



# ТЕХНОСФЕРА

## Лекция 6 Semi-supervised Learning

Владимир Гулин  
<https://goo.gl/SmqIOz>

11 марта 2016 г.

# План лекции

Мотивация

Semi-supervised learning

Generalized Regularization Theory

Semi-supervised Trees

Semi-supervised Boosting

# Мотивация

## Проблемы применения обучения с учителем на практике

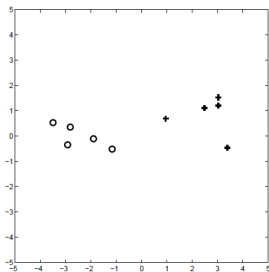
- ▶ В реальной жизни размеченных данных крайне мало
- ▶ Неразмеченных очень много
- ▶ Для хороших моделей нужно много данных
- ▶ Много данных долго собирать
- ▶ Много данных стоят дорого
- ▶ Чаще всего эти данные низкого качества
- ▶ Одно из решений проблемы активное обучение

## Вопрос:

- ▶ Можем ли мы как-то использовать неразмеченные данные для обучения?

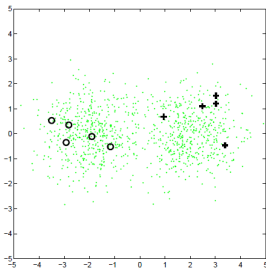
# Мотивация

Где провести границу между классами?

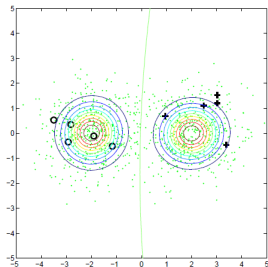
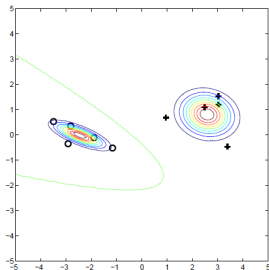


# Мотивация

Где провести границу между классами?



# Мотивация

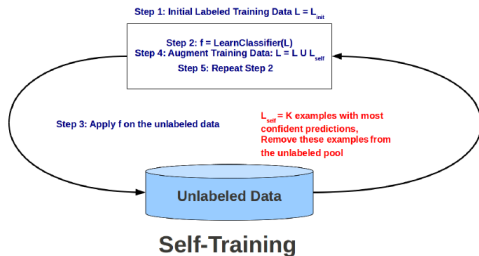


# Semi-supervised learning

- ▶ **Дано:** Множество размеченных данных  $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{N_L}$ ,  
множество неразмеченных данных  $\mathcal{U} = \{\mathbf{x}_j\}_{j=1}^{N_U}$
- ▶ Обучаем алгоритм  $a(\mathbf{x})$  для предсказания любой точки  $\mathbf{x}_0 \in \mathcal{X}$
- ▶ Это называется задачей индуктивного обучения
- ▶ Задача трансдуктивного обучения: Множество размеченных данных  $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{N_L}$ , множество неразмеченных данных  $\mathcal{U} = \{\mathbf{x}_j\}_{j=1}^{N_U}$ .
- ▶ Все возможные внешние данные находятся в  $\mathcal{U}$
- ▶ Все точки из  $\mathcal{U}$  доступны в момент обучения

# Простой алгоритм: Self training

- ▶ **Дано:** Небольшое количество размеченных данных
- ▶ **Идея:** Обучаться, предсказывать и переобучаться, используя свои собственные предсказания, в которых наиболее уверены

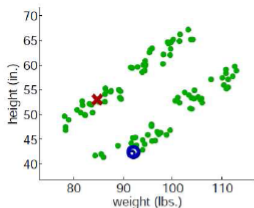


- ▶ Можно использовать с любым алгоритм обучения с учителем. Неплохо работает на практике
- ▶ Если в разметке есть ошибки, то они будут многократно усилены

