МЕТОД ОПОРНЫХ ВЕКТОРОВ II

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург
10 марта 2017 г.

Random facts:

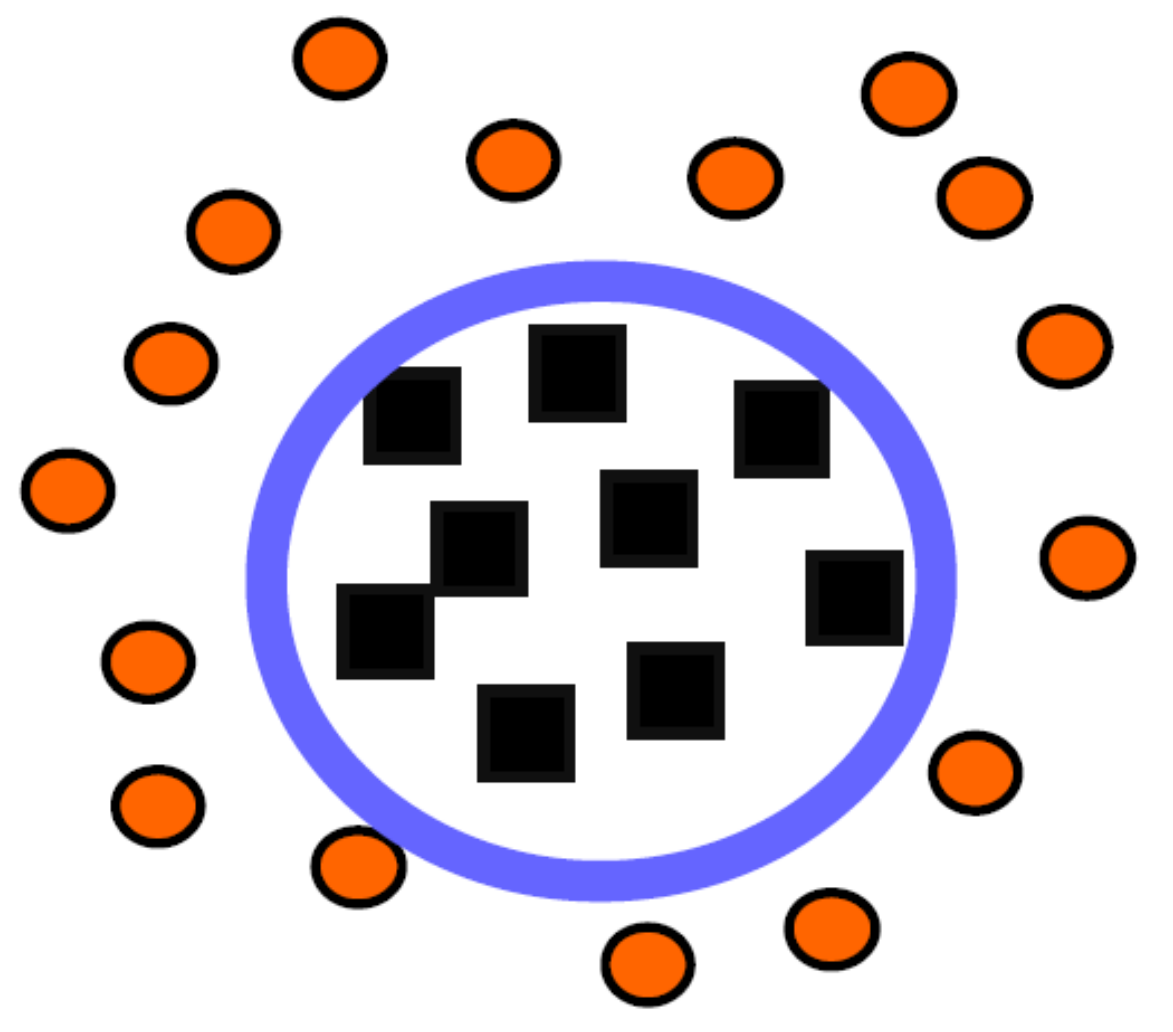
- 10 марта 1804 г. -- официальная церемония Louisiana purchase; к США за $15 млн перешли территории современных Арканзаса, Миссури, Айовы, Оклахомы, Канзаса, Небраски, Южной Дакоты, части Миннесоты, Северной Дакоты, Нью-Мексико, Монтаны, Вайоминга, Техаса, Колорадо... ну и, действительно, часть Луизианы
- в Южной Корее 10 марта -- День труда; как, впрочем, и большинство остальных

SVM и разделение нелинейными функциями

ВВЕДЕНИЕ

- Часто бывает нужно разделять данные не только линейными функциями.
- Что делать в таком случае?

