

Лекция 6:

Методы оптимизации для задач машинного обучения

Задача оптимизации

Безусловная:

$$\min f(x)$$

$x \in \mathbb{R}^n$ — переменные

$f: \mathbb{R}^n \rightarrow \mathbb{R}^1$ — целевая функция

$x^* \in \operatorname{argmin}_X f(x)$ — точка минимума,

$f^* = f(x^*)$ — значение минимума

Условная:

$$\min f(x)$$

$x \in X$ — множество ограничений

$f: Y \rightarrow \mathbb{R}^1, X, Y \subseteq \mathbb{R}^n, X \subseteq Y$

Определение минимума функции

Точка x^* называется точкой *локального минимума* функции $f(x)$, если существует $\varepsilon > 0$ такое, что $f(x) \geq f(x^*)$ для всех $x: \|x - x^*\| \leq \varepsilon$

Точка x^* называется точкой *глобального минимума* функции $f(x)$, если $f(x) \geq f(x^*)$ для всех $x \in \mathbb{R}^n$

Глобальное решение является локальным, обратное неверно

Может быть много локальных решений, глобальное решение единственно

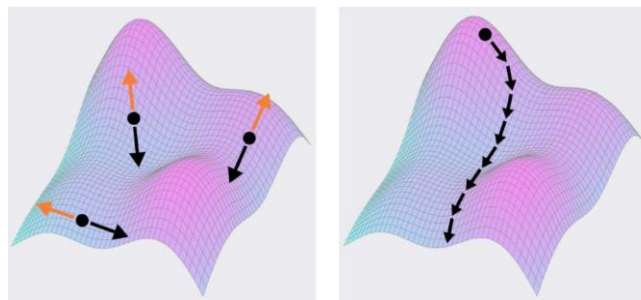


Некоторые важные определения

$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$

Градиент (вектор первых производных) – направление наибольшего роста функции в точке (можно доказать через разложение в ряд Тейлора и неравенство Коши-Буняковского):

$$f'(x) = \nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right)$$



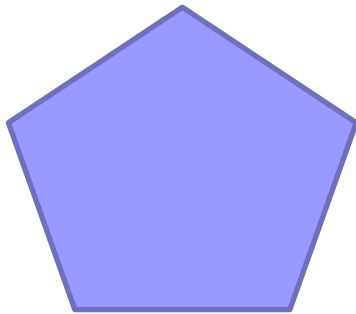
Матрица вторых производных, матрица Гессе, гессиан:

$$f''(x) = \nabla^2 f(x) = \left(\frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right)_{i,j=\overline{1,n}}$$

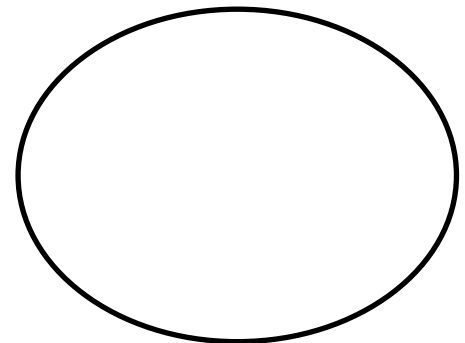
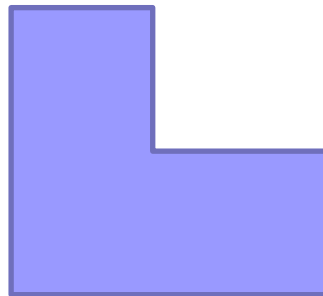
Выпуклые множества

Определение. Непустое множество $X \subseteq \mathbb{R}^n$ называется выпуклым, если для любых $x, y \in X, \lambda \in [0, 1]$ выполняется $\lambda x + (1 - \lambda)y \in X$

Выпуклое



Невыпуклые



Свойства выпуклых множеств:

- Пересечение выпуклых множеств выпукло
- Линейная комбинация выпуклых множеств выпукло
- Проекция на выпуклое множество единственна

Выпуклые функции

Функция $f: \mathbb{R}^n \rightarrow \mathbb{R}^1$, определенная на выпуклом множестве $X \subset \mathbb{R}^n$, называется *выпуклой*, если для любых $x, y \in X, \lambda \in [0,1]$:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

строго выпуклой

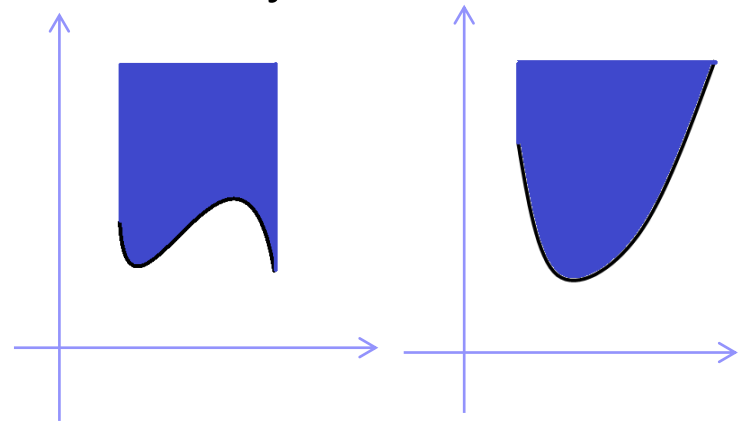
$$f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda)f(y)$$

