



Лекция 16: Частичное обучение

Постановка задачи частичного обучения (SSL)

■ Дано:

- множество объектов X , множество классов Y ;
- $X^k = \{x_1, \dots, x_k\}$ – размеченные объекты (labeled data);
 $\{y_1, \dots, y_k\}$
- $U = \{x_{k+1}, \dots, x_l\}$ – неразмеченные объекты (unlabeled data).

■ Два варианта постановки задачи:

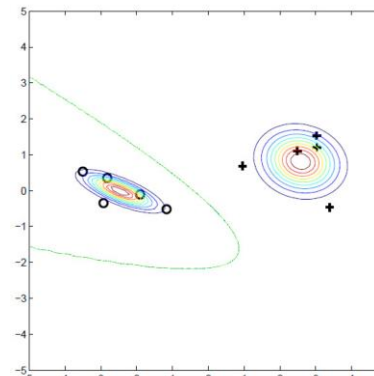
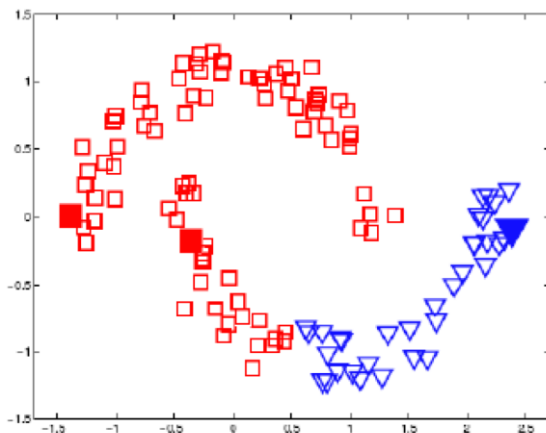
- Частичное обучение (semi-supervised learning): построить алгоритм классификации $a: X \rightarrow Y$.
- Трансдуктивное обучение (transductive learning): зная **все** $\{x_{k+1}, \dots, x_l\}$, получить метки $\{a_{k+1}, \dots, a_l\}$.

■ Типичные приложения:

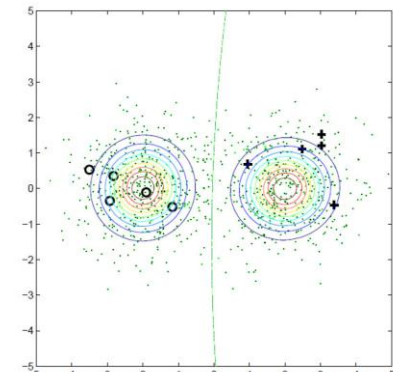
- Классификация и каталогизация текстов, изображений, и т.п.
- Применяется везде, где разметки мало или она дорогая

Отличия SSL

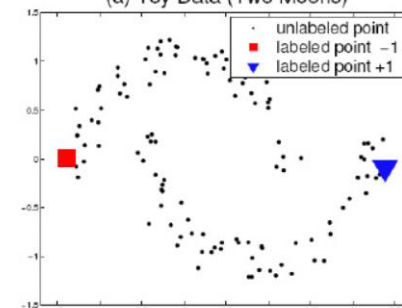
- Разные результаты прогноза, если восстанавливать плотности классов по всем или только по размеченным
- Классификация не учитывает кластерную структуру неразмеченных данных
- Кластеризация не учитывает приоритет разметки



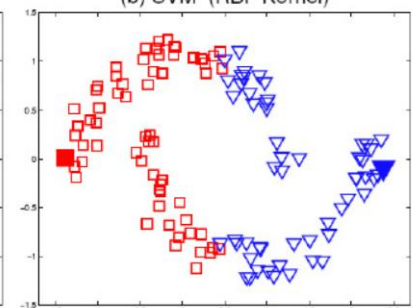
(a) Toy Data (Two Moons)



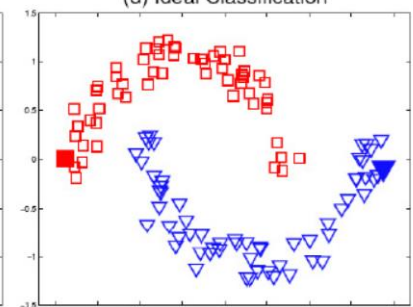
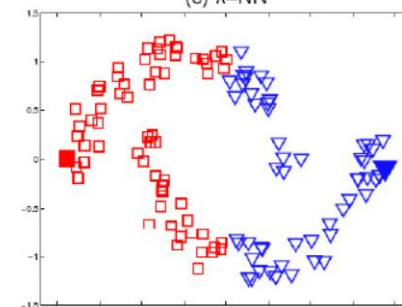
(b) SVM (RBF Kernel)