ПРИМЕР



ВВЕДЕНИЕ

- Часто бывает нужно разделять данные не только линейными функциями.
- Классический метод: развернуть нелинейную классификацию в пространство большей размерности (feature space), а там запустить линейный классификатор.
- Для этого просто нужно для каждого монома нужной степени ввести новую переменную.

ПРИМЕР

- Чтобы в двумерном пространстве $[r, s]$ решить задачу классификации квадратичной функцией, надо перейти в пятимерное пространство:

$$[r, s] \longrightarrow [r, s, rs, r^2, s^2].$$

- Или формальнее; определим $\theta : \mathbb{R}^2 \to \mathbb{R}^5$: $\theta(r, s) = (r, s, rs, r^2, s^2)$. Вектор в $\mathbb{R}^5$ теперь соответствует квадратичной кривой общего положения в $\mathbb{R}^2$, а функция классификации выглядит как

$$f(\vec{x}) = \operatorname{sign}(\theta(\vec{w}) \cdot \theta(\vec{x}) - b).$$

- Если решить задачу линейного разделения в этом новом пространстве, тем самым решится задача квадратичного разделения в исходном.

ПРОБЛЕМЫ С КЛАССИЧЕСКИМ ПОДХОДОМ

- Во-первых, количество переменных растёт экспоненциально.
- Во-вторых, по большому счёту теряются преимущества того, что гиперплоскость именно оптимальная; например, оверфиттинг опять становится проблемой.
- Важное замечание: *концептуально* мы задачу уже решили. Остались *технические* сложности: как обращаться с гигантской размерностью. Но в них-то всё и дело.

ОСНОВНАЯ ИДЕЯ И СХЕМА РАБОТЫ SVM

- Тривиальная схема алгоритма классификации такова:
  - входной вектор $\vec{x}$ трасформируется во входной вектор в feature space (большой размерности);
  - в этом большом пространстве мы вычисляем опорные векторы, решаем задачу разделения;
  - потом по этой задаче классифицируем входной вектор.
- Это нереально, потому что пространство слишком большой размерности.

Основная идея и схема работы SVM

- Оказывается, кое-какие шаги здесь можно переставить. Вот так:
  - опорные векторы вычисляются в исходном пространстве малой размерности;
  - там же они перемножаются (сейчас увидим, что это значит);
  - и только потом мы делаем нелинейную трансформацию того, что получится;
  - потом по этой задаче классифицируем входной вектор.
- Осталось теперь объяснить, что всё это значит. :)

ПОСТАНОВКА ЗАДАЧИ

- Напомним, что наша задача поставлена следующим образом:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j \right) - \sum_{i=1}^{m} \alpha_i, \right.$$

$$\left. \text{где } \sum_{i=1}^{m} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

ПОСТАНОВКА ЗАДАЧИ

- Мы теперь хотим ввести некое отображение $\theta : \mathbb{R}^n \to \mathbb{R}^N$, $N > n$. Получится:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \left(\theta(\vec{x}_i) \cdot \theta(\vec{x}_j) \right) - \sum_{i=1}^{m} \alpha_i, \right.$$
$$\left. \text{где } \sum_{i=1}^{m} y_i \alpha_i = 0, \quad 0 \le \alpha_i \le C. \right\}$$

ТЕОРИЯ ГИЛЬБЕРТА--ШМИДТА--МЕРСЕРА

- Придётся немножко вспомнить (или изучить) функциональный анализ.
- Мы хотим обобщить понятие *скалярного произведения*; давайте введём новую функцию, которая (минуя трансформацию) будет сразу вычислять скалярное произведение векторов в feature space:

$$k(\vec{u}, \vec{v}) := \theta(\vec{u}) \cdot \theta(\vec{v}).$$

ТЕОРИЯ ГИЛЬБЕРТА--ШМИДТА--МЕРСЕРА

- Первый результат: любая симметрическая функция $k(\vec{u}, \vec{v}) \in L_2$ представляется в виде

$$k(\vec{u}, \vec{v}) = \sum_{i=1}^{\infty} \lambda_i \theta_i(\vec{u}) \cdot \theta_i(\vec{v}),$$

где $\lambda_i \in \mathbb{R}$ — собственные числа, а θ_i — собственные векторы интегрального оператора с ядром k, т.е.