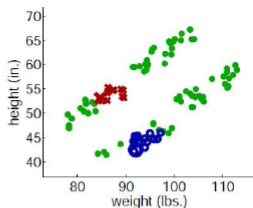


# Self training: Хороший случай

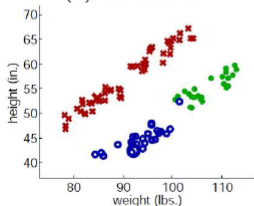
- ▶ Базовый классификатор: KNN



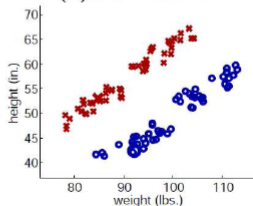
(a) Iteration 1



(b) Iteration 25



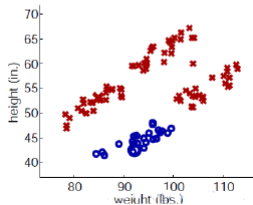
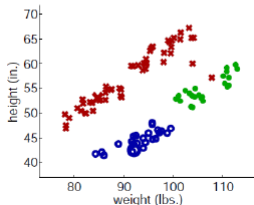
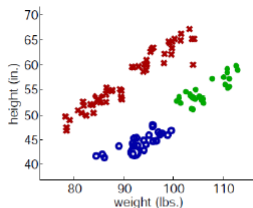
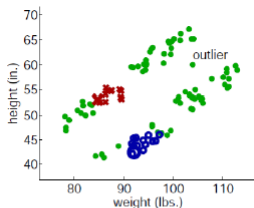
(c) Iteration 74



(d) Final labeling of all instances

# Self training: Плохой случай

- ▶ Базовый классификатор: KNN
- ▶ Если в данных есть выбросы, то все может сломаться



# Expectation Maximization Approach

- ▶ Размеченное  $\mathcal{L}$  и неразмеченное  $\mathcal{U}$  множества
- ▶ Expectation Maximization based Semi-Supervised learning
  - ▶ Построить начальную модель, используя **только размеченные данные  $\mathcal{L}$**  (MLE)

$$\theta = \text{TrainModel}(\mathcal{L})$$

- ▶ Использовать данную модель для оценки метки класса для каждого  $\mathbf{x}_j \in \mathcal{U}$  (считаем ожидание метки класса) Для бинарной классификации  $\{-1, +1\}$

$$E[y_j] = +1 \cdot P(y_j = +1|\theta, \mathbf{x}_j) + (-1) \cdot P(y_j = -1|\theta, \mathbf{x}_j)$$

- ▶ **Перестроить** модель используя  $\mathcal{L}$  и  $\mathcal{U}$  с оценками меток классов
- ▶ Использовать новую модель  $\theta$  для вычисления оценок меток  $E[y]$  на  $\mathcal{U}$
- ▶ Повторять пока не сойдется
- ▶ **Общая схема, которая может быть использована с любой вероятностной моделью**

# Co-training

- ▶ Размеченное  $\mathcal{L}$  и неразмеченное  $\mathcal{U}$  множества
- ▶ Два существенно различных метода обучения  $\mu_1$  и  $\mu_2$ , порождающих алгоритмы  $a_1(\mathbf{x})$  и  $a_2(\mathbf{x})$ 
  - ▶ разные наборы признаков
  - ▶ разные обучающие наборы
  - ▶ разные алгоритмы обучения
- ▶ **Идея:** Алгоритмы поочередно обучают друг друга
- ▶ **Замечание:** Хотим добиться статистической независимости между ответами алгоритмов и размечать только те, в которых более всего уверены

# Co-training

- ▶ Размеченное  $\mathcal{L}$  и неразмеченное  $\mathcal{U}$  множества
- ▶ Создать **два размеченных датасета**  $\mathcal{L}_1$  и  $\mathcal{L}_2$  используя два различных представления
- ▶ **Обучить** классификатор  $a_1$  используя  $\mathcal{L}_1$  и классификатор  $a_2$  используя  $\mathcal{L}_2$
- ▶ **Применить**  $a_1$  и  $a_2$  на неразмеченном пуле данных  $\mathcal{U}$ 
  - ▶ Предсказания делаются только при помощи тех представлений, которые использовались для обучения
- ▶ **Добавить**  $K$  наиболее уверенных предсказаний  $(\mathbf{x}, a_1(\mathbf{x}))$  классификатора  $a_1$  в  $\mathcal{L}_2$
- ▶ **Добавить**  $K$  наиболее уверенных предсказаний  $(\mathbf{x}, a_2(\mathbf{x}))$  классификатора  $a_2$  в  $\mathcal{L}_1$
- ▶ **Как измерить уверенность?**
- ▶ **Удалить** данные примеры из неразмеченного пула данных  $\mathcal{U}$
- ▶ **Перестроить**  $a_1$  используя  $\mathcal{L}_1$  и  $a_2$  используя  $\mathcal{L}_2$
- ▶ В итоге, использовать модель голосования двух моделей при предсказании на тестовых данных