(c) k -NN



(d) Ideal Classification



Основные подходы к SSL

- Кластеризация с учетом ограничений разметки:
 - не объединять в кластер объекты из разных классов
- Последовательная доразметка:
 - Последовательное дообучение на своих наиболее уверенных прогнозах (self-training)
 - Последовательное дообучение на чужих наиболее уверенных прогнозах, включая ансамбли (co-training)
- Вероятностные модели:
 - Распространение меток– пересчет распределения меток классов на основе графа связей «сходства» наблюдений (label propagation)
 - Параметрическая оценка распределений классов с учетом неразмеченных примеров
- Оптимизационный подход:
 - Включение в целевую функцию потерь прогнозирования и потерь некомпактной кластеризации: $Q_{SSL} = Q_{class} + \gamma Q_{cluster} \rightarrow \min$

Метод К-средних для частичного обучения

- Модификация алгоритм Ллойда
- При наличии размеченных объектов $\{x_1, \dots, x_k\}$
 - Вход: $X^l, K = |Y|$;
 - Выход: центры кластеров $\mu_a, a \in Y$;
 - μ_a := начальное приближение центров, для всех $a \in Y$;
 - Повторять

Отнести каждый $x_i \in U$ к ближайшему центру:

$$a_i := \arg \min_{a \in Y} \|x_i - \mu_a\|, i = k + 1, \dots, l$$

Вычислить новые положения центров:

$$\mu_a := \frac{\sum_{i=1}^l [a_i = a] x_i}{\sum_{i=1}^l [a_i = a]}, a \in Y;$$

- Пока a_i не перестанут изменяться.

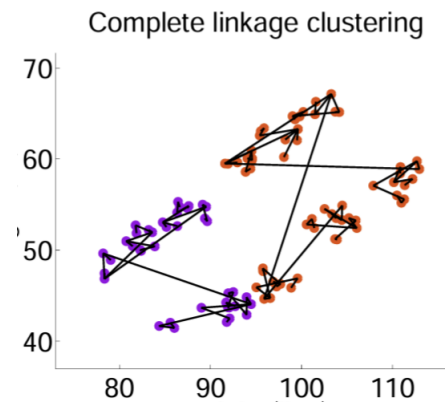
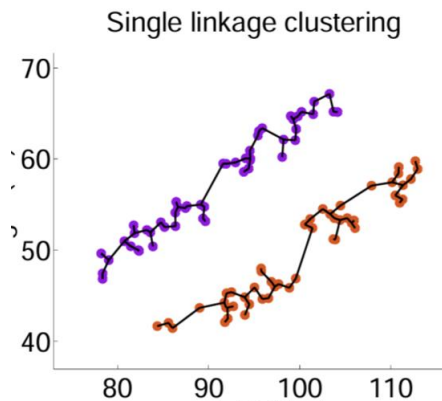
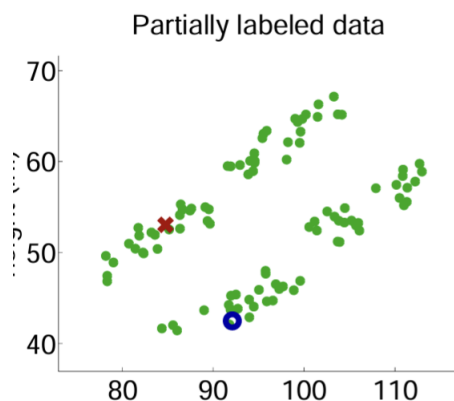
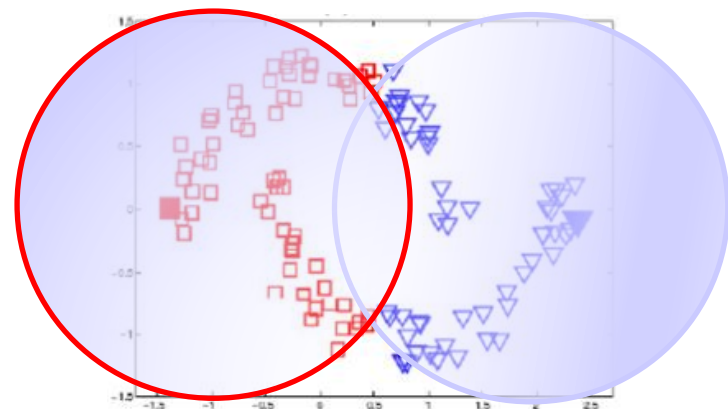
Алгоритм Ланса-Уильямса для частичного обучения

- Алгоритм иерархической кластеризации (Ланс, Уильямс, 1967):
- Итеративный пересчет расстояний R_{UV} между кластерами U, V .
- $C_1 := \{\{x_1\}, \dots, \{x_l\}\}$ – все кластеры 1-элементные;
- $R_{\{x_i\}\{x_j\}} := \rho(x_i, x_j)$ – расстояния между ними;
- Для всех $t = 2, \dots, l$ (t – номер итерации):
 - Найти в C_{t-1} пару кластеров (U, V) с минимальным R_{UV} , при условии, что **в $U \cup V$ нет объектов с разными метками**;
 - Слить их в один кластер:
 - $W := U \cup V$;
 - $C_t := C_{t-1} \cup \{W\} \setminus \{U, V\}$;
 - Для всех $S \in C_t$ Вычислить R_{wS} по формуле Ланса-Уильямса:
$$R_{wS} := \alpha_U R_{US} + \alpha_V R_{VS} + \beta R_{UV} + \gamma |R_{US} - R_{VS}|$$