$$\int k(\vec{u}, \vec{v}) \theta_i(\vec{u}) \mathrm{d}\vec{u} = \lambda_i \theta_i(\vec{v}).$$

ТЕОРИЯ ГИЛЬБЕРТА--ШМИДТА--МЕРСЕРА

- Чтобы k задавало скалярное произведение, достаточно, чтобы все собственные числа были положительными. А собственные числа положительны тогда и только тогда, когда (*теорема Мерсера*)

$$\int\int k(\vec{u},\vec{v})g(\vec{u})g(\vec{v})\mathrm{d}\vec{u}\mathrm{d}\vec{v} > 0$$

  для всех g таких, что $\int g^2(\vec{u})\mathrm{d}\vec{u} < \infty$.
- Вот, собственно и всё. Теперь мы можем вместо подсчёта $\theta(\vec{u}) \cdot \theta(\vec{v})$ в задаче квадратичного программирования просто использовать подходящее *ядро* $k(\vec{u},\vec{v})$.

ТЕОРИЯ ГИЛЬБЕРТА--ШМИДТА--МЕРСЕРА

- Итого задача наша выглядит так:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j) - \sum_{i=1}^{m} \alpha_i, \right.$$
$$\left. \text{где } \sum_{i=1}^{m} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C. \right\}$$

- Просто меняя ядро k, мы можем вычислять самые разнообразные разделяющие поверхности.
- Условия на то, чтобы k была подходящим ядром, задаются теоремой Мерсера.

ПРИМЕРЫ ЯДЕР

- Рассмотрим ядро
$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v})^2.$$
- Какое пространство ему соответствует?

ПРИМЕРЫ ЯДЕР

- После выкладок получается:

$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v})^2 = \\ = \left(u_1^2, u_2^2, \sqrt{2}u_1u_2\right) \cdot \left(v_1^2, v_2^2, \sqrt{2}v_1v_2\right).$$

- Иначе говоря, линейная поверхность в новом пространстве соответствует квадратичной поверхности в исходном (эллипс, например).

ПРИМЕРЫ ЯДЕР

- Естественное обобщение: ядро $k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v})^d$ задаёт пространство, оси которого соответствуют всем *однородным* мономам степени d.
- А как сделать пространство, соответствующее произвольной полиномиальной поверхности, не обязательно однородной?

ПРИМЕРЫ ЯДЕР

- Поверхность, описывающаяся полиномом степени d:

$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v} + 1)^d.$$

- Тогда линейная разделимость в feature space в точности соответствует полиномиальной разделимости в базовом пространстве.

ПРИМЕРЫ ЯДЕР

- Нормальное распределение (radial basis function):