# Multi-view Learning

- ▶ Общий класс алгоритмов частичного обучения
- ▶ Обобщение метода co-training
- ▶ Вместо 2-х используем  $T$  алгоритмов
- ▶ **Ключевая идея:** Большое количество независимых алгоритмов должны соглашаться на неразмеченном примере
- ▶ **Bonus:** На тестовых данных можно использовать композицию из обученных моделей

# Semi-supervised Learning vs Active Learning

- ▶ Пытаются зафиксировать одну и ту же проблему

**uncertainty sampling**  
query instances the model  
is least confident about



**self-training**  
**expectation-maximization (EM)**  
**entropy regularization (ER)**  
propagate confident labelings  
among unlabeled data

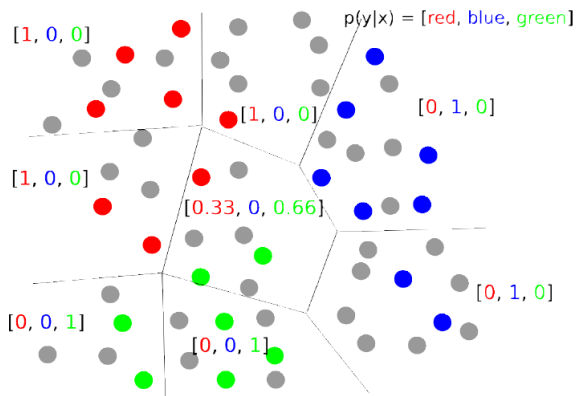
**query-by-committee (QBC)**  
use ensembles to rapidly  
reduce the version space



**co-training**  
**multi-view learning**  
use ensembles with multiple views  
to constrain the version space  
w.r.t. unlabeled data

# Cluster and Label Approach

## Интуиция





# Cluster and Label Approach

**Input:**  $\mathcal{L}$ ,  $\mathcal{U}$ , алгоритм кластеризации  $k$ , алгоритм обучения с учителем  $a$

1. Кластеризовать все данные  $\mathcal{L} \cup \mathcal{U}$  используя  $k$
2. Для каждого кластера выбрать размеченные данные в нем  $S$ :
3. Обучить на этих данных алгоритм  $a$ :  $a_S$
4. Разметить этим алгоритмом  $a_S$  оставшиеся неразмеченные данные

**Output:** Метки неразмеченных данных  $y_1, \dots, y_U$

- **Предположение:** Кластера согласуются с истинными разделяющими поверхностями, иначе все плохо!!!

# Overfitting

## Источники переобучения

- ▶ “Шумные” признаки данных
- ▶ Смещенная обучающая выборка
- ▶ Слишком сложная модель
- ▶ “Шумные” ответы для обучающих данных
- ▶ Выбросы в данных
- ▶ Все возможные комбинации выше перечисленного

# Борьба с переобучением

## Способы борьбы с переобучением

- ▶ Выбрать простую модель
- ▶ Использовать критерий ранней остановки в алгоритме оптимизации
- ▶ Добавить “шум” в обучающие данные
- ▶ Сделать модель более “гладкой” (регуляризация)

# Regularization

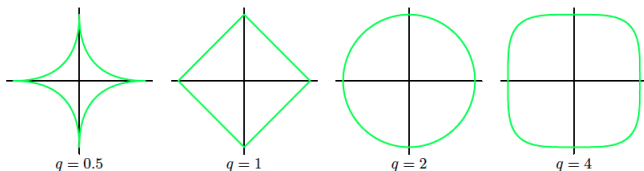
## Linear regression with regularization

$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$  - линейная модель

$$err(h) = \sum_{i=1}^N (h(x_i) - y_i)^2$$

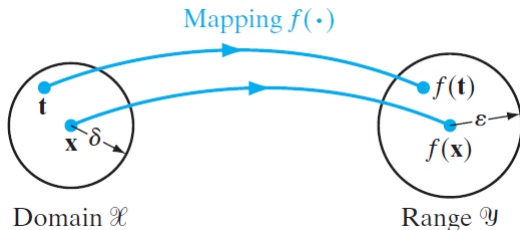
## Регуляризация

$$err(h) = \sum_{i=1}^N (h(x_i) - c_i)^2 + \lambda \|\mathbf{w}\|_q$$



# Корректно поставленные задачи

Hadamard (1902)

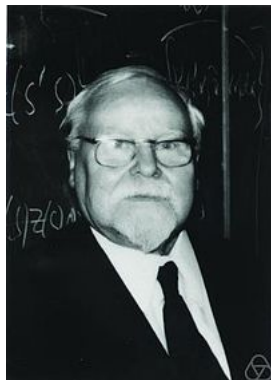


- ▶ Решение существует
- ▶ Решение единственно
- ▶ Решение непрерывно зависит от входных данных

Является ли задача обучения с учителем корректно поставленной?

