Машинное обучение

Лекция 6. Знакомство с линейными моделями

Виктор Кантор

План

I. Линейная классификация
II. Линейная регрессия

I. Линейная классификация

Линейная классификация

- Линейный классификатор на простых примерах
- Формализация линейного классификатора
- Функции потерь
- Метод стохастического градиентного спуска (SGD)
- Регуляризация
- Стандартные классификаторы

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент
2. Вам хочется где-то поесть
3. Вам хочется спать
4. Вам хочется увидеться с друзьями

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть
3. Вам хочется спать
4. Вам хочется увидеться с друзьями

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать
4. Вам хочется увидеться с друзьями

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями **+4**

Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями **+4**

Порог для решающего правила: +1

Если сумма больше – выходим :)

Более серьезный пример: дать кредит или нет

Признаки (1/0):

- Работоспособный возраст
- Имеет счет в вашем банке
- Много просрочек по платежам за другие кредиты
- Нет просрочек по платежам за другие кредиты, причем другие кредиты есть

Скоринговые карты

ПОКАЗА-ТЕЛЬ	ДИАПАЗОН ЗНАЧЕНИЙ
Возраст заемщика	До 35 лет
	От 35 до 45 лет
	От 45 и старше
Образова-ние	Высшее
	Среднее специальное
	Среднее
Состоит ли в браке	Да
	Нет
Наличие кредита в прошлом	Да
	Нет
Стаж работы	До 1 года
	От 1 до 3 лет
	От 3 до 6 лет
	Свыше 6 лет
Наличие автомобиля	Да
	Нет

Скоринговые карты

ПОКАЗА-ТЕЛЬ	ДИАПАЗОН ЗНАЧЕНИЙ	СКОРИНГ-БАЛЛ
Возраст заемщика	До 35 лет	7,60
	От 35 до 45 лет	29,68
	От 45 и старше	35,87
Образова-ние	Высшее	29,82
	Среднее специальное	20,85
	Среднее	22,71
Состоит ли в браке	Да	29,46
	Нет	9,38
Наличие кредита в прошлом	Да	40,55
	Нет	13,91
Стаж работы	До 1 года	15,00
	От 1 до 3 лет	18,14
	От 3 до 6 лет	19,85
	Свыше 6 лет	23,74
Наличие автомобиля	Да	51,69
	Нет	15,93

Подбор весов признаков и порога

Почему нельзя продолжать также:

- Сложно настраивать вручную
- Требуется эксперт в области
- Требуется проверка на данных и уточнение весов (эксперт может что-то не учесть)

Подбор весов признаков и порога

Почему нельзя продолжать также:

- Сложно настраивать вручную
- Требуется эксперт в области
- Требуется проверка на данных и уточнение весов (эксперт может что-то не учесть)

Решение – автоматизируем подбор параметров: придумаем функцию от параметров, которую надо минимизировать, и используем методы численной оптимизации

Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, \text{если } f(x) > 0 \\ -1, \text{если } f(x) > 0 \end{cases}$$

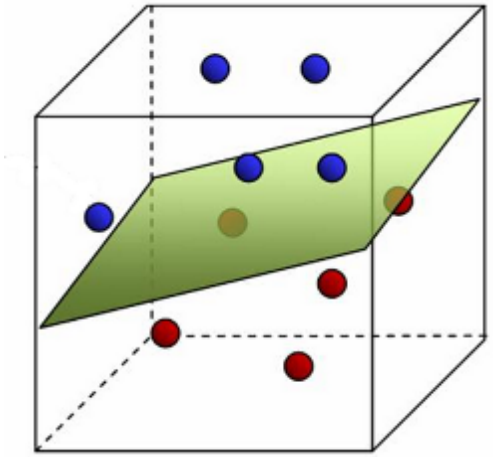
$$f(x) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, \text{если } f(x) > 0 \\ -1, \text{если } f(x) > 0 \end{cases}$$

$$f(x) = w_0 + w_1 x_1 + \cdots + w_n x_n = w_0 + \langle w, x \rangle$$

Геометрическая интерпретация:
разделяем классы плоскостью

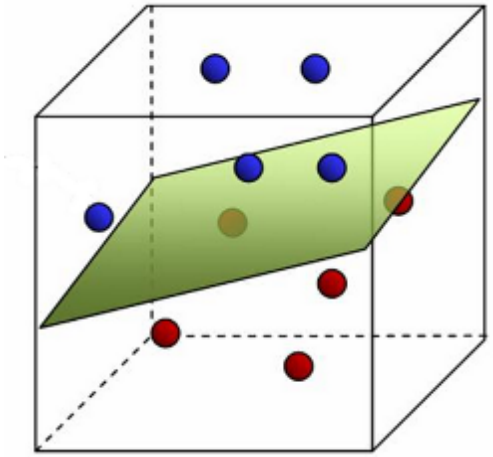


Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, \text{если } f(x) > 0 \\ -1, \text{если } f(x) > 0 \end{cases}$$

$$f(x) = \langle w, x \rangle$$

Геометрическая интерпретация:
разделяем классы плоскостью



Как выглядит код: применение модели

```
import numpy as np

def f(x):
    return np.dot(w, x) + w0

def a(x):
    return 1 if f(x) > 0 else 0
```

Отступ (margin)

Отступом алгоритма $a(x) = sign\{f(x)\}$ на объекте x_i называется величина $M_i = y_i f(x_i)$

(y_i - класс, к которому относится x_i)

$$M_i \leq 0 \Leftrightarrow y_i \neq a(x_i)$$
$$M_i > 0 \Leftrightarrow y_i = a(x_i)$$