сильно выпуклой

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) - \alpha \lambda(1 - \lambda)\|x - y\|^2$$

Свойства:

- Линейная комбинация выпуклых функций выпукла
- Надграфики выпуклой функции является выпуклым множеством



Свойства выпуклых функций для задач оптимизации

Выпуклая функция достигает максимума и минимума в **единственной точке**

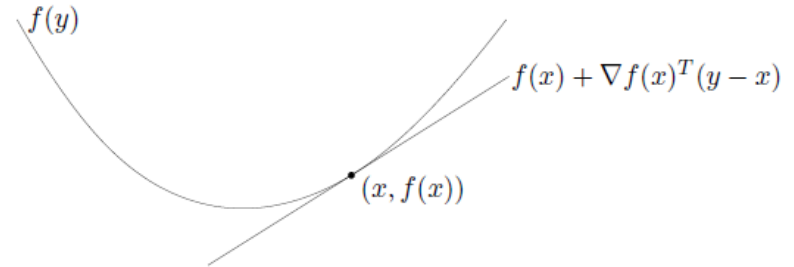
Неравенство Йенсена:

Пусть $f(x)$ – выпуклая функция на выпуклом множестве X , тогда:

$$f(\sum_{i=1}^m \lambda_i x^i) \leq \sum_{i=1}^m \lambda_i f(x^i),$$

$$\forall m = 1, 2, \dots, x^i \in X,$$

$$\lambda_i \geq 0, i = \overline{1, m}, \sum_{i=1}^m \lambda_i = 1$$



Условие выпуклости первого порядка:

- График функции лежит не ниже касательной гиперплоскости к нему в точке

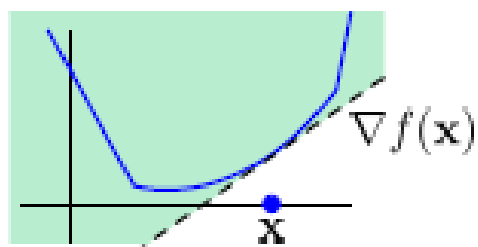
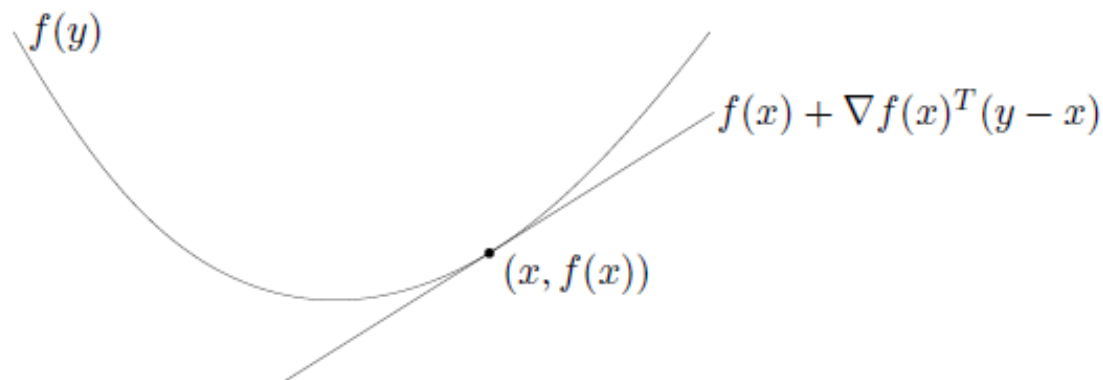
$$\forall x, y \in X, f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle$$

- Строго выпукла, если строгое неравенство
- Сильно выпукла, если есть «отступ»

$\exists \lambda$:

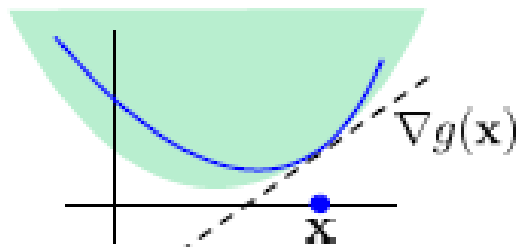
$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle + \lambda \|y - x\|^2 / 2$$

Условие выпуклости



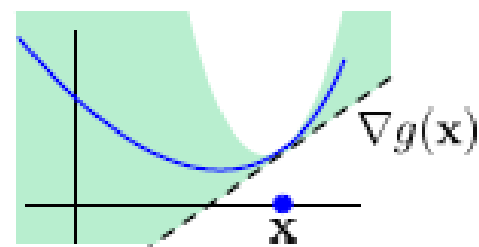
— $f : \mathbb{R}^d \rightarrow \mathbb{R}$

выпуклая



— $g : \mathbb{R}^d \rightarrow \mathbb{R}$

строго



— $g : \mathbb{R}^d \rightarrow \mathbb{R}$

сильно

Необходимые и достаточные условия экстремума

Условия первого порядка:

- **Теорема. (необходимое условие)** Если x^* - точка минимума дифференцируемой в точке x^* функции $f(x)$. Тогда $\nabla f(x^*) = 0$.
- **Теорема (достаточное условие).** Пусть $f(x)$ – выпуклая функция, дифференцируемая в точке x^* , и $\nabla f(x^*) = 0$. Тогда x^* - точка глобального минимума функции $f(x)$

Условия второго порядка:

- **Теорема. (необходимое условие)** Пусть x^* точка минимума функции $f(x)$, дважды дифференцируема в точке x^* . Тогда гессиан $\nabla^2 f(x^*)$ неотрицательно определен: $\forall h \langle \nabla^2 f(x^*) h, h \rangle \geq 0$
- **Теорема. (достаточное условие)** Пусть $f(x)$ дважды дифференцируема в точке x^* , $\nabla f(x^*) = 0$ и $\nabla^2 f(x^*)$ неотрицательно определен. Тогда x^* - точка локального минимума.