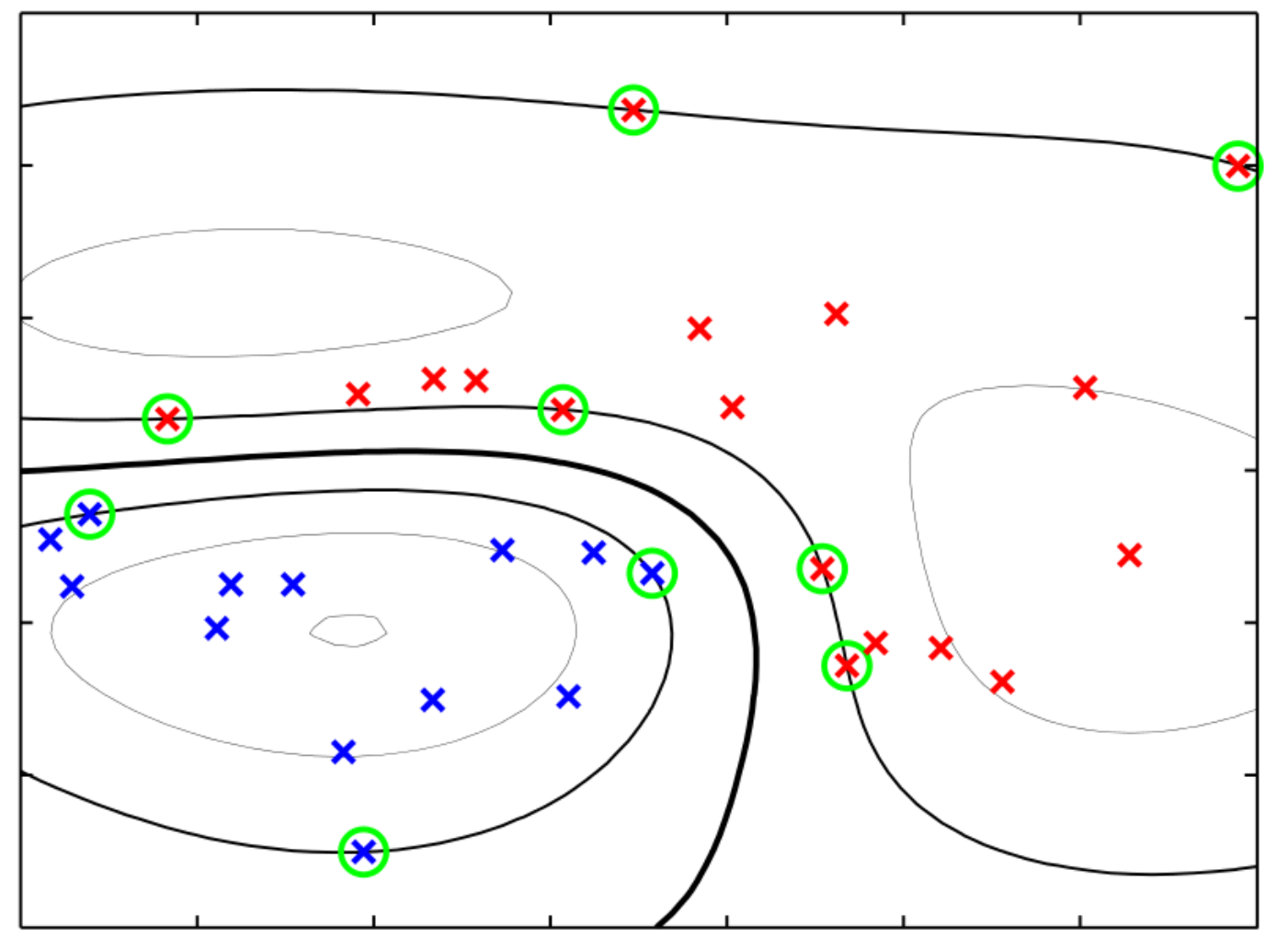
$$k(\vec{u}, \vec{v}) = e^{-\frac{||\vec{u}-\vec{v}||^2}{2\sigma}}.$$

- Двухуровневая нейронная сеть:

$$k(\vec{u}, \vec{v}) = o(\eta \vec{u} \cdot \vec{v} + c),$$

где o — сигмоид.

ПРИМЕР



РЕЗЮМЕ

- Вот какой получается в итоге алгоритм.
  1. Выбрать параметр C, от которого зависит акцент на минимизации ошибки или на максимизации зазора.
  2. Выбрать ядро и параметры ядра, которые у него, возможно, есть.
  3. Решить задачу квадратичного программирования.
  4. По полученным значениям опорных векторов определить w_0 (как именно?).
  5. Новые точки классифицировать как

$$f(\vec{x}) = \mathrm{sign}(\sum_i y_i \alpha_i k(\vec{x}, \vec{x}_i) - w_0).$$

ν-SVM

- Другой вариант для неразделимых данных – ν-SVM [Schölkopf et al., 2000].
- Максимизируем