Недостатки SSL на основе кластеризации с учетом разметки

■ Все недостатки методов кластеризации проявляются сильнее:

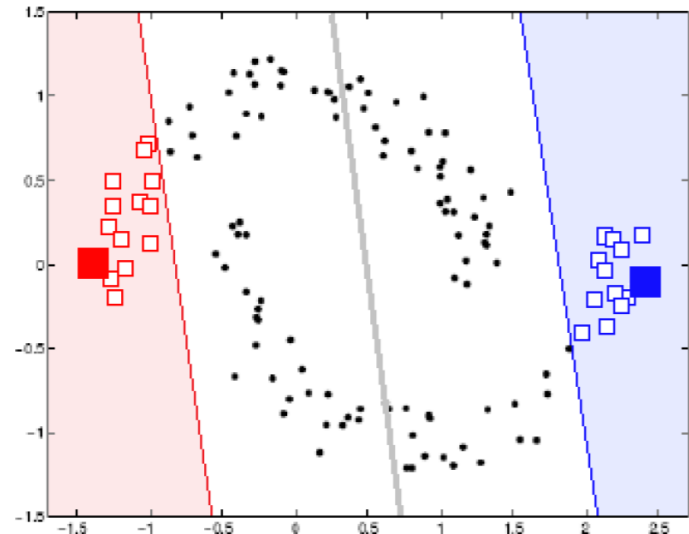
- Для кластеризации на основе прототипов ожидаются сферические (или эллиптические) формы классов
- Нужно угадать с числом кластеров или брать «с запасом», но сколько?
- Для иерархической кластеризации важно межкластерное расстояние



Метод частичного обучения self-training (1965–1970)

- Пусть $\mu: X^k \rightarrow a$ – метод обучения классификации;
- Классификаторы имеют вид $a(x) = \operatorname{argmax}_{y \in Y} \Gamma_y(x)$;
- Псевдоотступ – степень уверенности классификации $a_i = a(x_i)$:
$$M_i(a) = \Gamma_{a_i}(x_i) - \max_{y \in Y \setminus a_i} \Gamma_y(x_i)$$
- Алгоритм self-training – обертка (wrapper) над методом μ :

- $Z := X^k$;
- Пока $|Z| < l$
 - $a := \mu(Z)$
 - $\Delta := \{x_i \in U \setminus Z \mid M_i(a) \geq M_0\}$
 - $y_i := a(x_i)$ для всех $x_i \in \Delta$
 - $Z := Z \cup \Delta$
- M_0 можно определять, например, из условия $|\Delta| = 0.05|U|$



Метод частичного обучения co-training (Blum, Mitchell, 1990)

- Пусть $\mu_1: X^{k_1} \rightarrow a_1, \mu_2: X^{k_2} \rightarrow a_2$ — два существенно различных метода обучения, использующих
 - Либо разные наборы признаков;
 - Либо разные парадигмы обучения (inductive bias);
 - Либо разные источники данных $X_1^{k_1}, X_2^{k_2}$.
- $Z_1 := X_1^{k_1}; Z_2 := X_2^{k_2};$
- Пока $|Z_1 \cup Z_2| < l$
 - $a_1 := \mu_1(Z_1); \Delta_1 := \{x_i \in U \setminus Z_1 \setminus Z_2 | M_i(a_1) \geq \mathbf{M}_{01}\};$
 - $y_i := a_1(x_i)$ для всех $x_i \in \Delta_1$;
 - $Z_2 := Z_2 \cup \Delta_1$;
 - $a_2 := \mu_2(Z_2); \Delta_2 := \{x_i \in U \setminus Z_1 \setminus Z_2 | M_i(a_2) \geq \mathbf{M}_{02}\};$
 - $y_i := a_2(x_i)$ для всех $x_i \in \Delta_2$;
 - $Z_1 := Z_1 \cup \Delta_2$;

Метод частичного обучения со- learning (deSa, 1993)

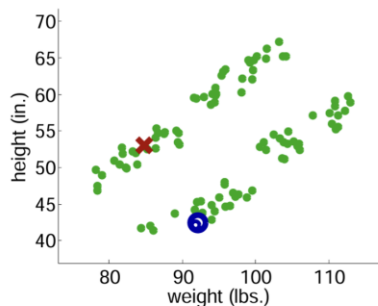
- Пусть $\mu_t: X^k \rightarrow a_t$ – разные методы обучения, $t = 1, \dots, T$.
- Алгоритм co-learning – это self-training для композиции – простого голосования базовых алгоритмов a_1, \dots, a_T :

$$a(x) = \arg \max_{y \in Y} \Gamma_y(x), \Gamma_y(x_i) = \sum_{t=1}^T [a_t(x_i) = y]$$