# Теория регуляризации Тихонова

- ▶ В 1967 году предложил метод решения некорректно поставленных задач
- ▶ Идея заключалась в введении вспомогательного неотрицательного функционала, учитывающего априорную информацию о решении



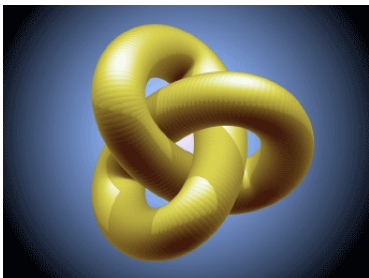
$$err(h) = \frac{1}{2} \sum_{i=1}^N (y_i - h(\mathbf{x}_i))^2 + \frac{1}{2} \|\mathbf{D}h\|^2$$

**D** - линейный дифференциальный оператор, несущий априорную информацию о решении

# Manifold assumption

## Идея

Будем предполагать, что данные лежат на некотором многообразии меньшей размерности, включенном в Евклидово пространство

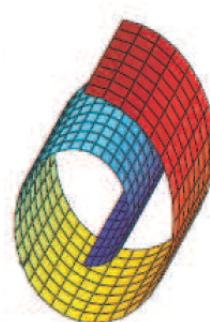
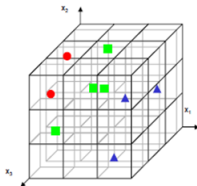
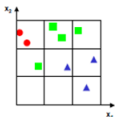


Для простоты можно представлять данные многообразия, как некоторые гладкие поверхности заключенные в Евклидово пространство

# Manifold assumption

## В чем смысл?

Хотим добиться “гладкости” решения в соответствии с внутренней структурой данных



- ▶ Расстояния в многомерных пространствах неинформативны (проклятие размерности)
- ▶ Работаем с расстояниями в пространстве меньшей размерности, отражающим внутреннюю структуру данных



# Обобщенная теория регуляризации

## Дополнительное ограничение на решение

Если две точки  $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$  находятся близко на многообразии, описывающим скрытую геометрию данных, то значения модели на этих точках также должны быть близки  $h(\mathbf{x}_i) \approx h(\mathbf{x}_j)$

## Функционал эмпирического риска с manifold regularization

$$err(h) = \frac{1}{2} \sum_{i=1}^N (y_i - h(\mathbf{x}_i))^2 + \frac{1}{2} \alpha_A \|h\|_K^2 + \frac{1}{2} \alpha_I \|h\|_I^2$$

$\|h\|_K^2$  - внешний регуляризатор, контролирующий сложность модели в исходном пространстве

$\|h\|_I^2$  - внутренний регуляризатор, отражает согласованность модели со скрытой внутренней геометрией данных

# SVM

Линейный классификатор на два класса  $Y = \{-1, 1\}$

$$h(x) = \text{sign}(\mathbf{w}^T \mathbf{x} - w_0)$$

Отступ объекта  $\mathbf{x}_i$

$$m_i(\mathbf{w}) = y_i(\mathbf{w}^T \mathbf{x}_i - w_0)$$

Функционал качества

$$J(\mathbf{w}) = \sum_{i=1}^N (1 - m_i)_+ + \frac{1}{2C} \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}, w_0}$$

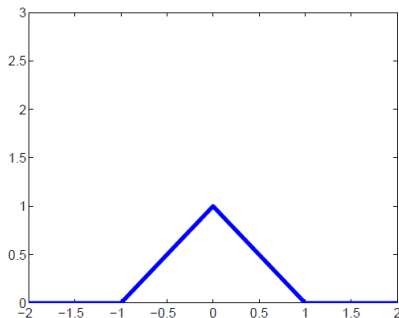
$$f_+ = \max(0, f)$$

Идея:

Функция  $L(m) = (1 - m)_+$  штрафует за попадание объекта внутрь разделяющей полосы