Функция потерь

$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0]$$

Функция потерь

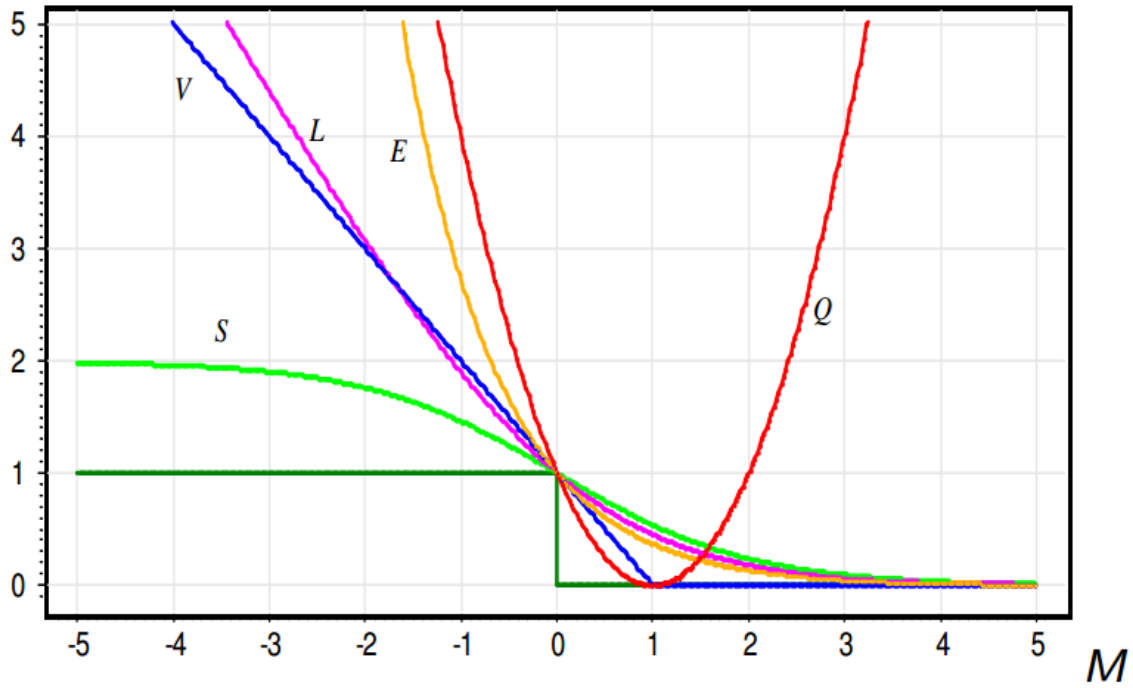
$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0] \quad \leqslant \quad \widetilde{Q}(w) = \sum_{i=1}^{\ell} \mathscr{L}(M_i(w)) \to \min_w;$$

Функция эмпирического риска

Функция потерь

Функция потерь

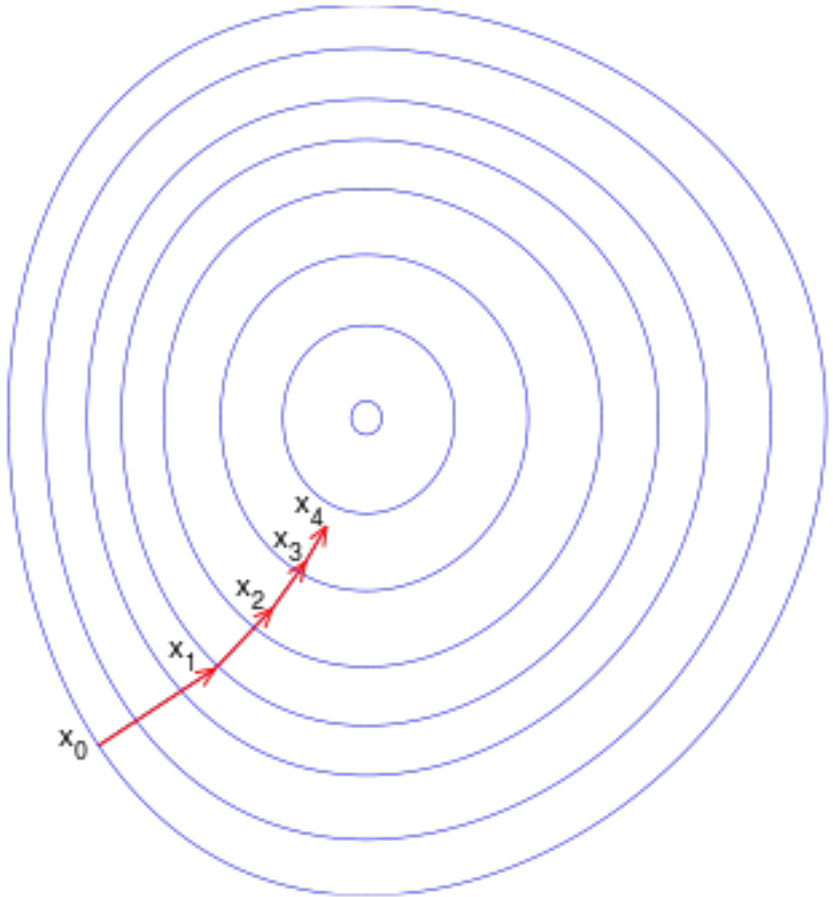
$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0] \quad \leqslant \quad \widetilde{Q}(w) = \sum_{i=1}^{\ell} \mathscr{L}(M_i(w)) \to \min_w;$$



$$Q(M) = (1 - M)^2$$
$$V(M) = (1 - M)_+$$
$$S(M) = 2(1 + e^M)^{-1}$$
$$L(M) = \log_2(1 + e^{-M})$$
$$E(M) = e^{-M}$$

