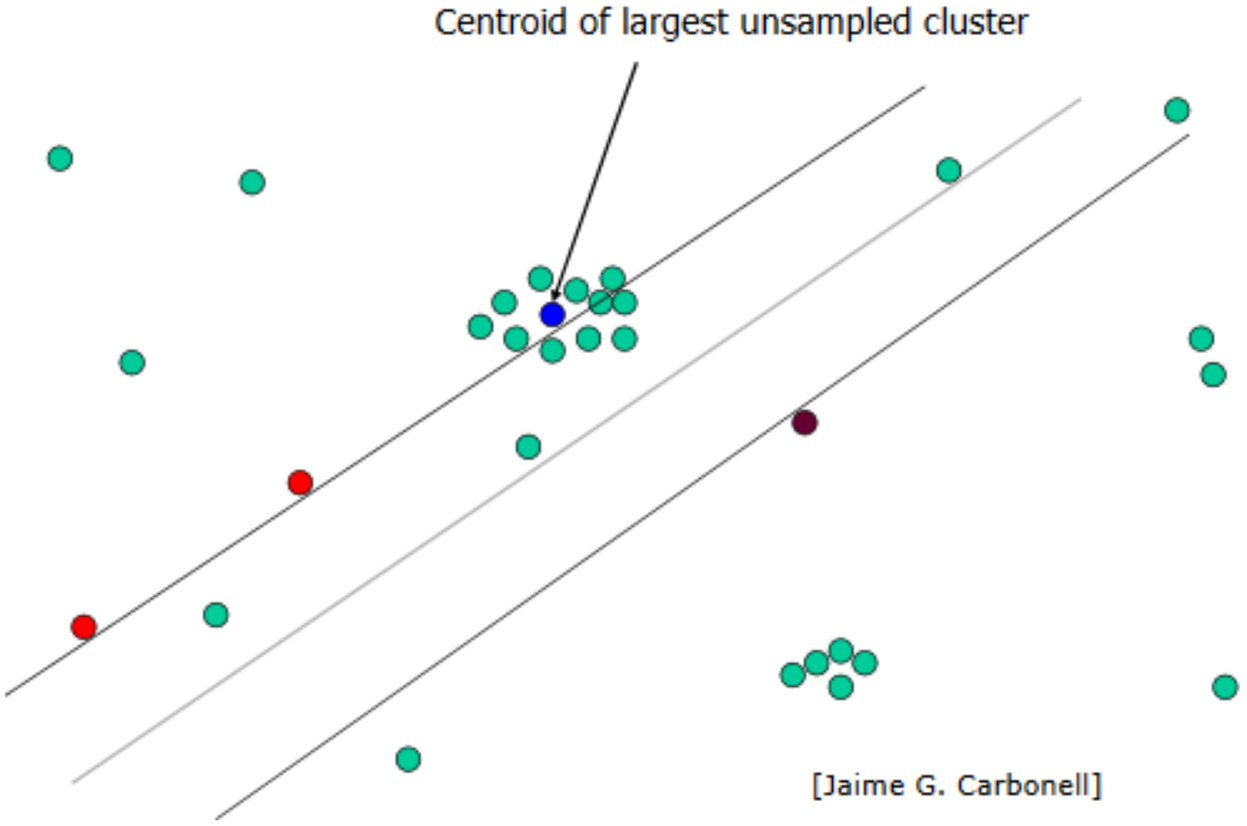
+ учитываем функцию ошибки

– не всегда можно оценить

– есть опасность включать выбросы в разметку

Density-Based Sampling / AL на основе кластеризации

- делаем кластеризацию
- размечаем представителей кластеров



+ самая естественная идея

+ не зависит от модели

– не факт, что есть кластерная структура

– не факт, что метки представителей обобщаются на кластер

Можно в указанные выше ф-лы добавлять множитель

$$\left(\frac{1}{m} \sum_{i=1}^{m} \text{sim}(x, x_i) \right)^{\beta}$$

Density-Based Sampling / AL на основе кластеризации

Иерархичечское сэмплирование (hierarchical sampling)