# TSVM

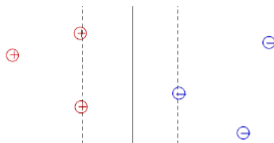
- ▶ **Наблюдение:** Величина отступа на неразмеченных данных есть  $(1 - |m|)_+$



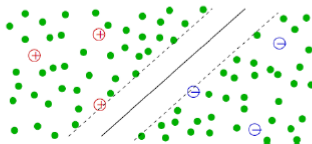
$$J(\mathbf{w}) = \sum_{i=1}^N (1 - m_i)_+ + \frac{1}{2C} \|\mathbf{w}\|^2 + \\ + \lambda \sum_{i=N+1}^{N+U} (1 - |m_i|)_+ \rightarrow \min_{\mathbf{w}, w_0}$$

# TSVM

SVMs



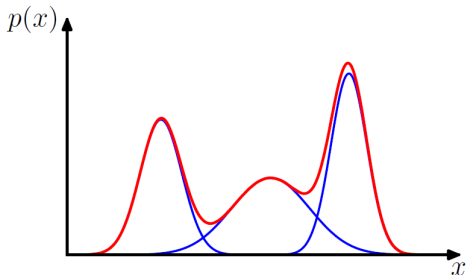
Semi-supervised SVMs (S3VMs) = Transductive SVMs (TSVMs)



- ▶ Решение неустойчиво, если нет зазора между классами
- ▶ Требуется настройка дополнительного параметра  $\lambda$

# Напоминание

## Смесь нормальных распределений

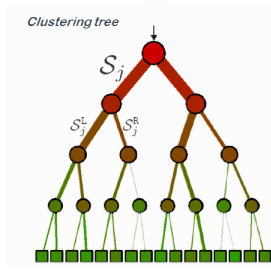
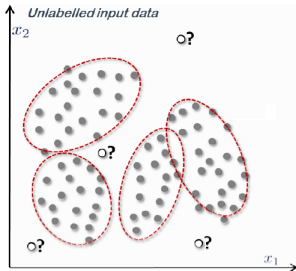


$$p(\mathbf{x}) = \sum_k \pi_k \mathcal{N}(\mathbf{x} | \mu_k, \Sigma_k)$$

# Оценка плотности при помощи деревьев решений

## Идея:

Вместо того, чтобы подбирать линейную модель давайте подбирать нелинейную в виде дерева решений



# Оценка плотности при помощи деревьев решений

Пусть  $\mathbf{X}$  - данные попадающие в текущий лист дерева

Критерий выбора разделяющего предиката

$$\theta^* = \arg \max_{\theta} \mathcal{I}$$

Information gain

$$\mathcal{I} = H(\mathbf{X}) - \frac{|\mathbf{X}^L|}{|\mathbf{X}|} H(\mathbf{X}^L) - \frac{|\mathbf{X}^R|}{|\mathbf{X}|} H(\mathbf{X}^R)$$

Энтропия  $d$ -мерного нормального распределения

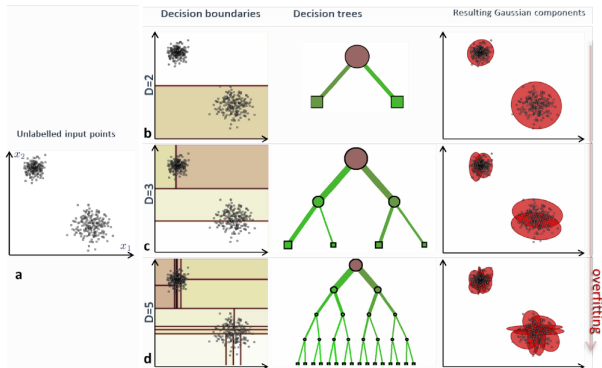
$$H(\mathbf{X}) = \frac{1}{2} \log((2\pi e)^d |\boldsymbol{\Sigma}(\mathbf{X})|)$$

где  $|\boldsymbol{\Sigma}(\mathbf{X})|$  - определитель ковариационной матрицы

$$\mathcal{I} = \log |\boldsymbol{\Sigma}(\mathbf{X})| - \frac{|\mathbf{X}^L|}{|\mathbf{X}|} \log |\boldsymbol{\Sigma}(\mathbf{X}^L)| - \frac{|\mathbf{X}^R|}{|\mathbf{X}|} \log |\boldsymbol{\Sigma}(\mathbf{X}^R)|$$