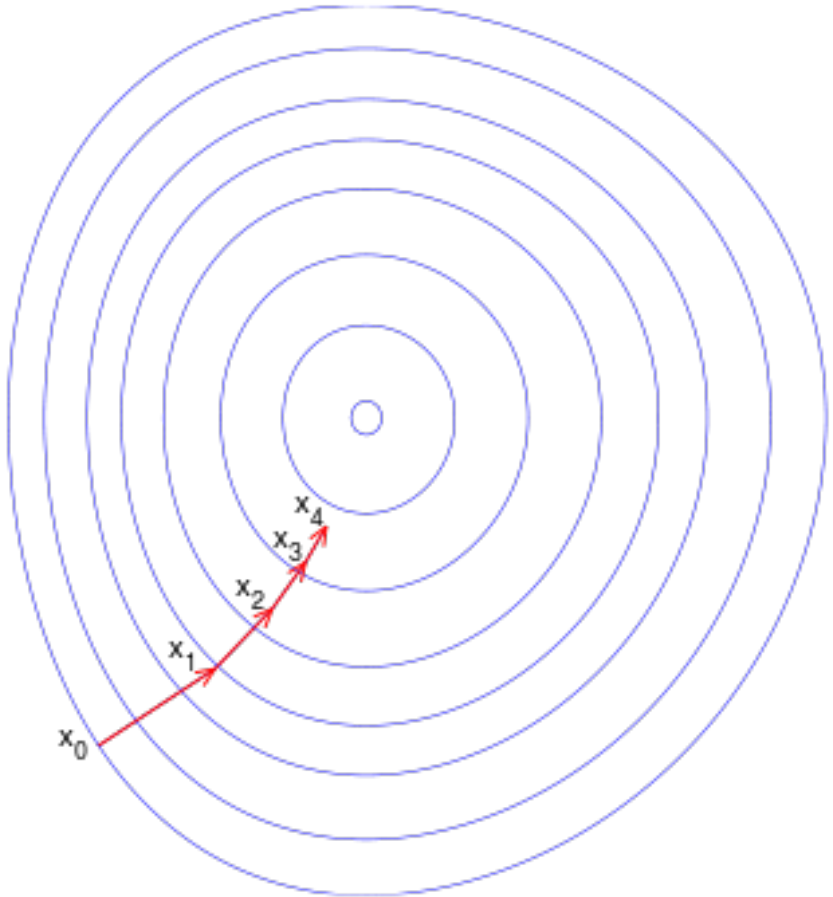
Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \ n \geq 0.$$



Градиентный спуск

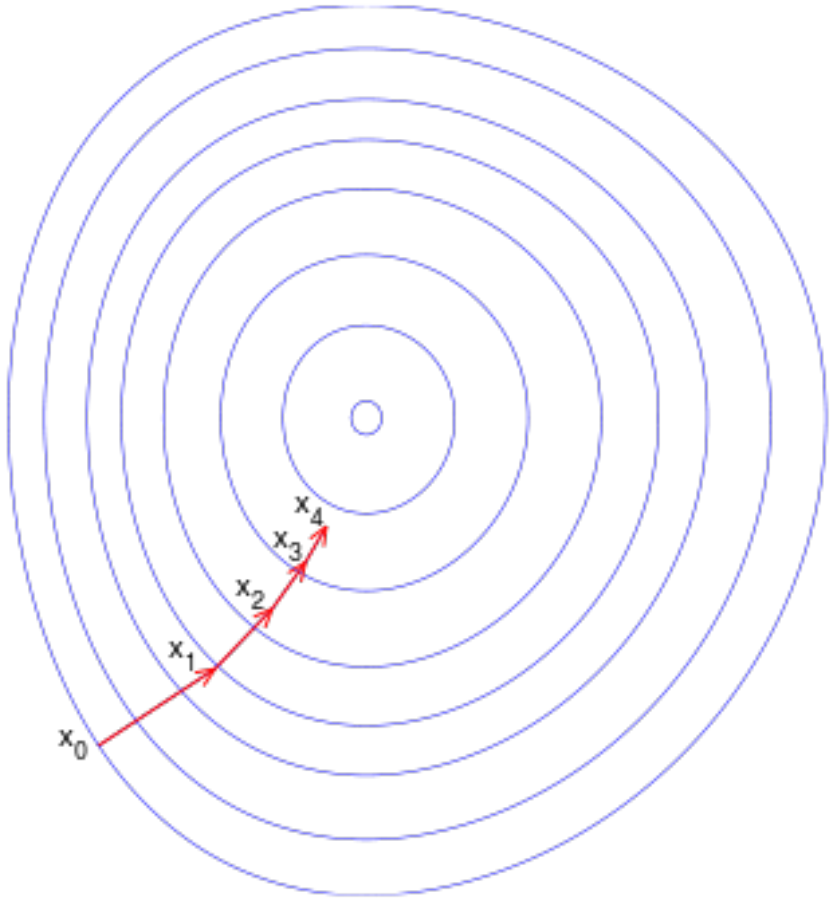
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$