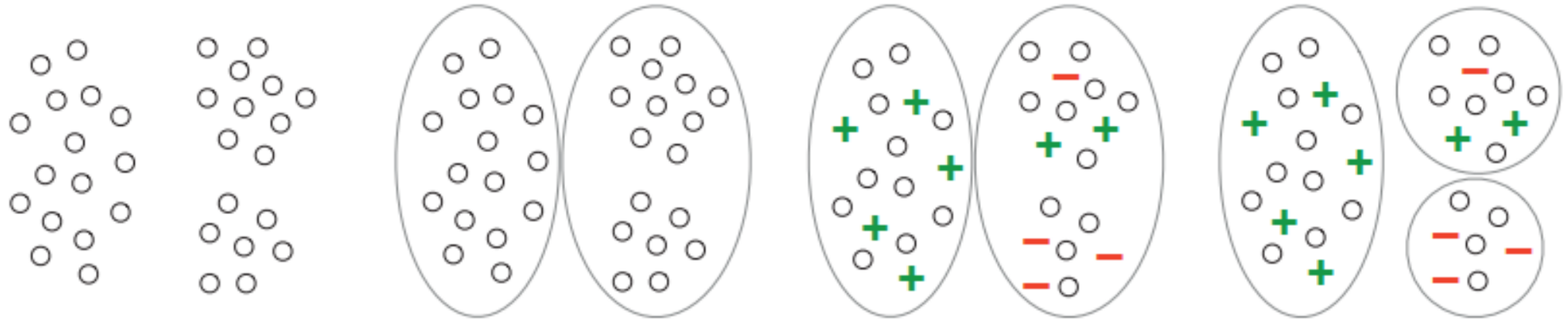
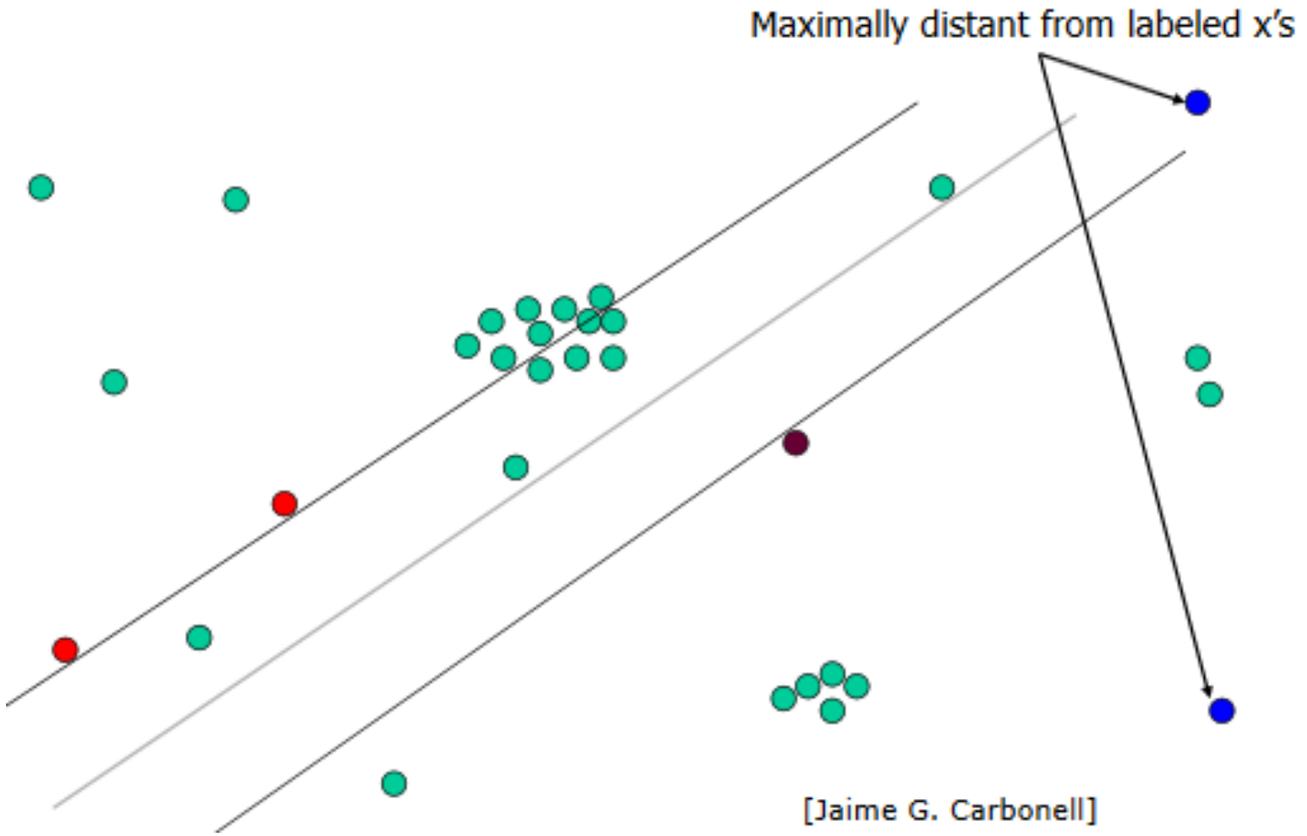


Figure 5.4: The basic stages of active learning by hierarchical sampling.