# Оценка плотности при помощи деревьев решений

## Зависимость от глубины дерева





# Semi supervised trees

Критерий выбора разделяющего предиката

$$\theta^* = \arg \max_{\theta} \mathcal{I}$$

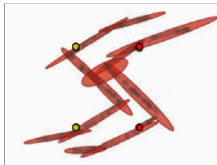
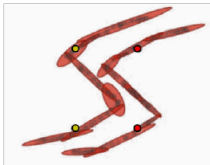
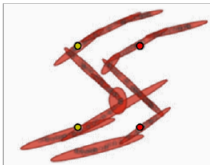
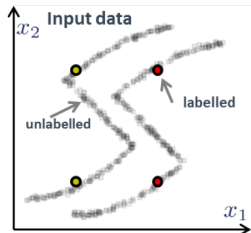
Information gain

$$\mathcal{I} = \mathcal{I}^S + \alpha \mathcal{I}^U$$

$$\mathcal{I}^S = H(\mathbf{X}_D) - \frac{|\mathbf{X}_D^L|}{|\mathbf{X}_D|} H(\mathbf{X}_D^L) - \frac{|\mathbf{X}_D^R|}{|\mathbf{X}_D|} H(\mathbf{X}_D^R)$$

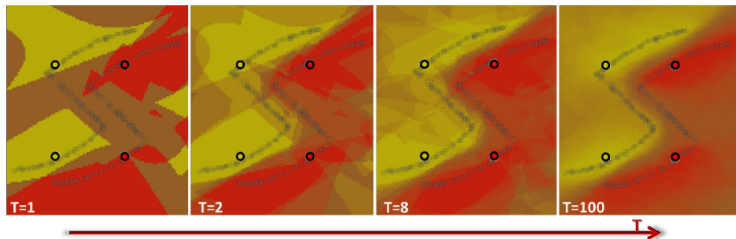
$$\mathcal{I}^U = \log |\boldsymbol{\Sigma}(\mathbf{X})| - \frac{|\mathbf{X}_{D \cup U}^L|}{|\mathbf{X}_{D \cup U}|} \log |\boldsymbol{\Sigma}(\mathbf{X}_{D \cup U}^L)| - \frac{|\mathbf{X}_{D \cup U}^R|}{|\mathbf{X}_{D \cup U}|} \log |\boldsymbol{\Sigma}(\mathbf{X}_{D \cup U}^R)|$$

# Semi supervised trees



# Semi supervised Random Forest

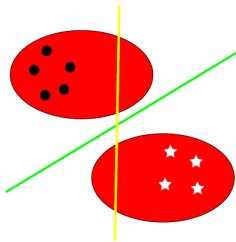
$$p(y|\mathbf{x}) = \frac{1}{T} \sum_{t=1}^T p_t(y|\mathbf{x})$$



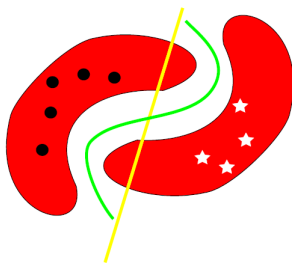
# Semi supervised boosting

$$h(\mathbf{x}) : \mathcal{X} \rightarrow Y = \{-1, +1\}$$

$$H(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$$

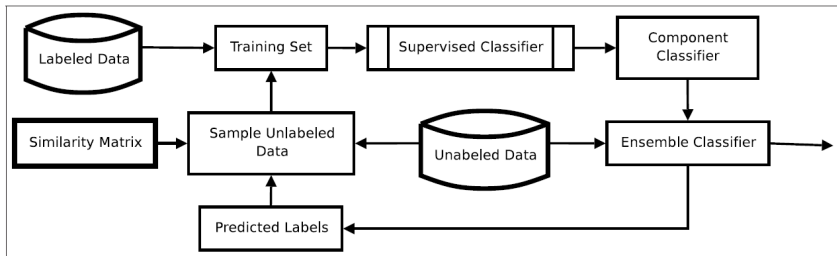


(a) Cluster assumption



(b) Manifold assumption

# SemiBoost



# SemiBoost

## Схема:

1. Start with empty ensemble
2. At each iteration
  - ▶ **Compute the pseudolabel (and its confidence)**  
for each unlabeled example, using existing ensemble, and pairwise similarity
  - ▶ **Sample most confident pseudolabeled examples;**  
combine them with labeled samples and train new model  $h$  with supervised learning algorithm
  - ▶ **Add new model to ensemble**  
with an appropriate weight