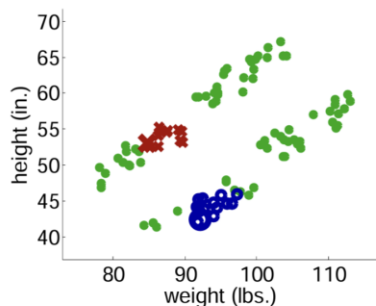
- Тогда $M_i(a)$ – степень уверенности классификации $a(x_i)$.
- $Z := X^k$;
- Пока $|Z| < l$
 - $a := \mu(Z)$;
 - $\Delta := \{x_i \in U \setminus Z | M_i(a) \geq M_0\}$;
 - $y_i := a(x_i)$ для всех $x_i \in \Delta$;
 - $Z := Z \cup \Delta$

Self- и co- training не всегда работают

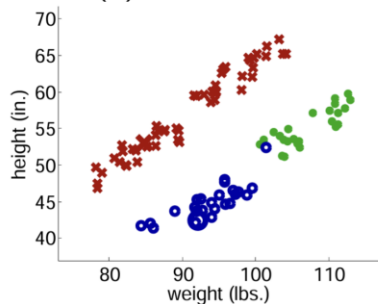
Self-training 1NN – все Ok



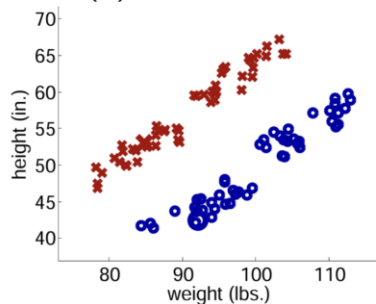
(a) Iteration 1



(b) Iteration 25

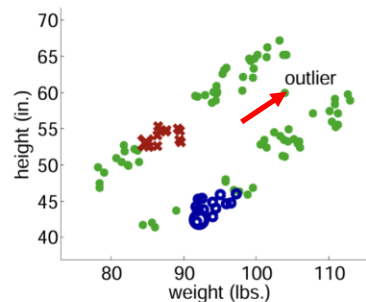


(a) Iteration 1

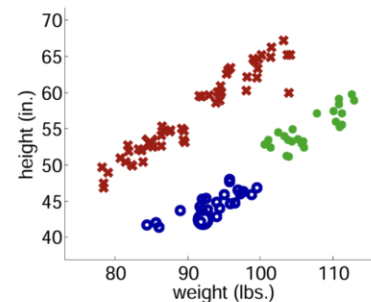


(b) Iteration 25

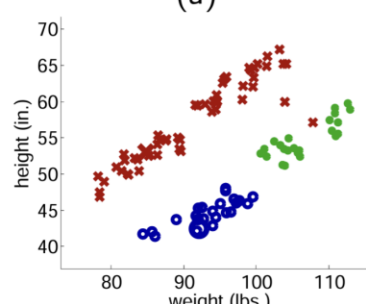
Self-training 1NN + 1 выброс -
все сломалось



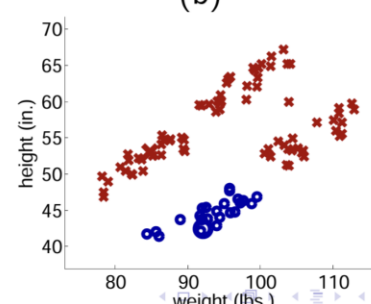
(a)



(b)



(a)



(b)

Общий оптимизационный подход к задачам SSL

- Дано:

- $X^k = \{x_1, \dots, x_k\}$ – размеченные объекты (labeled data);
 $\{y_1, \dots, y_k\}$
- $U = \{x_{k+1}, \dots, x_l\}$ – неразмеченные объекты (unlabeled data).

- Найти: модель классификации $a(x, w)$

- Критерий одновременной классификации и кластеризации:

$$\underbrace{\sum_{i=1}^k \mathcal{L}(a(x_i, w), y_i)}_{\text{классификация}} + \lambda \underbrace{\sum_{i=1}^l \mathcal{L}_u(a(x_i, w))}_{\text{кластеризация}} \rightarrow \min_w$$

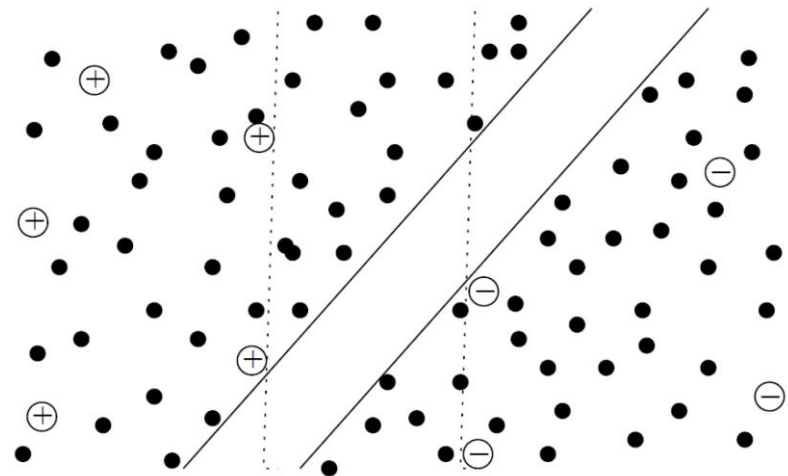
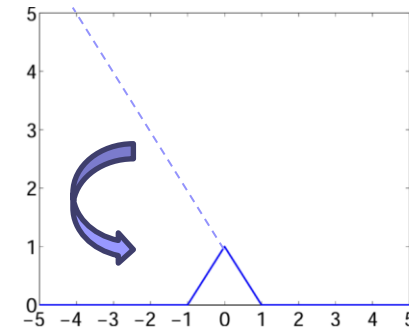
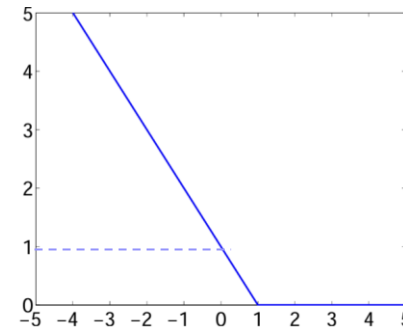
- Где $\mathcal{L}(a, y)$ – функция потерь классификации,
- $\mathcal{L}_U(a)$ – функция потерь для неразмеченных данных.