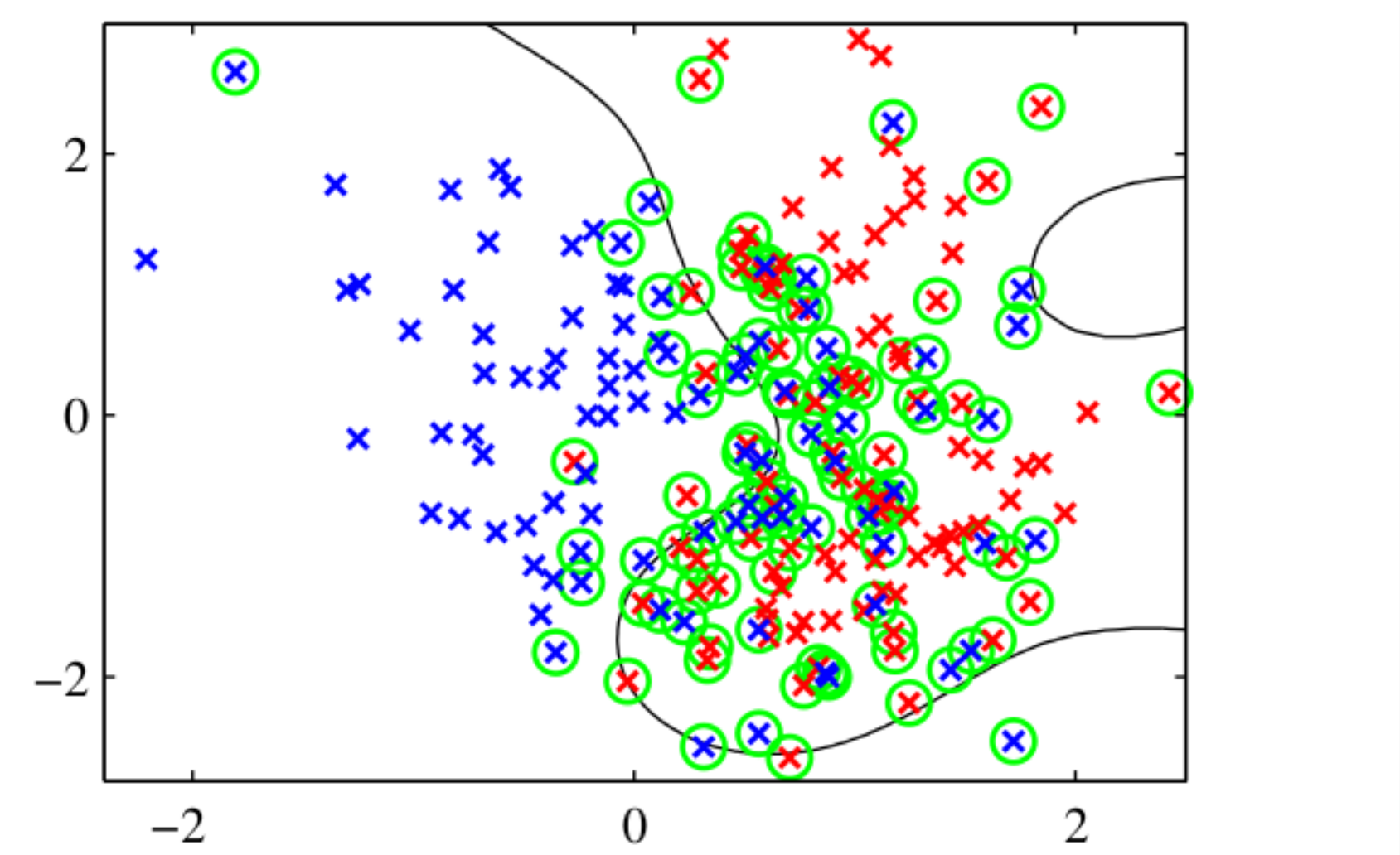
$$L(\mathbf{a}) = -\frac{1}{2}\sum_n \sum_m a_n a_m t_n t_m k\left(\mathbf{x}_n, \mathbf{x}_m\right)$$

  с ограничениями

$$0 \leq a_n \leq \frac{1}{N}, \ \sum_n a_n t_n = 0, \ \sum_n a_n \geq \nu.$$

- Параметр ν можно интерпретировать как верхнюю границу на долю ошибок.

SVM для классификации



Связь с логистической регрессией

- В случае SVM с возможными ошибками мы минимизируем

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2.$$

- Для точек с правильной стороны $\xi_n = 0$, с неправильной – $\xi_n = 1 - y_n t_n$.
- Так что можно записать *hinge error function* $E_{SV}(y_n t_n) = [1 - y_n t_n]_+$ и переписать как задачу с регуляризацией