# SemiBoost

- ▶ Неразмеченные точки, похожие друг на друга должны иметь одну метку класса
- ▶ Точки похожие на размеченные должны иметь такую же метку класса

$$L(h(\mathbf{x})) = L_{\mathcal{L}}(y, H(\mathbf{x})) + \lambda L_{\mathcal{U}}(H(\mathbf{x}))$$

## Supervised

$$L_{\mathcal{L}}(\mathbf{y}, S) = \sum_{i=1}^{N_{\mathcal{L}}} \sum_{j=1}^{N_{\mathcal{U}}} S_{ij} \exp(-2y_i^l y_j^u)$$

## Unsupervised

$$L_{\mathcal{U}}(\mathbf{y}_u, S) = \sum_{i=1}^{N_{\mathcal{U}}} \sum_{j=1}^{N_{\mathcal{U}}} S_{ij} \cosh(y_i^u - y_j^u)$$

# SemiBoost

- ▶ Вычислить матрицу похожести между каждой парой примеров
- ▶ Инициализировать  $H(\mathbf{x}) = 0$
- ▶ Для  $t = 1, \dots, T$ 
  - ▶ Вычислить  $p_i$  и  $q_i$  для каждого примера

$$p_i = \sum_{j=1}^{N_L} S_{ij} e^{-2H_i} I(y_j = 1) + \frac{\lambda}{2} \sum_{j=1}^{N_U} S_{ij} e^{H_j - H_i}$$

$$q_i = \sum_{j=1}^{N_L} S_{ij} e^{2H_i} I(y_j = -1) + \frac{\lambda}{2} \sum_{j=1}^{N_U} S_{ij} e^{H_i - H_j}$$

- ▶ Выбрать примеры  $\mathbf{x}_i$  по весу  $|p_i - q_i|$
- ▶ Обучить классификатор  $h_t$  на семплированных данных
- ▶ Вычислить

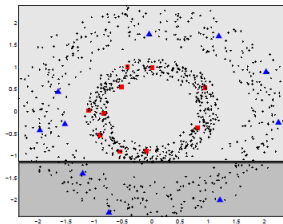
$$\alpha_t = \frac{1}{4} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right), \quad \epsilon_t = \frac{\sum_i^{N_U} p_i I(h_i = -1) + \sum_i^{N_U} q_i I(h_i = 1)}{\sum_i (p_i + q_i)}$$

- ▶ Обновить финальную модель

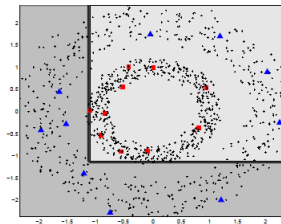
$$H(\mathbf{x}) \leftarrow H(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$$



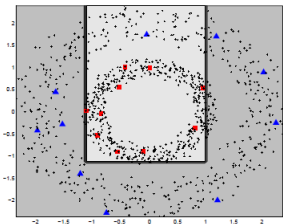
# SemiBoost with Decision Stumps



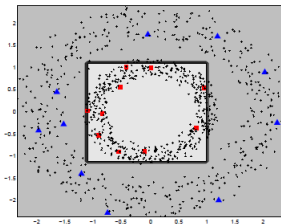
(a) Iter. 1, (65.0%)



(b) Iter. 2, (76.5%)



(c) Iter. 3, (86.4%)



(d) Iter. 12, (94.6%)

# Gradient Boosting with Priors and Manifold regularization

## Общий вид функции потерь

$$L(h) = \sum_{i=1}^{N_L} L(y_i, h(\mathbf{x}_i)) + \\ + \lambda J_p(p, q) + (1 - \lambda) \sum_{i=1}^{N_L} \sum_{j=1}^{N_U} s(\mathbf{x}_i, \mathbf{x}_j) J_m(\mathbf{x}_i, \mathbf{x}_j, h)$$

$p(y|\mathbf{x})$  - априорное распределение классов по кластерам

$q(y|\mathbf{x})$  - апостериорное (выдаваемое моделью) распределение классов по кластерам

# Итоги

- ▶ Частичное обучение занимает положение между обучением с учителем и обучением без учителя
- ▶ Методы обучения без учителя легче адаптируются к задаче частичного обучения, чем методы обучения с учителем
- ▶ Как правило, применение методов обучения с учителем к задаче частичного обучения приводит к лучшим результатам, чем применение метода обучения без учителя
- ▶ На практике часто используется вместе с активным обучением
- ▶ Если ваша выборка несостоятельна по фичам, то ничего не работает!!!