Функция потерь для трансдуктивного SVM

- Функция потерь $L = (1 - |M|)_+$:
 - штрафует за попадание объекта **внутри** разделяющей полосы
- Обучение весов w, w_0 по частично размеченной выборке:

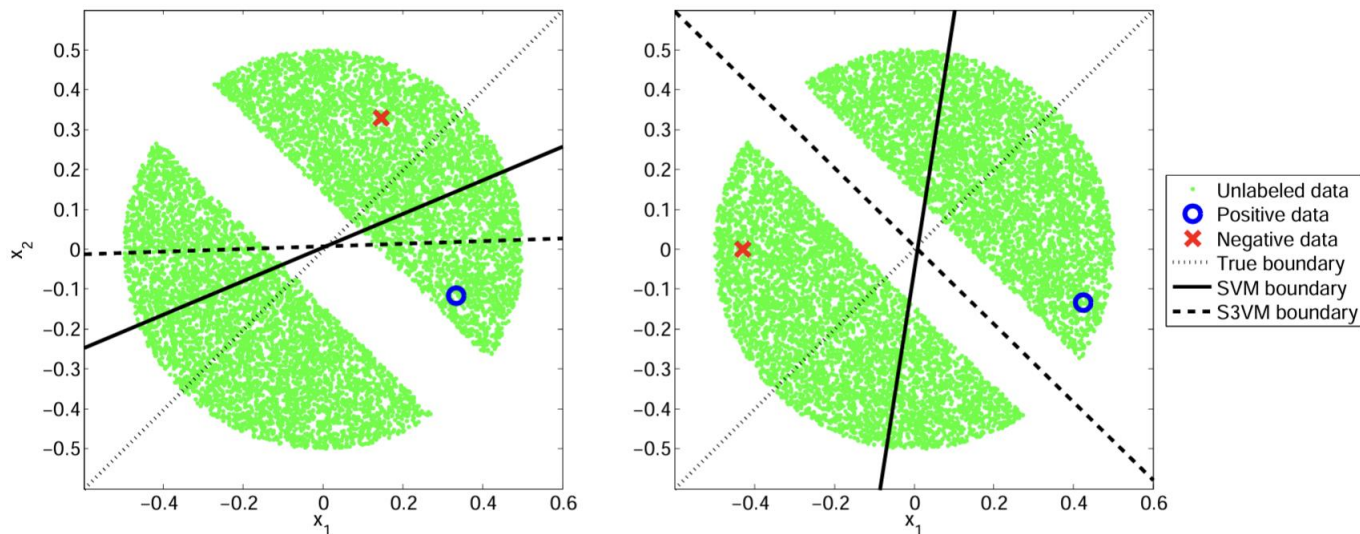
$$Q = \sum_{i=1}^k (1 - M_i(w, w_0))_+ + \frac{1}{2C} \|w\|^2 + \gamma \sum_{i=k+1}^l (1 - |M_i(w, w_0)|)_+ \rightarrow \min_{w, w_0}$$

- Достоинства TSVM:
 - как и в обычном SVM, можно использовать ядра;
 - имеются эффективные реализации для больших данных



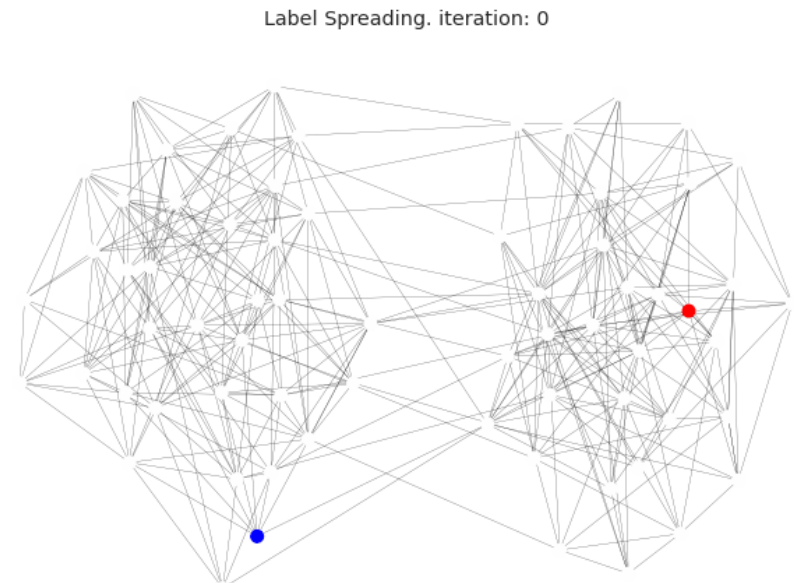
Проблемы TSVM

- задача невыпуклая - методы оптимизации сложнее и неединственное решение;
- требуется настройка двух параметров регуляризации C, γ
- решение неустойчиво или неверно, если нет области разреженности между классами;