$$\sum_{n=1}^{N} E_{SV}(y_n t_n) + \lambda\|\mathbf{w}\|^2.$$

Связь с логистической регрессией

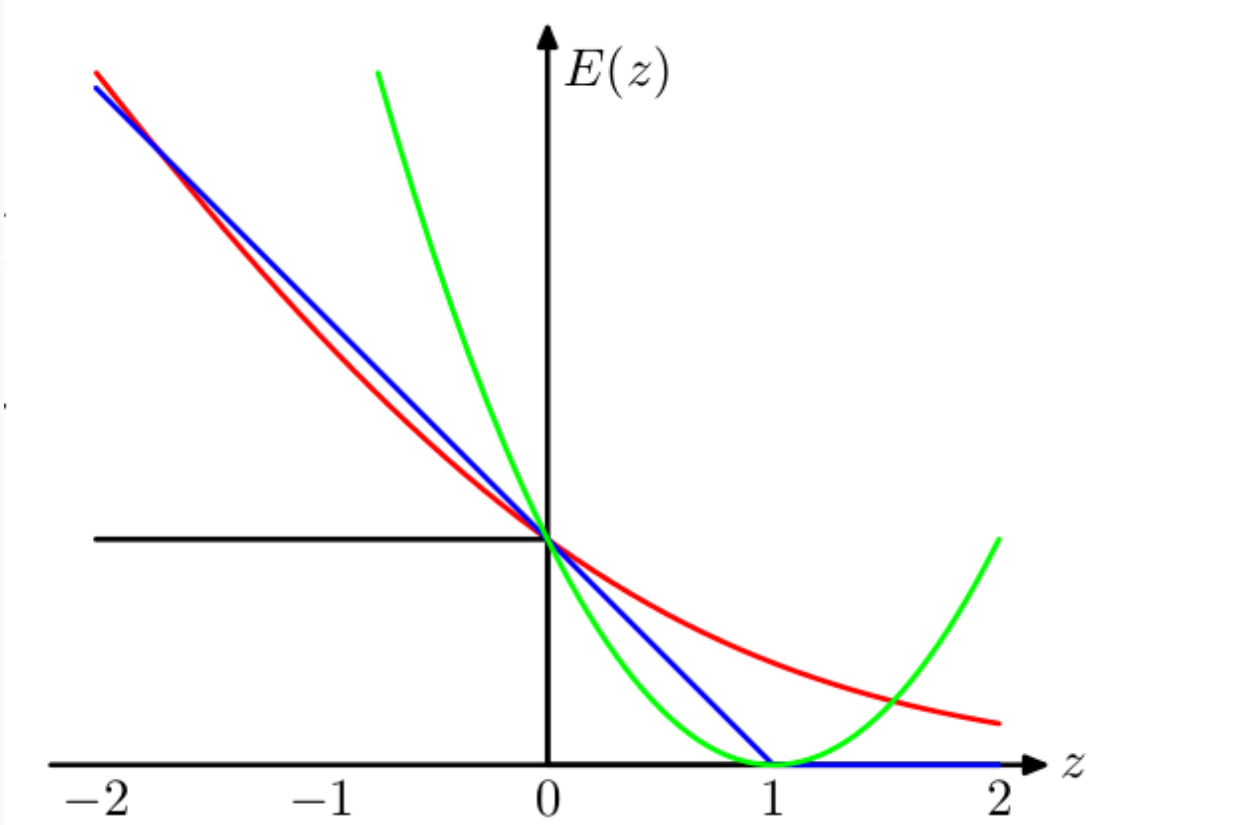
- Вспомним логистическую регрессию и переформулируем её для целевой переменной $t \in \{-1, 1\}$: $p(t = 1 \mid y) = \sigma(y)$, значит, $p(t = -1 \mid y) = 1 - \sigma(y) = \sigma(-y)$, и $p(t \mid y) = \sigma(yt)$.
- И логистическая регрессия – это минимизация

$$\sum_{n=1}^{N} E_{LR}(y_n t_n) + \lambda \|\mathbf{w}\|^2,$$

где $E_{LR}(y_n t_n) = \ln(1 + e^{-yt})$.

СВЯЗЬ С ЛОГИСТИЧЕСКОЙ РЕГРЕССИЕЙ

- График hinge error function, вместе с функцией ошибки для логистической регрессии:



SVM С НЕСКОЛЬКИМИ КЛАССАМИ

- Как обобщить SVM на несколько классов? Варианты (без подробностей):
  1. обучить одну против всех и классифицировать $y(\mathbf{x}) = \max_k y_k(\mathbf{x})$ (нехорошо, потому что задача становится несбалансированной, и $y_k(\mathbf{x})$ на самом деле несравнимы);
  2. можно сформулировать единую функцию для всех K SVM одновременно, но обучение становится гораздо медленнее;
  3. можно обучить попарно $K(K-1)/2$ классификаторов, а потом считать голоса – кто победит;
  4. DAGSVM: организуем попарные классификаторы в граф и будем идти по графу, для классификации выбирая очередной вопрос;
  5. есть даже методы, основанные на кодах, исправляющих ошибки.

SVM с одним классом

- SVM также можно использовать с *одним* классом.
- Как и зачем?

SVM с одним классом

- SVM также можно использовать с *одним* классом.
- Как и зачем?
- Можно при помощи SVM очертить границу области высокой плотности.
- Тем самым найдём выбросы данных (outliers).
- Задача будет такая: найти наименьшую поверхность (сферу, например), которая содержит все точки, кроме доли ν.