$$\nabla_w \tilde{Q} = \sum_{i=1}^{l} \nabla L(M_i)$$

Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$

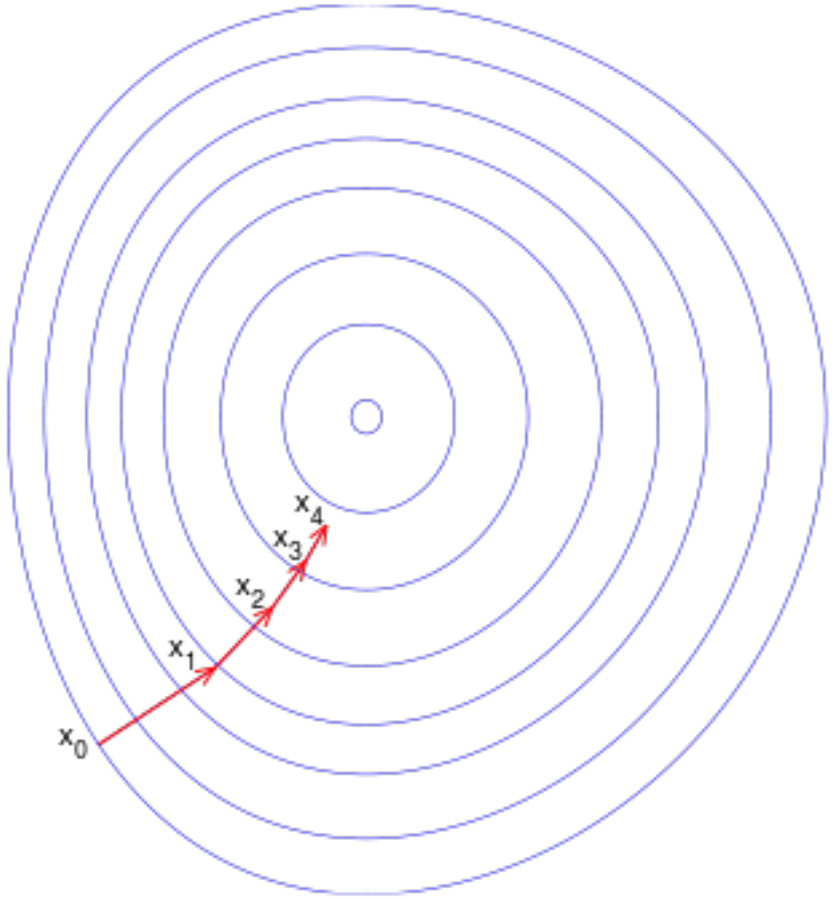


$$\nabla_w \tilde{Q} = \sum_{i=1}^{l} \nabla L(M_i)$$

$$\nabla \tilde{Q} = \sum_{i=1}^{l} L'(M_i) \frac{\partial M_i}{\partial w}$$

Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \ n \geq 0.$$



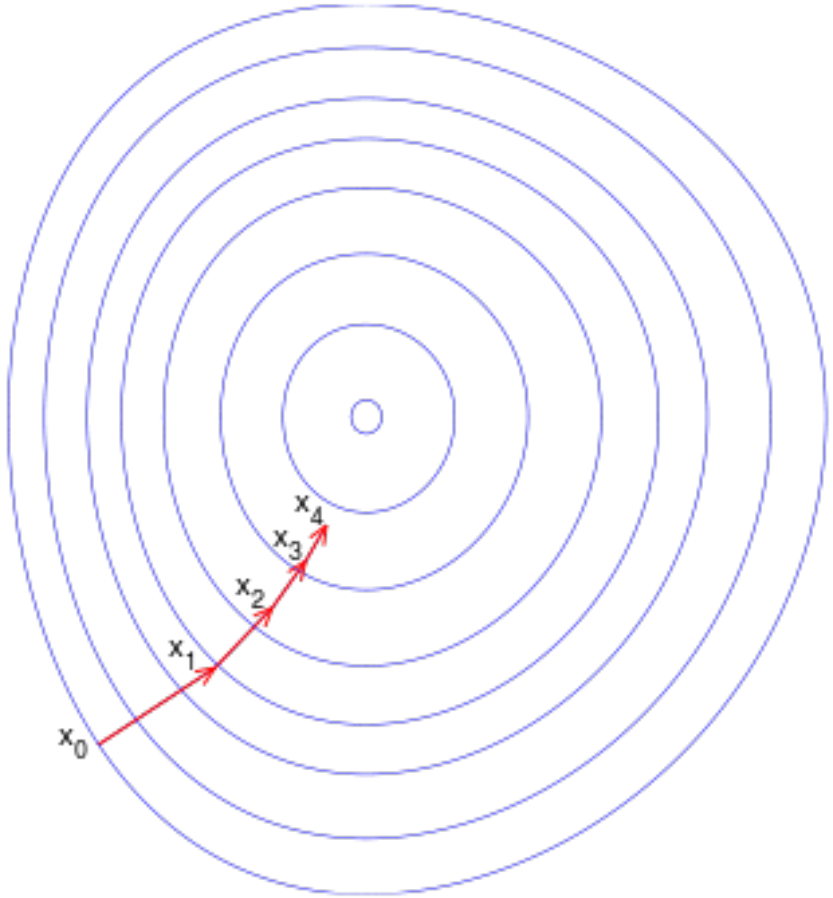
$$\nabla_w \tilde{Q} = \sum_{i=1}^{l} \nabla L(M_i)$$

$$\nabla \tilde{Q} = \sum_{i=1}^{l} L'(M_i) \frac{\partial M_i}{\partial w}$$

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \ n \geq 0.$$



$$\nabla_w \tilde{Q} = \sum_{i=1}^{l} \nabla L(M_i)$$

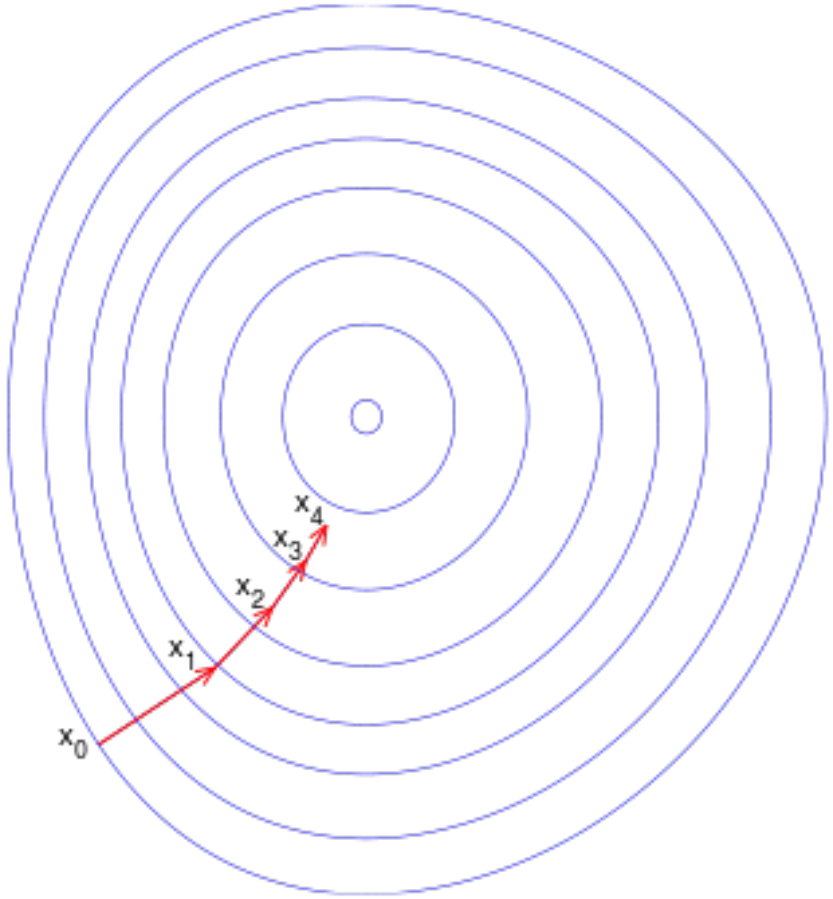
$$\nabla \tilde{Q} = \sum_{i=1}^{l} L'(M_i) \frac{\partial M_i}{\partial w}$$