Задача безусловной оптимизации

Чтобы найти $x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} f(x)$ необходимо решить $\nabla f(x) = 0$

Если $f(x)$ выпуклая \Rightarrow единственный минимум

Если $f(x)$ невыпуклая:

- решения может не быть (функция вогнута, не имеет минимума)
- решений может быть много \Rightarrow необходимо определить, является ли оно экстремумом, если да, то максимум или минимум.

Если нет аналитического решения \Rightarrow итерационная процедура, определяющее решение приближенно: $|f(x) - f(x^*)| \leq \varepsilon$

Поиск приближенного решения

Метод решения задачи оптимизации – построение приближения к решению исходной задачи (точке минимума x^*) на основе информации о характеристиках целевой функции.

Методы нулевого порядка используют только значения функции

Методы первого порядка – первые производные

Методы второго порядка – вторые производные

Общий вид метода оптимизации

Многие методы оптимизации имеют вид

$$x^{k+1} = x^k + \eta_k d^k, d^k \in \mathbb{R}^n, \eta_k \in \mathbb{R}, k = 0, 1 \dots$$

x^0 - начальное приближение

d^k – направление минимизации, определяется характеристиками минимизируемой функции и выбранной процедуры

η_k – длина шага (в МО называют скорость обучения)

- Если точное решение находится за конечное число шагов, метод называется *конечношаговым*, иначе – *бесконечношаговым*
- Бесконечношаговый сходится, если $x^k \rightarrow x^*, k \rightarrow \infty$
- Скорость сходимости (число шагов, не количество вычислений):
 - Линейная $\|x^{k+1} - x^*\| \leq q \|x^k - x^*\|, q \in (0, 1)$
 - Сверхлинейная $\|x^{k+1} - x^*\| \leq q_k \|x^k - x^*\|, q_k \rightarrow 0$
 - Квадратичная $\|x^{k+1} - x^*\| \leq q \|x^k - x^*\|^2$
 - и т.д.

Методы спуска

Определение. Вектор $d \in \mathbb{R}^n$ называется направлением убывания функции $f: \mathbb{R}^n \rightarrow \mathbb{R}$ в точке $x \in \mathbb{R}^n$, если для малого η :

$$f(x + \eta d) < f(x)$$

$D_f(x)$ – множество (конус) всех направлений убывания в точке (все вектора, чье скалярное произведение с антиградиентом положительно)

$$\forall d \in D_f(x), \langle \nabla f(x), d \rangle \leq 0$$

Методы спуска:

$$x^{k+1} = x^k + \eta_k d^k, d \in D_f(x^k), k = 0, 1, \dots,$$

$$\eta_k: \{f(x^k)\} \text{ убывает}$$

Каждый метод – свой подход к выбору **шага** и выбору **направления** убывания

Критерии остановки: число шагов, малое изменение целевой функции, близость нормы градиента к 0

Градиентный метод

Направление спуска противоположно градиенту: $d^k = -\nabla f(x)$

- Выбираем начальное приближение $x^0 \in \mathbb{R}^n$
- Каждое следующее приближение определяется по правилу:

$$x^{k+1} = x^k - \eta_k \nabla f(x^k), k = 0, 1, \dots$$

Правила выбора шага:

- априорное задание $\{\eta_k\}_{k=0}^{\infty}$ в виде константы или правил пересчета, например:

$$\eta_k = \eta, \eta_k = \frac{\eta}{1 + \beta k}$$

- полная релаксация - *метод наискорейшего спуска*

$$\eta_k = \arg \min_{\eta \geq 0} (x^k - \eta \nabla f(x^k))$$

- правило Армихо, Вульфа и другие



Правило Армихо (линейный поиск)

Фиксируем параметры: дробления θ (например, пополам), допуск $\varepsilon \in (0,1)$, начальный шаг $\eta_0 > 0$

Путем дробления $\eta := \eta\theta$ добиваемся выполнения неравенства

$$f(x^k + \eta d^k) \leq f(x^k) + \varepsilon \eta \langle f(x^k), d^k \rangle$$

Как только неравенство выполнено, полагаем $\eta_k := \eta$

Полезные свойства, доказанные для случая выбора шага по правилу Армихо:

- Сходимость к стационарной точке (минимуму если выпуклая функция), если градиент удовлетворяет условию Липшица