SVM ДЛЯ РЕГРЕССИИ

- SVM можно использовать для регрессии, сохраняя свойство разреженности (т.е. то, что SVM зависит только от опорных векторов).
- В обычной линейной регрессии мы минимизировали

$$\frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2.$$

- В SVM мы сделаем так: если мы попадаем в ϵ-окрестность предсказания, то ошибки, будем считать, совсем нет.

SVM ДЛЯ РЕГРЕССИИ

- ϵ-insensitive error function:

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & |y(\mathbf{x}) - t| < \epsilon, \\ |y(\mathbf{x}) - t| - \epsilon & \text{иначе.} \end{cases}$$

- И задача теперь выглядит как минимизация

$$C \sum_{n=1}^{N} E_\epsilon \left(y(\mathbf{x}_n) - t_n \right) + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

SVM ДЛЯ РЕГРЕССИИ

- Чтобы переформулировать, нужны по две slack переменные, для обеих сторон «трубки»:

$$y(\mathbf{x}_n) - \epsilon \leq t_n \leq y(\mathbf{x}_n) + \epsilon$$

превращается в

$$\begin{aligned} t_n &\leq y(\mathbf{x}_n) + \epsilon + \xi_n, \\ t_n &\geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n, \end{aligned}$$

и мы оптимизируем

$$C \sum_{n=1}^{N} E_\epsilon \left(\xi_n + \hat{\xi}_n \right) + \frac{\lambda}{2} \|\mathbf{w}\|^2.$$

SVM ДЛЯ РЕГРЕССИИ

- Если же теперь пересчитать дуальную задачу, то получится

$$L(\mathbf{a}, \hat{\mathbf{a}}) = -\frac{1}{2} \sum_n \sum_m (a_n - \hat{a}_n)(a_m - \hat{a}_m)\, k(\mathbf{x}_n, \mathbf{x}_m) -$$

$$- \epsilon \sum_{n=1}^{n} (a_n + \hat{a}_n) + \sum_{n=1}^{N} (a_n - \hat{a}_n)\, t_n,$$

и мы её минимизируем по $a_n, \hat{a}_n$ с условиями

$$0 \leq a_n \leq C,$$

$$0 \leq \hat{a}_n \leq C,$$

$$\sum_{n=1}^{N} (a_n - \hat{a}_n) = 0.$$

SVM ДЛЯ РЕГРЕССИИ

- Когда решим эту задачу, сможем предсказывать новые значения как

$$y(\mathbf{x}) = \sum_{n=1}^{N} (a_n - \hat{a}_n)\, k(\mathbf{x}, \mathbf{x}_n) + b,$$

где b можно найти как

$$b = t_n - \epsilon - \mathbf{w}^\top \phi(\mathbf{x}_n) =$$

$$= t_n - \epsilon - \sum_{m=1}^{N} (a_m - \hat{a}_m)\, k(\mathbf{x}_n, \mathbf{x}_m).$$

SVM ДЛЯ РЕГРЕССИИ

- А условия ККТ превращаются в

$$
\begin{aligned}
a_n\left(\epsilon + \xi_n + y(\mathbf{x}_n) - t_n\right) &= 0,\\
\hat{a}_n\left(\epsilon + \hat{\xi}_n - y(\mathbf{x}_n) + t_n\right) &= 0,\\
(C - a_n)\xi_n &= 0,\\
(C - \hat{a}_n)\hat{\xi}_n &= 0.
\end{aligned}
$$

- Отсюда очевидно, что либо a_n, либо $\hat{a}_n$ всегда равны 0, и хотя бы один из них не равен, только если точка лежит на или за границей «трубки».
- Опять получили решение, зависящее только от «опорных векторов».