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

$$\nabla \tilde{Q} = \sum_{i=1}^{l} y_i x_i L'(M_i)$$

Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \ n \geq 0.$$



$$\nabla_w \tilde{Q} = \sum_{i=1}^{l} \nabla L(M_i)$$

$$\nabla \tilde{Q} = \sum_{i=1}^{l} L'(M_i) \frac{\partial M_i}{\partial w}$$

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

$$\nabla \tilde{Q} = \sum_{i=1}^{l} y_i x_i L'(M_i)$$

$$w_{n+1} = w_n - \gamma_n \sum_{i=1}^{l} y_i x_i L'(M_i)$$

Стохастический градиент

$$w_{n+1} = w_n - \gamma_n \sum_{i=1}^{l} y_i x_i L'(M_i)$$

$$w_{n+1} = w_n - \gamma_n y_i x_i L'(M_i)$$

x_i – случайный элемент обучающей выборки

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

$$L(M) = \max\{0, 1 - M\} = (1 - M)_+$$

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

$$\gamma_n = (\alpha + n)^{-\beta}$$

Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

$$\gamma_n = (\alpha + n)^{-\beta}$$

$$\gamma_n = \tau\beta_n$$

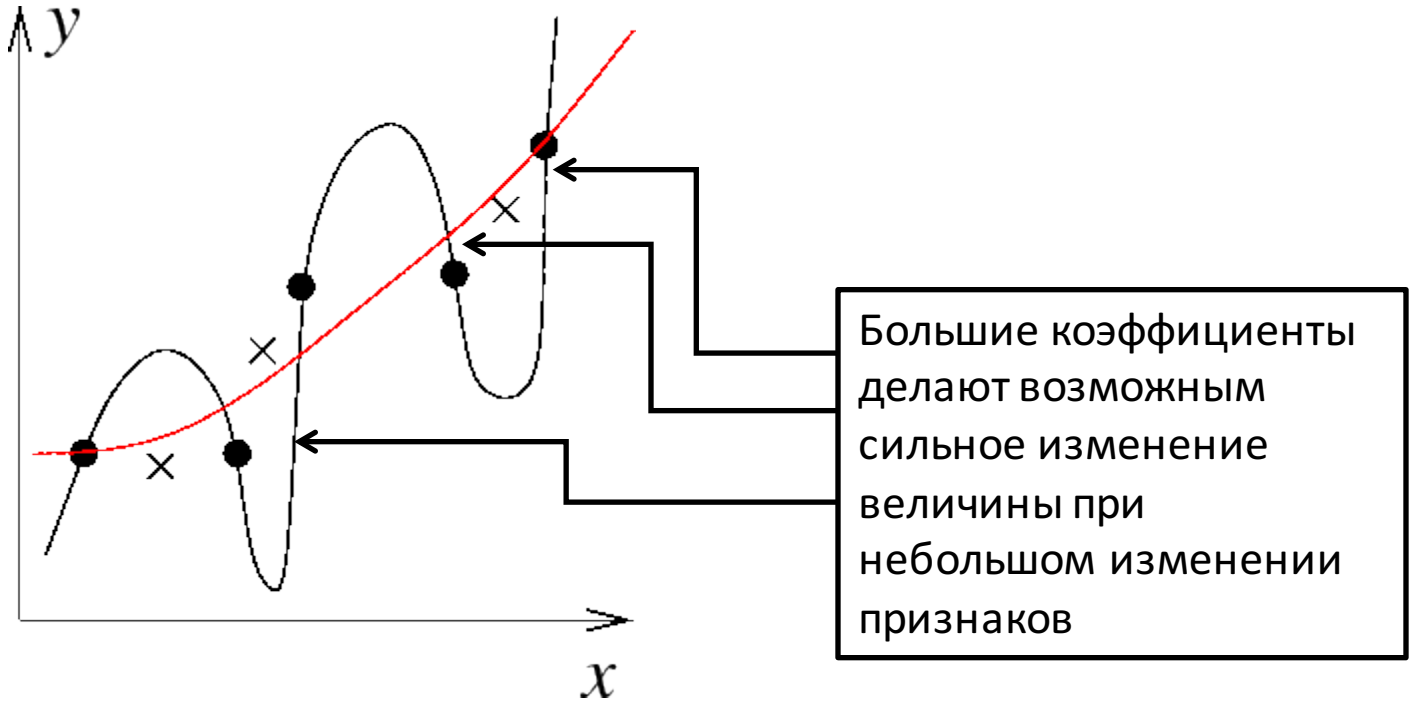
Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$w_{n+1} = w_n - \gamma_n y_i x_i L'(M_i)$$

Регуляризация

- Переобучение в задаче обучения с учителем как правило означает большие коэффициенты:



- Идея: добавить ограничение на коэффициенты

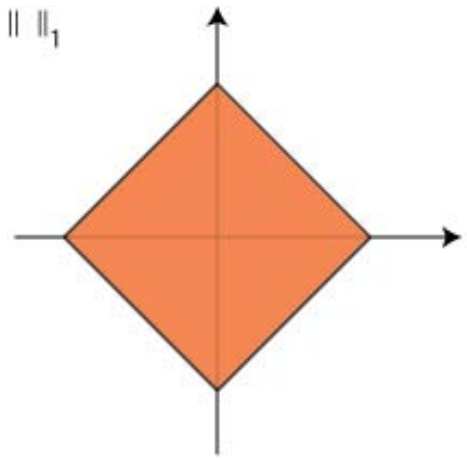
Регуляризация

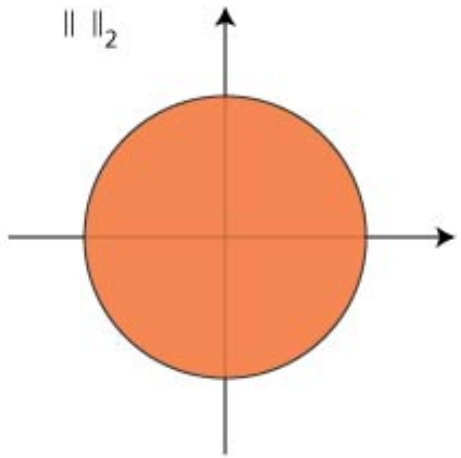
$$\begin{cases} \tilde{Q} = \sum_{i=1}^{l} L(M_i) \to min \\ \sum_{k=1}^{m} |w_k| \leq \tau \end{cases}$$

$l1$ – регуляризация

$$\begin{cases} \tilde{Q} = \sum_{i=1}^{l} L(M_i) \to min \\ \sum_{k=1}^{m} {w_k}^2 \leq \tau \end{cases}$$

$l2$ – регуляризация





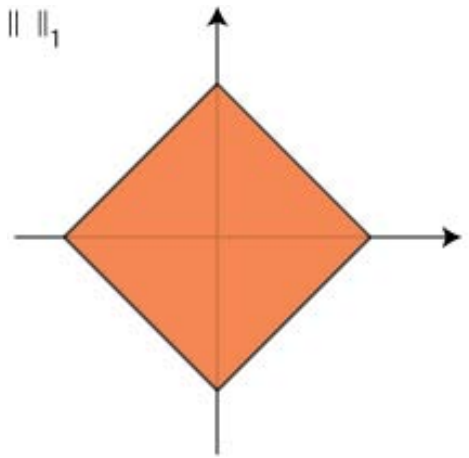
Регуляризация

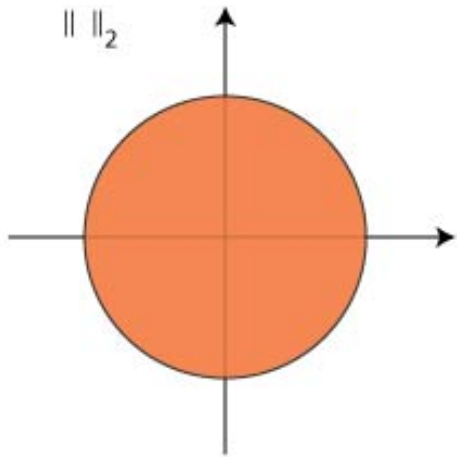
$$\sum_{i=1}^{l} L(M_i) + \gamma \sum_{k=1}^{m} |w_k| \to min$$

$$\sum_{i=1}^{l} L(M_i) + \gamma \sum_{k=1}^{m} {w_k}^2 \to min$$

$l1$ – регуляризация

$l2$ – регуляризация



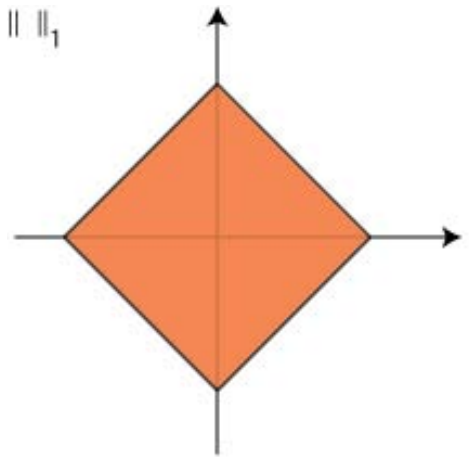


Регуляризация

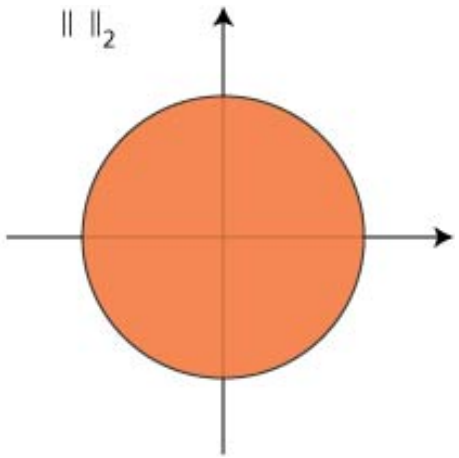
$$\sum_{i=1}^{l} L(M_i) + \gamma \sum_{k=1}^{m} |w_k| \to min$$

$$\sum_{i=1}^{l} L(M_i) + \gamma \sum_{k=1}^{m} {w_k}^2 \to min$$

$l1$ – регуляризация



$l2$ – регуляризация



Вопрос:
вы заметили, что в регуляризатор не включается вес w_o?

Различия между *l1* и *l2*

- Разреженность – l1-регуляризация делает вектор весов более разреженным (содержащим больше нулей)
- В случае линейной классификации это означает отбор признаков: признаки с нулевыми весами не используются в классификации

Упражнение

Выпишете, как поменяется правило обновления весов признаков в линейном классификаторе с помощью SGD при добавлении регуляризатора

Стандартные линейные классификаторы

Классификатор	Функция потерь	Регуляризатор
SVM (Support vector machine, метод опорных векторов)	$L(M) = \max\{0, 1 - M\} = = (1 - M)_+$	$\sum_{k=1}^{m} {w_k}^2$
Логистическая регрессия	$L(M) = \log(1 + e^{-M})$	Обычно $\sum_{k=1}^{m} {w_k}^2$ или $\sum_{k=1}^{m} \lvert w_k \rvert$

Обязательно ли функция потерь – функция от отступа?