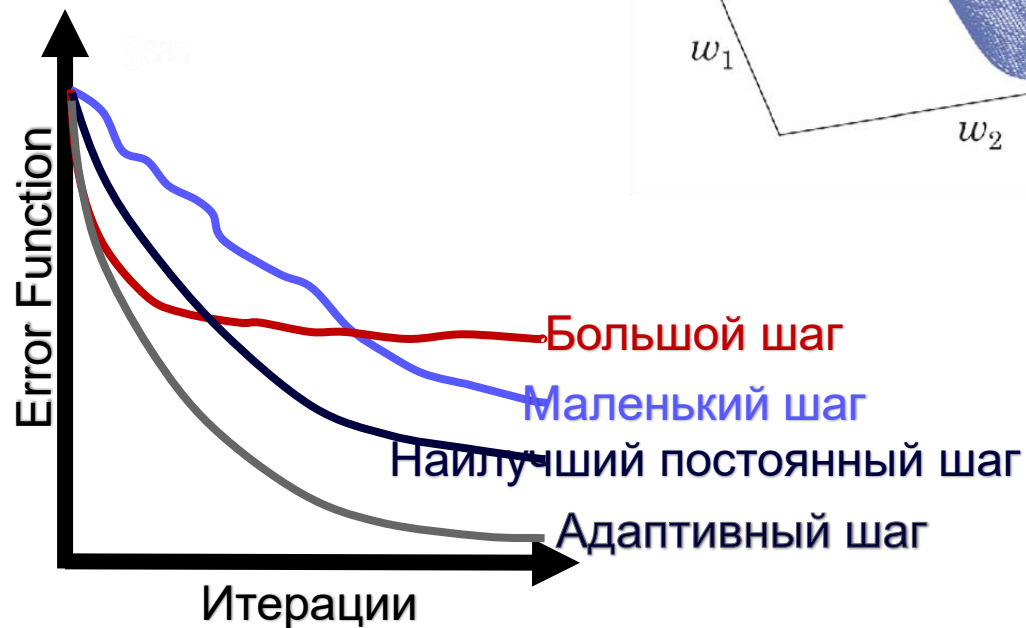
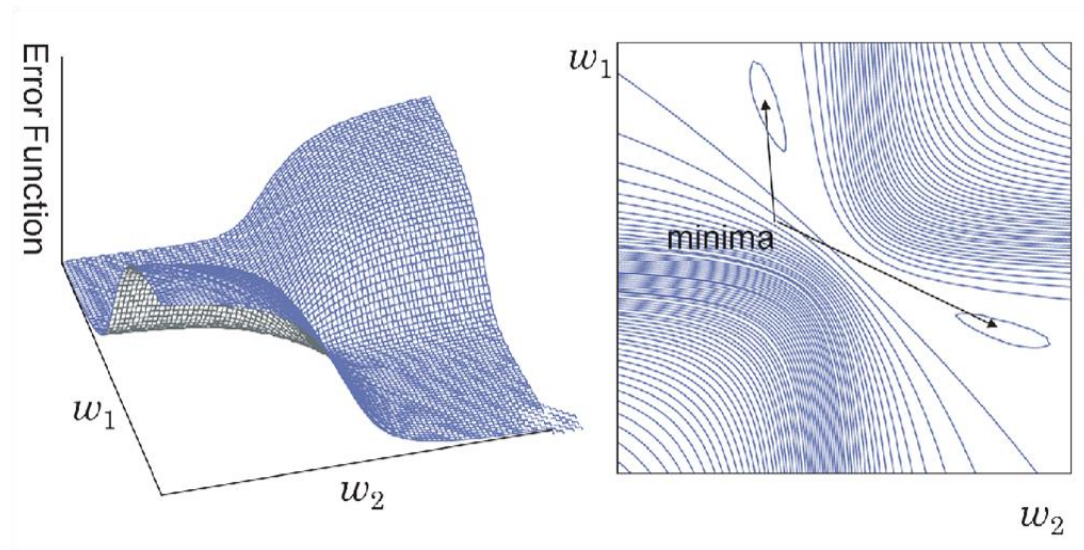
$$\forall x, y: \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

- Линейная скорость сходимости, если функция дважды дифференцируема и выполнено ограничение на гессиан:

$$\forall x, h: \langle \nabla^2 f(x)h, h \rangle \leq D\|h\|^2$$

Градиентный метод: примеры

«Поверхность ошибки» и
контурная проекция -
невыпуклая



Зависимость от выбора
шага для невыпуклой
функции

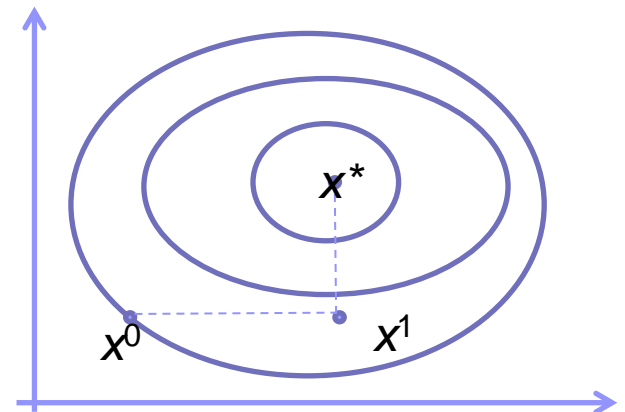
Покоординатный спуск

Суть метода:

- спуск, но по одной или нескольким координатам, их компоненты в d^k ненулевые, остальные равны 0
- Выбор шага обычно – наискорейший спуск
- Популярный частный случай – alternating optimization, исходя из условий $\nabla f(x^*) = 0$ выражаются и последовательно вычисляются $x_i^k = g(x_1^{k-1}, \dots, x_{i-1}^{k-1}, x_{i+1}^{k-1}, \dots, x_n^{k-1})$
- Метод ведет себя разумно только для гладких функций

Выбор ненулевых направлений:

- Последовательный
- Блочный
- Случайный



Метод Ньютона

Пусть $f(x)$ – выпуклая дважды дифференцируемая функция

По определению дважды дифференцируемой функции (разложим в ряд Тейлора)

$$f(x) - f(x^k) = \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2} \langle \nabla^2 f(x^k)(x - x^k), x - x^k \rangle + o(\|x - x^k\|^2)$$

Минимизируем квадратичную часть:

$$\langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2} \langle \nabla^2 f(x^k)(x - x^k), x - x^k \rangle \rightarrow \min$$

Она выпукла, так как ее гессиан $\nabla^2 f(x^k) \geq 0$

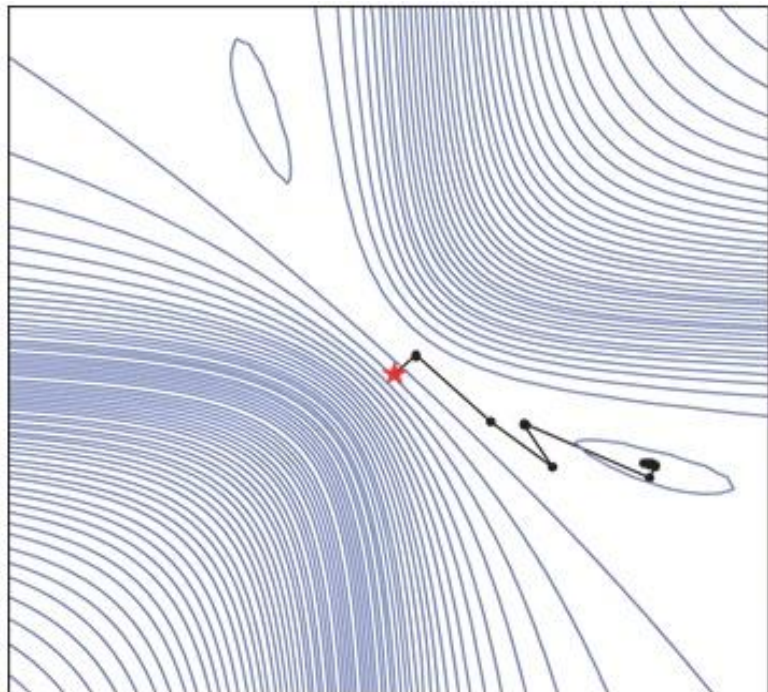
Необходимое и достаточное условие минимума:

$$\nabla f(x^k) + \nabla^2 f(x^k)(x - x^k) = 0$$

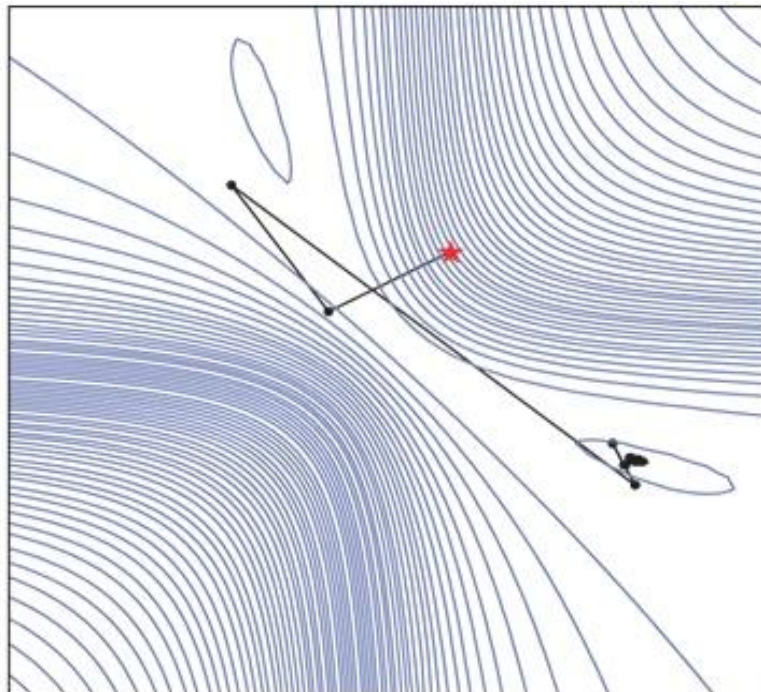
Отсюда получаем метод Ньютона:

$$x^{k+1} = x^k + h^k, h^k = -(\nabla^2 f(x^k))^{-1} \nabla f(x^k)$$

Пример



38 итераций



57 итераций

Не нужно задавать длину шага, мало итераций, но много вычислений на каждой – нужно считать матрицу вторых производных и обращать ее