Распространение меток

- Общая идея подхода (есть много разных версий):
 - Графовая модель представления всех данных (и размеченных, и неразмеченных), где узлы – наблюдения, ребра – оценка сходства без учета разметки
 - Разметка на основе случайного блуждания (random walk), где вероятность перехода задаётся, пропорционально «весу» ребра - сходству
 - Метка размеченного наблюдения не меняется, а для неразмеченного выбирается голосованием или пропорционально сходству всех уже размеченных ближайших соседей



Распространение меток

■ Дано:

- $X^k = \{x_1, \dots, x_k\}, \{y_1, \dots, y_k\}$ – размеченные объекты (labeled data);
- $U = \{x_{k+1}, \dots, x_l\}$ – неразмеченные объекты (unlabeled data), обычно имеет смысл применять распространение меток при $k \ll l$;
- Y – различные метки классов, $A^{(0)}$ – матрица начальной разметки $l \times |Y|$, где $A_{ij}^{(0)} = 1$ ттт, когда $y_i = j$ для $i \leq k$, иначе $A_{ij}^{(0)} = 0$
- мера близости двух любых примеров (не зависит от разметки), обычно на основе RBF или другого ядра: $w_{ij} = K(\rho(x, x_i)^2/h)$, чем разреженной матрица $l \times l$ ядра тем лучше;
- $\{w_{ij}\}$ задает матрицу $l \times l$ переходов $T_{ij} = P(i \rightarrow j) = w_{ij} / \sum_{s=1}^l w_{sj}$

■ Алгоритм в цикле находит матрицу разметки $A^{(t)}: X \rightarrow Y$:

- Пересчет меток: $A^{(t)} = \alpha T A^{(t-1)} + (1 - \alpha) A^{(0)}, 0 < \alpha < 1$
- Нормализация $Y^{(t)}$: обычно фиксируют размеченные метки $A_{x \notin U}^{(t)} = A_{x \notin U}^{(0)}$, остальные нормируют ($\sum_{s=1}^{|Y|} A_{is}^{(t)} = 1$) или выбирает по максимальной уверенности в прогнозе ($y_{i>k} = \operatorname{argmax}_j [A_{ij}^{(t)}]$)

Пример

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.semi_supervised import LabelSpreading
from sklearn.semi_supervised import SelfTrainingClassifier

iris = datasets.load_iris()

X = iris.data[:, :2]
y = iris.target

# step size in the mesh
h = .02

rng = np.random.RandomState(0)
y_rand = rng.rand(y.shape[0])
y_30 = np.copy(y)
y_30[y_rand < 0.3] = -1 # set random samples to be unlabeled
y_50 = np.copy(y)
y_50[y_rand < 0.5] = -1
# we create an instance of SVM and fit out data. We do not scale our
# data since we want to plot the support vectors
ls30 = (LabelSpreading().fit(X, y_30), y_30, 'Label Spreading 30% data')
ls50 = (LabelSpreading().fit(X, y_50), y_50, 'Label Spreading 50% data')
ls100 = (LabelSpreading().fit(X, y), y, 'Label Spreading 100% data')

# the base classifier for self-training is identical to the SVC
base_classifier = SVC(kernel='rbf', gamma=.5, probability=True)
st30 = (SelfTrainingClassifier(base_classifier).fit(X, y_30),
        y_30, 'Self-training 30% data')
st50 = (SelfTrainingClassifier(base_classifier).fit(X, y_50),
        y_50, 'Self-training 50% data')

rbf_svc = (SVC(kernel='rbf', gamma=.5).fit(X, y), y, 'SVC with rbf kernel')

# create a mesh to plot in
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
```

```
color_map = {-1: (1, 1, 1), 0: (1, 0, 0), 1: (0.2, 0.2, 0.6), 2: (0.7, 0.7, 0.7)}

classifiers = (ls30, st30, ls50, st50, ls100, rbf_svc)
for i, (clf, y_train, title) in enumerate(classifiers):

    plt.subplot(3, 2, i + 1)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

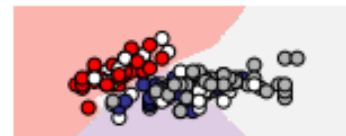
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap="Pastel1")
    plt.axis('off')

    colors = [color_map[y] for y in y_train]
    plt.scatter(X[:, 0], X[:, 1], c=colors, edgecolors='black')