Пример:

$$y_i \in \{0, 1\} \qquad Q = -\sum_{i=1}^{\ell} y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \to \min_w$$

$$p_i = \sigma(\langle w, x_i \rangle) = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

Обязательно ли функция потерь – функция от отступа?

Пример:

$$y_i \in \{0, 1\} \qquad Q = -\sum_{i=1}^{\ell} y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \to \min_w$$

$$p_i = \sigma(\langle w, x_i \rangle) = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

Упражнение:

Показать, что это та же оптимизационная задача, что и в логистической регрессии

Общий случай

$$Q = \sum_{i=1}^{\ell} L\big(y_i, f(x_i)\big) + \gamma V(w) \to \min_w$$

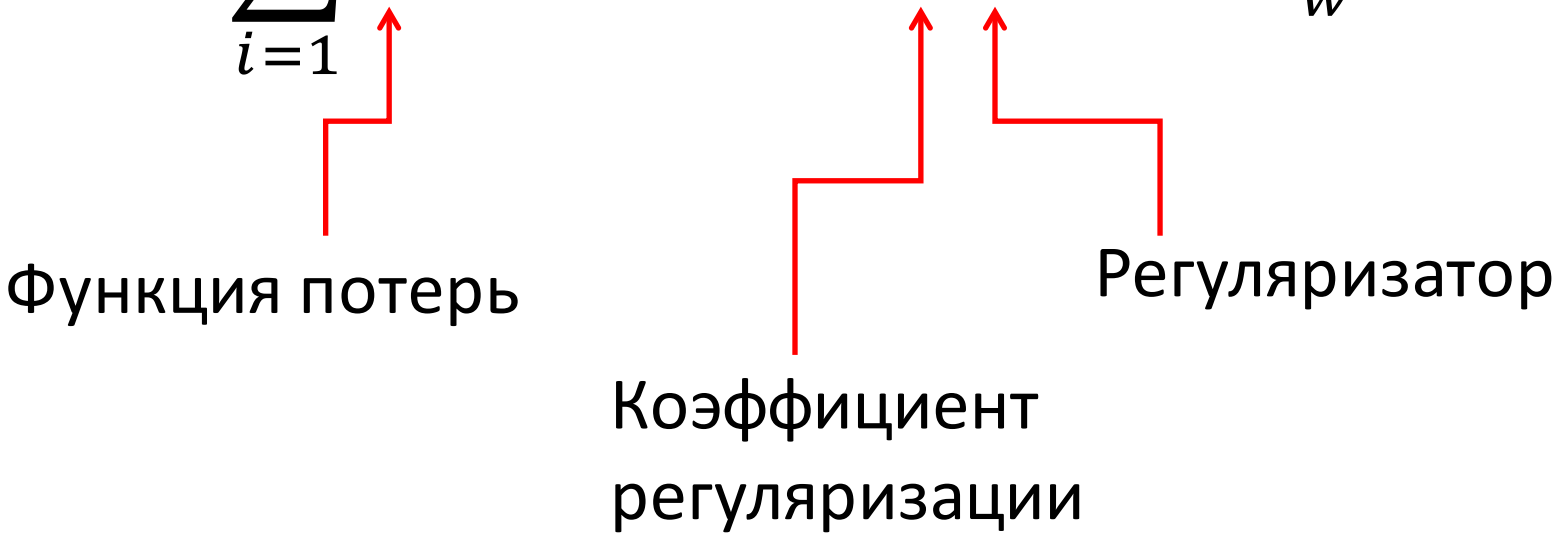
Функция потерь

Коэффициент регуляризации

Регуляризатор

Общий случай

$$Q = \sum_{i=1}^{\ell} L\big(y_i, f(x_i)\big) + \gamma V(w) \to \min_w$$



Упражнение:

Как будет меняться качество на обучающей и на тестовой выборке с ростом коэффициента регуляризации в SVM и в логистической регрессии в sklearn? Выясните, почему результат такой.

II. Линейная регрессия

Линейная регрессия

- Модель и матричная запись
- Аналитический вывод
- Геометрическая интерпретация
- Регуляризация: LASSO и гребневая регрессия

Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^{N} L\big(y_i, a(x_i)\big)$$

Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^{N} L\big(y_i, a(x_i)\big)$$

$$L\big(y_i, a(x_i)\big) = \ (y_i - a(x_i))^2$$

Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

$$Q = \sum_{i=1}^{N} L\big(y_i, a(x_i)\big)$$

$$L\big(y_i, a(x_i)\big) = \ (y_i - a(x_i))^2$$

$$L\big(y_i, a(x_i)\big) = |y_i - a(x_i)|$$

Модель и матричная запись

$$\text{Модель: } y_i \approx \hat{y}_i = < w, x_i > + w_0$$

Если добавить $x_{i0} = 1$:

$$y_i \approx \hat{y}_i = < w, x_i >$$

$$y_1 \approx \hat{y}_1 = x_1{}^T w$$
$$\dots$$
$$y_i \approx \hat{y}_i = x_i{}^T w$$
$$\dots$$
$$y_l \approx \hat{y}_l = x_l{}^T w$$