Особенности метода Ньютона

- Нет проблемы **выбора шага**
- **Сходимость:**
 - Если целевая функция дважды дифференцируема, сильно выпуклая, тогда получаемая в методе Ньютона последовательность приближений сходится к точке минимума с квадратичной скоростью.
 - Для невыпуклых функций сходимость гарантирована только при условии близости начальной точки к точке минимума.
- Необходимость выбора **начального приближения:**
 - в окрестности решения, условие близости трудно проверить
- **Вычислительная** трудоемкость, а иногда и нестабильность:
 - нужно вычислять на каждом шаге матрицу вторых производных и затем обратную к ней.
- Желание сохранить быструю скорость сходимости и устранить недостатки привели к разработке множества **модификаций** метода Ньютона и **квазиньютоновских** методов

Простые модификации метода Ньютона

Метод Ньютона с регулировкой шага:

$$x^{k+1} = x^k + \eta_k h^k, \eta_k > 0, h^k = -(\nabla^2 f(x^k))^{-1} \nabla f(x^k)$$

- Шаг выбирается либо из условия минимизации функции вдоль заданного направления (наискорейший спуск), либо по правилу дробления шага (линейный поиск).
- Скорость сходимости сверхлинейная или квадратичная
- Сократить вычисления можно, пересчитывая гессиан один раз в несколько шагов. Шаг пересчета подбирается эмпирически.

Быстрое обратное распространение ошибки:

- Приближаем целевую функцию «параболой», вычисляем диагональ Гессиана «приближенной» функции:

$$x^{k+1} = x^k - [\text{diag}(\tilde{H})]^{-1} \nabla f(x^k), H \approx \nabla^2 f(x^k)$$

Регуляризация:

$$x^{k+1} = x^k - [\nabla^2 f(x^k) + \gamma I]^{-1} \nabla f(x^k)$$

Общая схема квазиньютоновских методов

Суть подхода:

$$x^{k+1} = x^k + \eta_k d^k, d^k = -H_k \nabla f(x^k), k = 0, 1, \dots$$

$H_k \in \mathbb{R}(n, n)$ - некоторая симметричная положительно определенная матрица, тогда d^k - направление спуска, поскольку $\langle \nabla f(x^k), d^k \rangle = -\langle H_k \nabla f(x^k), \nabla f(x^k) \rangle < 0$

Если $Q_k \equiv I$ - градиентный метод, если $H_k = (\nabla^2 f(x^k))^{-1}$ и $\eta_k = 1$ - метод Ньютона

Квазиньютоновские методы аппроксимируют:

$$H_k \approx [\nabla^2 f(x^k)]^{-1}$$

Например, метод Бroyдена-Флетчера-Голдфарба-Шэнно (БФГШ) (Broyden-Fletcher-Goldfarb-Shanno):

$$H_{k+1} = H_k + \frac{(r^k - H_k s^k)(r^k)^T + r^k (r^k - H_k s^k)^T}{\langle r^k, s^k \rangle} - \frac{\langle r^k - H_k s^k, s^k \rangle r^k (r^k)^T}{\langle r^k, s^k \rangle^2}$$
$$r^k = x^{k+1} - x^k, s^k = \nabla f(x^{k+1}) - \nabla f(x^k)$$

Задача математического программирования (условная оптимизация)

ЗМП:

$$\begin{aligned} \min f(x) \\ g_i(x) \leq 0, i = \overline{1, k} = E \\ g_i(x) = 0, i = \overline{k+1, m} = I \\ x \in P \end{aligned}$$

Допустимое множество $X = \{x \in P | g_i(x) \leq 0, i \in I; g_i(x) = 0, i \in E\}$

- *линейное программирование* – целевая функция и ограничения линейны
- *квадратичное программирование* – квадратичная целевая функция и линейные ограничения
- *выпуклое программирование* – ограничения и целевая функции выпуклые, P – выпуклое множество

Принцип Лагранжа

Функция Лагранжа: $L(x, \lambda) = f(x) + \langle \lambda, g(x) \rangle$,

множители Лагранжа $\lambda = (\lambda_1, \dots, \lambda_m)$

$$L: \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}, f, g_i(x): \mathbb{R}^n \rightarrow \mathbb{R},$$

$$\nabla L(x, \lambda) = \nabla f(x) + \sum_{i=1}^m \lambda_i \nabla g_i(x)$$

Рассмотрим случай отсутствия ограничений-неравенств

Принцип Лагранжа (необходимое условие оптимальности):

- Пусть f дифференцируема в точке $x^* \in \mathbb{R}^n$, $g_i, i = \overline{1, m}$ непрерывно дифференцируемы в некоторой окрестности x^* и в точке x^* выполнено дополнительное условие регулярности.
- Если x^* является локальным решением задачи (ЗМП), то существует $\lambda^* \in \mathbb{R}^m: \nabla L(x^*, \lambda^*) = 0$

Система Лагранжа:

$$\begin{cases} \nabla L(x, \lambda) = 0 \\ g(x) = 0 \end{cases}$$

$\bar{x} \in X: \nabla L(\bar{x}, \bar{\lambda}) = 0$ для некоторого $\bar{\lambda}$ - стационарная точка (ЗМП)

Условия Каруша-Куна-Таккера

Общий случай (смешанные ограничения) $0 \leq k \leq m$

Теорема. Каруша–Куна–Таккера (ККТ) (необходимое условие оптимальности).

Пусть $f, g_i, i = \overline{1, k}$, дифференцируемы в точке x^* , $g_i, i = \overline{k+1, m}$ непрерывно дифференцируемы в некоторой окрестности x^* и в точке x^* выполнено условие регулярности.