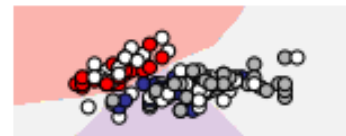
    plt.title(title)

plt.suptitle("Unlabeled points are colored white", y=0.1)
plt.show()
```

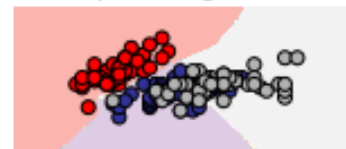
Label Spreading 30% data



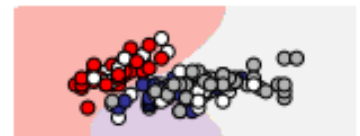
Label Spreading 50% data



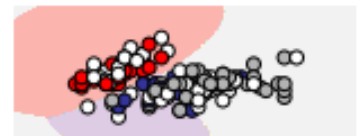
Label Spreading 100% data



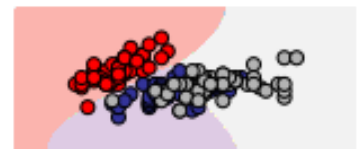
Self-training 30% data



Self-training 50% data



SVC with rbf kernel



Unlabeled points are colored white

Вероятностный (байесовский) SSL

- Для размеченной выборки (если число компонент смеси = числу классов):

- Модель: $p(x, y|\theta) = P(y|\theta)p(x|y, \theta)$
- Задача максимизации правдоподобия по θ :

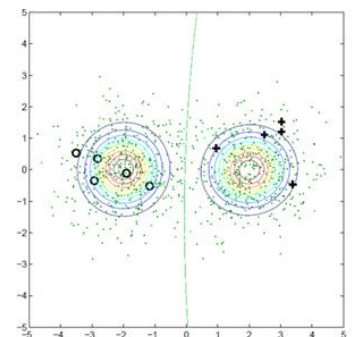
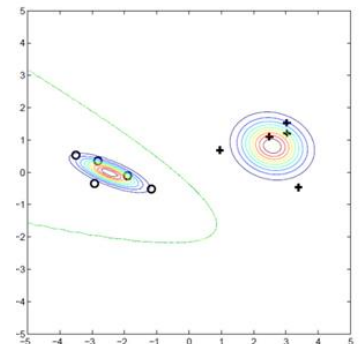
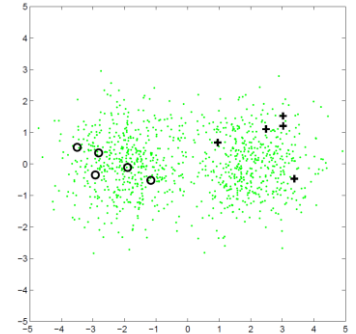
$$\log(p(X_l, Y_l|\theta)) = \sum_l \log(P(y_l|\theta)p(x_l|y_l, \theta)) \rightarrow \max_{\theta}$$

- Классификация $P(y|x, \theta) = \frac{p(x, y|\theta)}{\sum_{y'} p(x, y'|\theta)}$

- Для частично размеченной выборки:

- $X^k = \{x_1, \dots, x_k\}, \{y_1, \dots, y_k\}$ – размеченные объекты;
- $U = \{x_{k+1}, \dots, x_l\}$ – неразмеченные объекты;
- Задача максимизации правдоподобия по θ :

$$\begin{aligned} \log(p(X_k, Y_k, U|\theta)) = & \sum_{i=1}^k \log(P(y_i|\theta)p(x_i|y_i, \theta)) + \\ & + \sum_{i=k+1}^l \log\left(\sum_j P(j|\theta)p(x_i|j, \theta)\right) \rightarrow \max_{\theta} \end{aligned}$$



SSL GMM (Gaussian Mixture Model)

- Вход:

- $X^k = \{x_1, \dots, x_k\}, \{y_1, \dots, y_k\}$ – размеченные объекты (labeled data);
- $U = \{x_{k+1}, \dots, x_l\}$ – неразмеченные объекты (unlabeled data);

- Выход: $(w_j, \mu_j, \Sigma_j)_{j=1}^{|Y|}$ – параметры смеси гауссиан;

- Инициализировать $(\mu_j, \Sigma_j)_{j=1}^{|Y|}, w_j := \frac{1}{|Y|}$

- Повторять:

- **Е-шаг** (expectation): для $x_i \notin U$ $g_{ij} := 1$ если $y_i = j$, иначе $g_{ij} := 0$
для всех $x_i \in U, j = 1, \dots, |Y|$:

$$g_{ij} := p(y_i = j | x_i) \frac{w_j N(x_i; \mu_j, \Sigma_j)}{\sum_{s=1}^{|Y|} w_s N(x_i; \mu_s, \Sigma_s)}$$

- **М-шаг** (maximization): для всех $j = 1, \dots, |Y|$

$$w_j := \frac{1}{l} \sum_{i=1}^l g_{ij}, \mu_j := \frac{1}{lw_j} \sum_{i=1}^l g_{ij} x_i, \Sigma_j := \frac{1}{lw_j} \sum_{i=1}^l g_{ij} (x_i - \mu_j)(x_i - \mu_j)^T$$

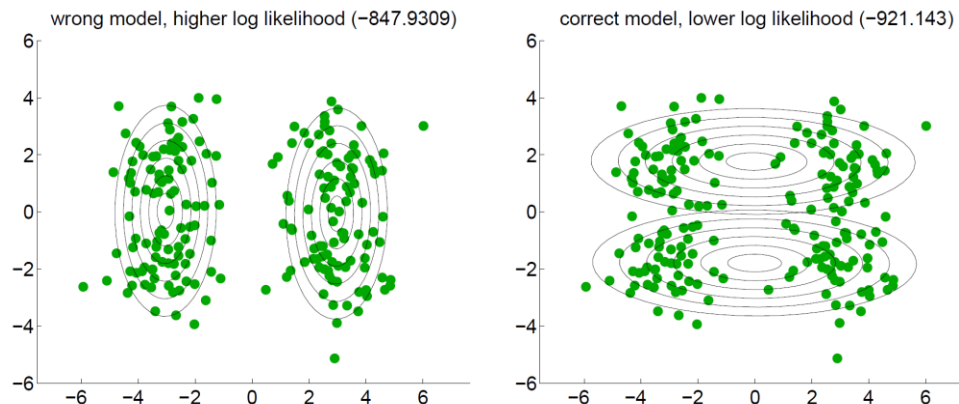
- Пока (w_j, μ_j, Σ_j) и / или g_{ij} не сошлись.

Особенности SSL EM

- SSL EM – специальный случай co-training
- Чтобы контролировать важность разметки можно использовать регуляризацию ($0 < \gamma < 1$):