точное описание не даём

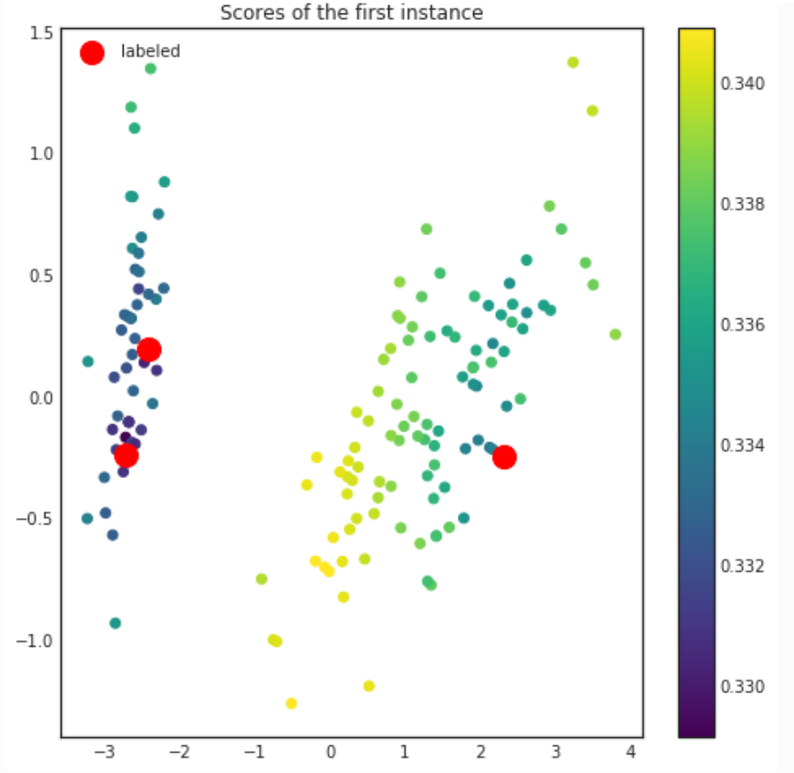
Стратегии AL: Maximal Diversity Sampling



Ranked batch-mode sampling = Uncertainty + Maximal Diversity Sampling

Хотим, чтобы наша модель была максимально не уверена в классификации + максимально не похож на размеченную часть выборки

$$\alpha(1-\text{sim}(x, X_{l\text{abeled}}))+(1-\alpha)U(x)\to\max$$



Стратегии AL: ансамбли

сразу разные стратегии

Ковариальный сдвиг – Covariate Shift

Когда распределения на обучении и тесте различны…

Часто делают так…

$$\sum_{t=1}^{m} \frac{p_{\text{test}}(x_t)}{p_{\text{train}}(x_t)} L(y_t, a(x_t)) + R(a) \to \min$$

почему…

$$E_{(x,y)\sim p_{\text{test}}} L(y, a(x)) = E_{(x,y)\sim p_{\text{train}}} \left[\frac{p_{\text{test}}(x,y)}{p_{\text{train}}(x,y)} L(y, a(x)) \right]$$

если $p(y \mid x)$ **не меняется, то**

$$\frac{p_{\text{test}}(x,y)}{p_{\text{train}}(x,y)} = \frac{p_{\text{test}}(x)}{p_{\text{train}}(x)}$$

ну и очень просто… достаточно знать отношение плотностей, а не сами плотности

Если алгоритм показывает плохое качество

- **изменить признаки**
  - **больше (генерация)**
  - **меньше (отбор, сокращение размерности)**
  - **другие (генерация, преобразование)**

- **изменить объекты**
  - **больше (досбор, доразметка, активное обучение)**
  - **меньше (поиск аномалей, отбор объектов)**
  - **другие (если меняется распределение)**

- **изменить модель / обучение**
  - **проще (линейную), регуляризация, ранний останов**
  - **сложнее (нелинейную, ансамбль), увеличить время обучения**
  - **другую, другой метод оптимизации**
  - **изменить (настроить) гиперпараметры**

Ссылки

«Learning from imbalanced data»

https://www.jeremyjordan.me/imbalanced-data/

Полезный ноутбук по теме

https://github.com/jeremyjordan/imbalanced-data/blob/master/Learning%20from%20imbalanced%20data.ipynb

Другие стратегии сэмплирования

https://basegroup.ru/community/articles/imbalance-datasets

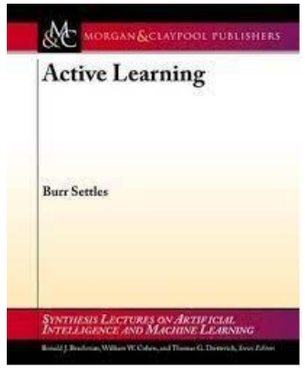
Библиотека imblearn

https://imbalanced-learn.org/stable/

Ссылки

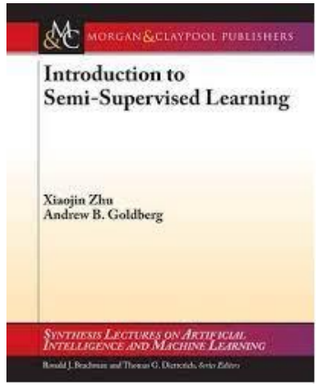
Библиотека по активному обучению

https://modal-python.readthedocs.io



Лучшая книга (но старая) по активному обучению

Burr Settles «Active Learning»



Лучшая книга (но старая) по обучению с частичной разметкой

Xiaojin Zhu, Andrew B. Goldberg «Introduction to Semi-Supervised Learning»