Если x^* является локальным решением задачи (ЗМП), то существует λ^* :

$$\nabla_x L(x^*, \lambda^*) = 0 \text{ - стационарность}$$

$$\lambda_i^* g_i(x^*) = 0 \text{ - дополняющая нежёсткость}$$

$$g_i(x^*) = 0, i = \overline{1, k}, g_i(x^*) \leq 0, i = \overline{k+1, m}$$

$$\lambda_i^* \geq 0, \text{ - неотрицательность (знакоопределенность)}$$

ДВОЙСТВЕННОСТЬ

Прямая задача

$$f(x) \rightarrow \min$$

$$g_i(x) \leq 0, i = \overline{1, k}$$

$$g_i(x) = 0, i = \overline{k+1, m}$$

$$x \in P \subseteq \mathbb{R}^n$$

$$f^* = \inf_{x \in X} f(x)$$

Двойственная задача

$$\varphi(\lambda) \rightarrow \max$$

$$\varphi(\lambda) = \inf_{x \in P} L(x, \lambda) = \inf_{x \in P} (f(x) + \langle \lambda, g(x) \rangle)$$

$$\lambda \in Y = \{\lambda \in Q \mid \varphi(\lambda) > -\infty\}$$

$$Q = \{\lambda \in \mathbb{R}^m \mid \lambda_i \geq 0, i = \overline{1, k}\}$$

$$\varphi^* = \sup_{\lambda \in Y} \varphi(\lambda)$$

Прямая задача представима в виде, где $\psi(x) = \sup_{\lambda \in Q} L(x, \lambda)$:

$$\min \psi(x)$$

$$x \in P$$

Значения экстремумов полагаются бесконечными вне допустимого множества.

Прямая и двойственная задачи определяются симметрично относительно функции Лагранжа (седловая точка):

$$f^* = \inf_{x \in P} \sup_{\lambda \in Q} L(x, \lambda) \text{ и } \varphi^* = \sup_{\lambda \in Q} \inf_{x \in P} L(x, \lambda)$$

Теорема двойственности и седловые точки

Теорема двойственности.

Пусть в (ЗМП) существует вектор ККТ

$$\lambda^* \in Q: f^* \leq f(x) + \langle \lambda^*, g(x) \rangle = L(x, \lambda^*) \quad \forall x \in P$$

Если $f^* > -\infty$, то множество решений двойственной задачи не пусто и совпадает с множеством векторов ККТ прямой задачи.

При этом $f^* = \varphi^*$.

Пара $(x^*, \lambda^*) \in P \times Q$ **седловая точка** $L(x, \lambda)$ на $P \times Q$, если

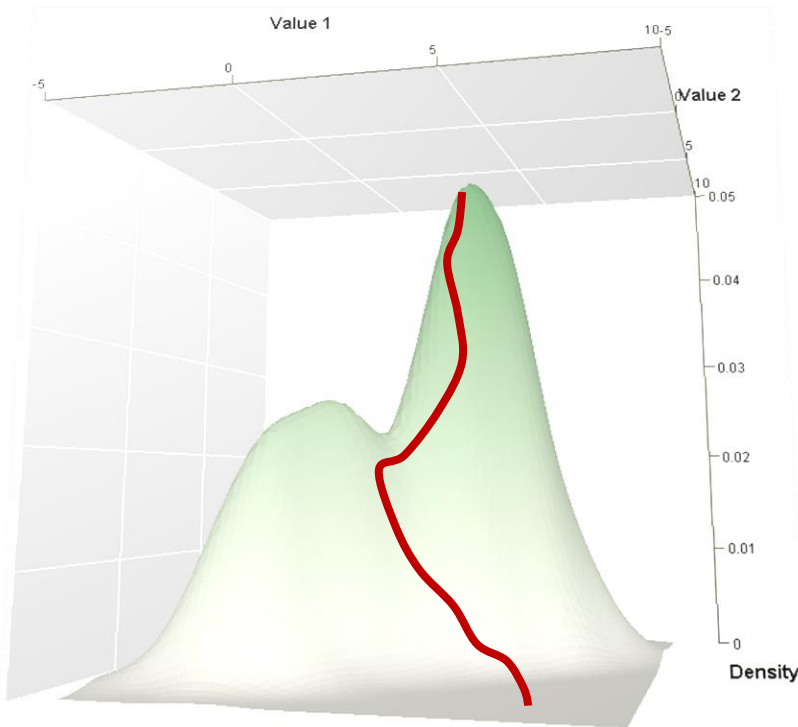
$$L(x^*, \lambda^*) = \min_{x \in P} L(x, \lambda^*) = \max_{\lambda \in Q} L(x^*, \lambda),$$

т.е. иначе: $\forall x \in P, \lambda \in Q: L(x, \lambda^*) \geq L(x^*, \lambda^*) \geq L(x^*, \lambda)$

Теорема ККТ.