$$\log(p(X_k, Y_k, U|\theta)) = \sum_{i=1}^k \log(p(y_i|\theta)p(x_i|y_i, \theta)) + \\ + \gamma \sum_{i=k+1}^l \log\left(\sum_j p(j|\theta)p(x_i|j, \theta)\right) \rightarrow \max_{\theta}$$

- Зачастую надо брать компонент смеси больше чем классов
- В целом не всегда работает:



Рассмотрим линейный классификатор с независимыми бинарными признаками

- Линейный классификатор по конечному множеству классов:

$$a(x) = \operatorname{argmax}_{y \in Y} \langle w_y, x \rangle, \text{ где } w_y, x \in \mathbb{R}$$

- Пусть $b_j(x)$ – бинарные признаки, $j = 1, \dots, m$
- Вероятность принадлежности наблюдения классу можно задать через softmax:

$$P(y|x_i, w) = \frac{\exp\langle w_y, x \rangle}{\sum_{c \in Y} \exp\langle w_c, x \rangle}$$

- Максимизации регуляризованного правдоподобия:

$$Q(w) = \sum_{i=1}^l \log P(y_i|x_i, w) - \frac{1}{2C} \sum_{c \in Y} \|w_c\|^2 \rightarrow \max_w$$

- Решение задачи возможно стохастическим градиентным спуском по вектору параметров w

Согласование модели на размеченных и неразмеченных данных

- Теперь учтем неразмеченные данные $U = \{x_{k+1}, \dots, x_l\}$.
- Оценим вероятности $P(y|b_j(x) = 1)$ двумя способами:

- Эмпирическая оценка по размеченным данным X^k :

$$\hat{p}_j(y) = \frac{\sum_{i=1}^k b_j(x_i)[y_i = y]}{\sum_{i=1}^k b_j(x_i)}$$

- Оценка по неразмеченным данным U и вероятностной модели:

$$p_j(y|w) = \frac{\sum_{i=k+1}^l b_j(x_i)P(y|x_i, w)}{\sum_{i=k+1}^l b_j(x_i)}$$

- Максимизируем правдоподобие вероятностной модели $p_j(y|w)$, приближающей эмпирическое распределение $\hat{p}_j(y)$ по j -му признаку:

$$L_j(w) = \sum_{y \in Y} \hat{p}_j(y) \log p_j(y|w) \rightarrow \max_w$$

Построение регуляризатора (XR, eXpectation Regularization)

- Регуляризация критерия $Q(w)$ суммой log-правдоподобий $L_j(w)$ с коэффициентом регуляризации γ :

$$Q(w) + \gamma \sum_{j=1}^m L_j(w) = \sum_{i=1}^k \log P(y_i | x_i, w) - \frac{1}{2C} \sum_{y \in Y} \|w_y\|^2 + \\ + \gamma \sum_{j=1}^m \sum_{y \in Y} \hat{p}_j(y) \log \left(\sum_{i=k+1}^l b_j(x_i) P(y | x_i, w) \right) \rightarrow \max_w$$

- Оптимизация методом стохастического градиента.
- Возможные варианты задания переменных b_j :
 - $b_j(x) \equiv 1$, тогда $P(y | b_j(x) = 1)$ – априорная вероятность класса y (label regularization);
 - $b_j(x) = [\text{терм } j \in \text{документ } x]$ - для классификации текстов.
- Особенности:
 - слабо чувствителен к выбору C и γ , устойчив к погрешностям $\hat{p}_j(y)$, не требует большого k , хорош для текстовых данных