# Дз по частичному обучения:

## Задание:

Для варианта из вашего первого дз модифицировать алгоритм обучения с добавлением возможности работы в режиме частичного обучения. Использовать prior regularization и/или manifold regularization.

Варианты возможных регуляризаторов, представлены в приложении к данной презентации. Для того, чтобы узнать свой вариант необходимо по традиции воспользоваться функцией

```
def ComputeMyTaskNumber(your_name):  
    return 1 + hash(your_name) % number_of_cases
```

где `your_name` - это ваши фамилия и имя латиницей (например 'Pupkin Vasiliy')

`number_of_cases` - количество доступных для выбора вариантов

Дз по частичному обучения:

Имеется 5 дополнительных опций к заданию. Для первой опции нужно использовать функцию

```
def ComputeMyTaskNumber(your_name):  
    return 1 + hash(your_name) % number_of_cases
```

Для 2-его задания

```
def ComputeMyTaskNumber(your_name):  
    return 1 + hash(hash(your_name)) % number_of_cases
```

Для остальных по аналогии :)

# Manifolds

1. Principal Component Analysis
2. Linear Discriminant Analysis
3. Isomap
4. Locally Linear Embedding
5. Modified Locally Linear Embedding
6. Hessian Eigenmapping
7. Spectral Embedding
8. Local Tangent Space Alignment
9. Multi-dimensional Scaling
10. t-distributed Stochastic Neighbor Embedding (t-SNE)

## ВАЖНОЕ ЗАМЕЧАНИЕ

ВСЕ РЕАЛИЗОВАНО В `scikitlearne`. ПРЕОБРАЗОВАНИЯ САМИМ РЕАЛИЗОВЫВАТЬ НЕ НУЖНО!

# Similarity

## 1. Gauss

$$S(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

## 2. Laplas

$$S(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|)$$

## 3. Inverse quadratic

$$S(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2}$$

## 4. Inverse multiquadric

$$S(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\sqrt{1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2}}$$

## 5. Epanechnikov

$$S(\mathbf{x}_i, \mathbf{x}_j) = \frac{3}{4}(1 - \|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

# Penalty

## 1. Exponential

$$J_m(\mathbf{x}_i, \mathbf{x}_j, h) = \cosh(h(\mathbf{x}_i) - h(\mathbf{x}_j))$$

## 2. Polynomial

$$J_m(\mathbf{x}_i, \mathbf{x}_j, h) = (h(\mathbf{x}_i) - h(\mathbf{x}_j))^k, \dots k = 2, 4, 6, \dots$$

## 3. Sublinear

$$J_m(\mathbf{x}_i, \mathbf{x}_j, h) = \ln(1 + |h(\mathbf{x}_i) - h(\mathbf{x}_j)|)$$



# Clustering

1. K-Means
2. DBSCAN
3. Affinity Propagation
4. Mean Shift
5. Spectral clustering
6. Hierarchical clustering
7. Birch

## ВАЖНОЕ ЗАМЕЧАНИЕ

Опять же ничего самим имплементировать не нужно. Берем из `sclearn`.

# Probability Distance Measures

## 1. Chi-square

$$J_p(p, q) = \sum_{i=1}^{N_U} \frac{(p(y|x_i) - q(y|x_i))^2}{q(y|x_i)}$$

## 2. Hellinger distance

$$J_p(p, q) = \frac{1}{2} \sum_{i=1}^{N_U} (\sqrt{p(y|x_i)} - \sqrt{q(y|x_i)})^2$$

## 3. Kullback-Leibler divergence

$$J_p(p, q) = \sum_{i=1}^{N_U} p(y|x_i) \ln \frac{p(y|x_i)}{q(y|x_i)}$$

## 4. Jensen-Shannon divergence

$$J_p(p, q) = \frac{1}{2} \sum_{i=1}^{N_U} p(y|x_i) \ln \frac{p(y|x_i)}{q(y|x_i)} + \frac{1}{2} \sum_{i=1}^{N_U} q(y|x_i) \ln \frac{q(y|x_i)}{p(y|x_i)}$$

## 5. Euclidean distance

$$J_p(p, q) = \sum_{i=1}^{N_U} (p(y|x_i) - q(y|x_i))^2$$

## 6. Bhattacharyya distance

$$J_p(p, q) = -\ln\left(\sum_{i=1}^{N_U} \sqrt{p(y|x_i)q(y|x_i)}\right)$$

# Вопросы