SVM ДЛЯ РЕГРЕССИИ

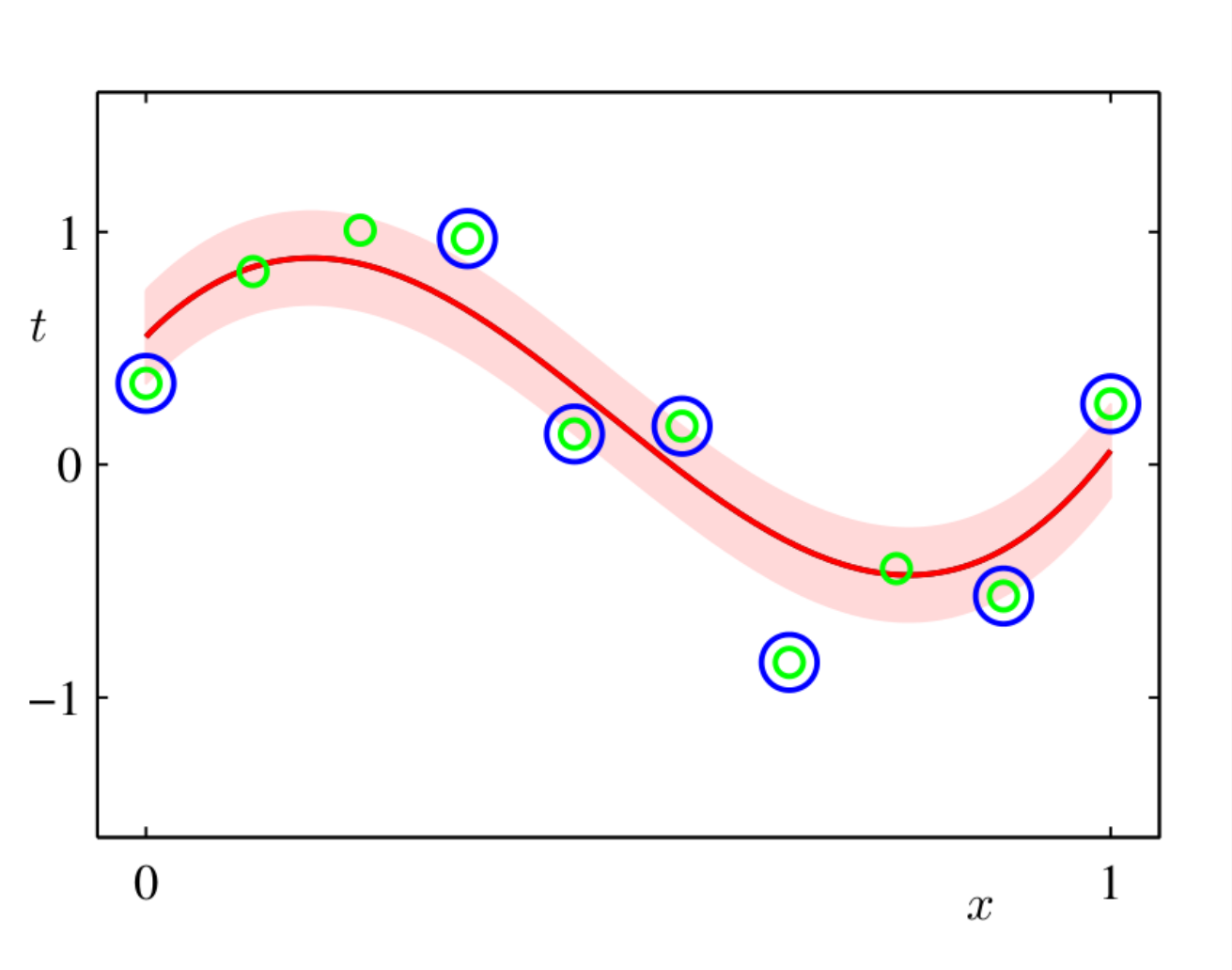
- Но снова можно переформулировать в виде ν-SVM, в котором параметр более интуитивно ясен: вместо ширины трубки ϵ рассмотрим ν – долю точек, лежащих вне трубки; тогда минимизировать надо

$$L(\mathbf{a}) = -\frac{1}{2}\sum_n\sum_m (a_n - \hat{a}_n)(a_m - \hat{a}_m)\, k(\mathbf{x}_n, \mathbf{x}_m) + \sum_{n=1}^{N}(a_n - \hat{a}_n)\, t_n$$

при условиях

$$\begin{array}{ll} 0 \le a_n \le \frac{C}{N}, & \sum_{n=1}^{N}(a_n - \hat{a}_n) = 0, \\ 0 \le \hat{a}_n \le \frac{C}{N}, & \sum_{n=1}^{N}(a_n + \hat{a}_n) \le \nu C. \end{array}$$

SVM ДЛЯ РЕГРЕССИИ



ПРАКТИКА

- На практике:
  - маленький C – гладкая разделяющая поверхность, мало опорных векторов;
  - большой C – сложная разделяющая поверхность, много опорных векторов.
- Для RBF ядра:
  - маленькое γ – опорные векторы влияют далеко, модель более простая;
  - большое γ – опорные векторы влияют только непосредственно рядом, модель более сложная.

СПАСИБО!

Спасибо за внимание!