Модель и матричная запись

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_l \end{pmatrix} \approx \begin{pmatrix} \widehat{y_1} \\ \widehat{y_2} \\ \dots \\ \widehat{y_l} \end{pmatrix} = \begin{pmatrix} x_1{}^T \\ x_2{}^T \\ \dots \\ x_l{}^T \end{pmatrix} w$$

$$y \approx \hat{y} = Fw$$

$$w = \underset{w}{\operatorname{argmin}} \|y - \hat{y}\|^2$$

Аналитический вывод

$$\frac{\partial (y - Fw)^2}{\partial w} = 2F^T(y - Fw) = 0$$

$$F^T F w = F^T y$$

$$\boxed{w = (F^T F)^{-1} F^T y}$$

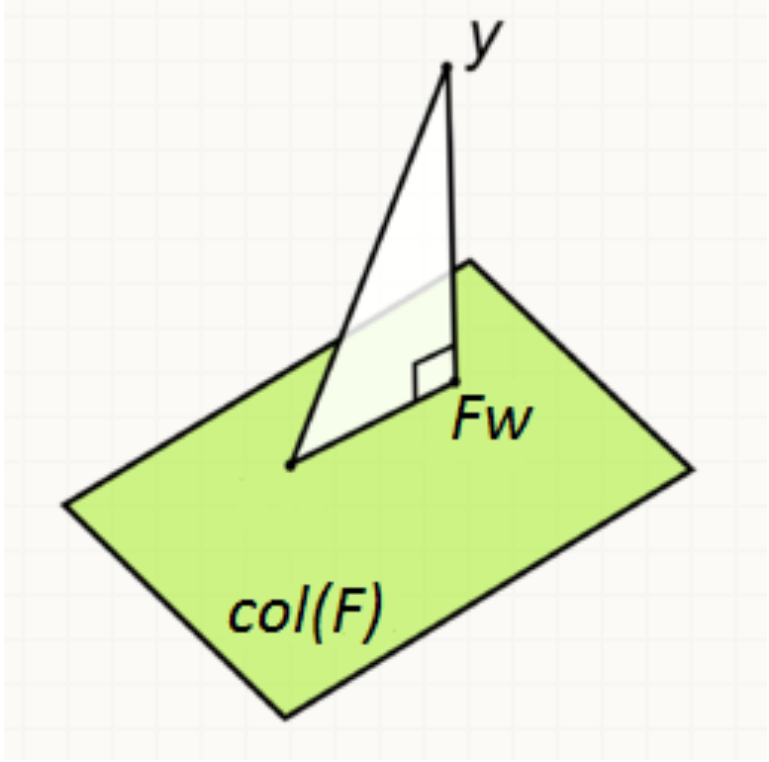
Геометрическая интерпретация

$$\begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_l \end{pmatrix} \approx \begin{pmatrix} \widehat{y_1} \\ \widehat{y_2} \\ \cdots \\ \widehat{y_l} \end{pmatrix} = (F_{(1)} \quad \cdots \quad F_{(m)})w$$

$$y \approx \hat{y} = Fw = w_1 F_{(1)} + \cdots + w_m F_{(m)}$$

Геометрическая интерпретация

$$(Fw - y) \perp F_{(k)} \; \forall k = 1, \ldots, m$$



$$F_{(k)}{}^{T}(Fw - y) = 0 \quad \forall k$$

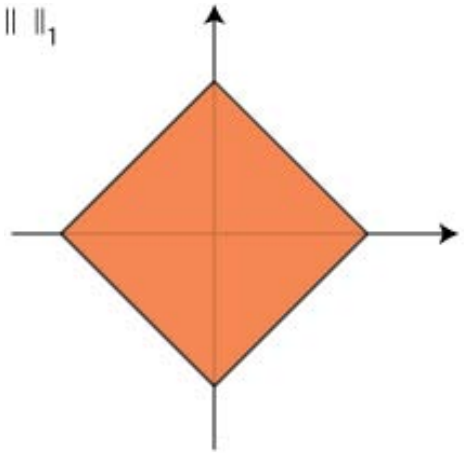
$$F^{T}(Fw - y) = 0$$

$$F^{T}Fw = F^{T}y$$

$$w = (F^{T}F)^{-1}F^{T}y$$

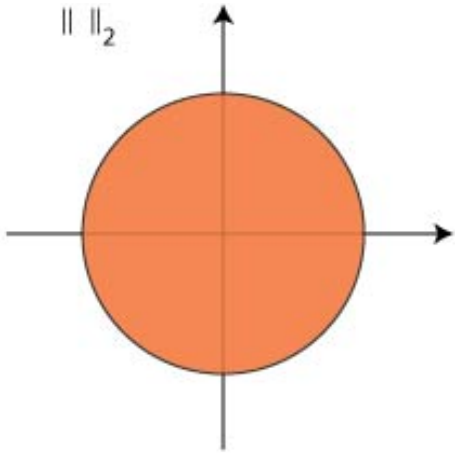
LASSO и гребневая регрессия

$l1 -$ регуляризация



LASSO

$l2 -$ регуляризация



Ridge

Гребневая регрессия ($\ell 2$-регуляризация)

$$\sum_{i=1}^{l}(a(x_i) - y_i)^2 + \gamma \sum_{k=1}^{m} {w_k}^2 \to min$$

Гребневая регрессия ($\ell 2$-регуляризация)

$$\sum_{i=1}^{l}(a(x_i)-y_i)^2+\gamma\sum_{k=1}^{m}{w_k}^2\to min$$

$$\frac{\partial\left((y-Fw)^2+\gamma w^2\right)}{\partial w}=2F^T(y\ -Fw)+2\gamma w=0$$

$$(F^TF+\gamma I)w=F^Ty$$

$$\boxed{w=(F^TF+\gamma I)^{-1}F^Ty}$$

Библиотеки

- libSVM
- liblinear
- sklearn.linear_models
- Vowpal Wabbit

Резюме

I. Линейная классификация
II. Линейная регрессия

Преимущества:

- легко реализовывать уже обученную модель
- не многим сложнее реализовывать и ее обучение
- быстро работают
- хорошо работают, когда много признаков
- нормально работают, когда мало данных

Резюме

I. Линейная классификация
II. Линейная регрессия

Недостатки:

- может быть слишком простым для вашей зависимости y(x)