Пусть в (ЗМП) существует вектор ККТ. Точка $x^* \in P$ является решением (ЗМП) тогда и только тогда, когда существует седловая точка $\lambda^* \in Q: (x^*, \lambda^*)$ функции Лагранжа $L(x, \lambda)$ на $P \times Q$

Классические методы оптимизации для задачи машинного обучения



- Целевая функция – эмпирический риск, усредненная сумма значений функций риска для всех наблюдений
- Используют все наблюдения выборки для расчета эмпирического риска, его градиента или приближения гессиана
- Результат – гладкая траектория оптимизации
- Но долго по времени и много по памяти

Градиентный спуск для задачи машинного обучения

- Дана «размеченная» выборка :

$$Z = \{(x_i, y_i)\}_{i=1}^l \in X \times Y$$

- Фиксируем параметрическое семейство алгоритмов (моделей)

$$A = \{g(x, w) \mid w \in W\}, g: X \times W \rightarrow Y$$

- Минимизируем:

$$Q(w, Z) = \frac{1}{l} \sum_Z L(y_i, g(x_i, w)) \rightarrow \min$$

- Градиентный спуск:

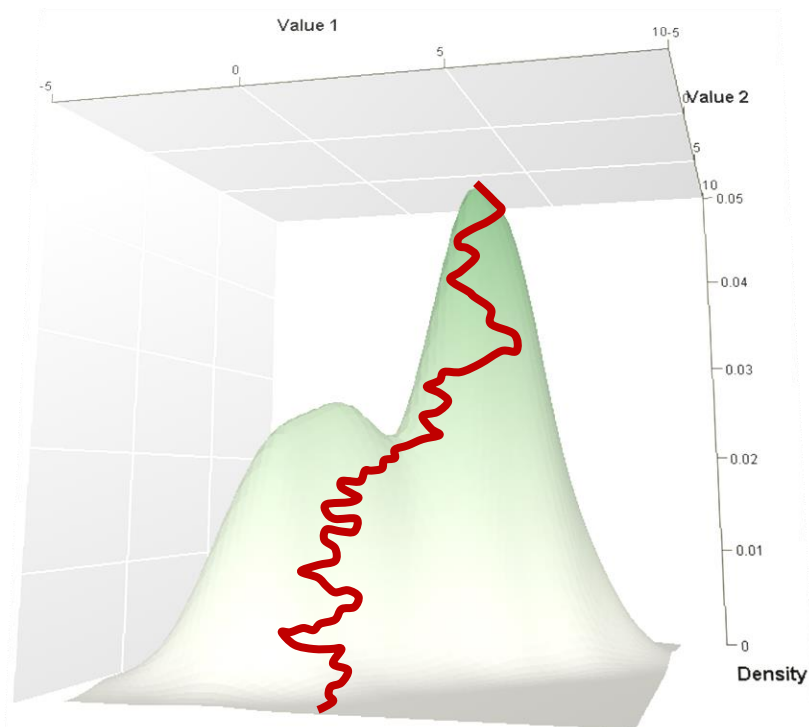
- ☐ Выбираем начальное приближение $w^{(0)}$

- ☐ В цикле до достижения условий сходимости считаем

$$w^{(t+1)} = w^{(t)} - \eta \nabla Q(w^{(t)}) = w^{(t)} - \eta \frac{1}{l} \sum_Z \nabla L(y_i, g(x_i, w^{(t)}))$$

- А что если считать ∇Q не по всей выборке?

Стохастические и пакетные методы обучения



- Аппроксимируют вычисление градиента эмпирического риска (или приближение гессiana) только по части выборки
- Стохастические методы используют только одно наблюдение
- Пакетные – часть наблюдений
- Результат – «хаотическая» траектория (чем больше пакет тем меньше «хаоса»)
- Зато быстро и не нужно все брать из памяти – MPP!!!

Стохастический градиентный спуск для задачи машинного обучения

- Параметры: η – скорость обучения, λ – скорость забывания
- Инициализация $w^{(0)}$ и $Q^{(0)}$ -средний по случайному подмножеству Z
- Цикл стохастического градиентного спуска
 - Выбираем x_i из Z
 - Вычисляем функцию потерь на нем $\varepsilon_i = L(y_i, g(x_i, w^{(t)}))$
 - Делаем шаг градиента $w^{(t+1)} = w^{(t)} - \eta \nabla L(y_i, g(x_i, w^{(t)}))$
 - Оцениваем $Q^{(t)} = \lambda \varepsilon_i + (1 - \lambda) Q^{(t-1)}$
- Приближение $Q^{(t)}$:
 - Среднее арифметическое $Q^{(t)} = \frac{1}{t} \varepsilon_t + \frac{1}{t} \varepsilon_{t-1} \dots$ или $Q^{(t)} = \frac{1}{t} \varepsilon_t + \left(1 - \frac{1}{t}\right) Q^{(t-1)}$
 - Экспоненциальное сглаживание $Q^{(t)} = \lambda \varepsilon_t + (1 - \lambda) Q^{(t-1)}$, λ порядка $\frac{1}{t}$
- Выбор объектов для обучения:
 - Случайный или на основе ошибки – чем хуже тем лучше

Важные модификации градиентного метода

- Учет инерции («метод моментов», или импульса) – «сгладить» траекторию за счет предыдущих направлений (импульса), α задает «важность» старого направления
- Метод Нестерова – «сглаживать» после применения старого шага:

$$w^{k+1} = w^k + v^k, v^k = -(1 - \alpha)\eta_k \nabla Q(w^k) + \alpha v^{k-1}$$

$$v^k = -(1 - \alpha)\eta_k \nabla Q(w^k + \alpha v^{k-1})$$

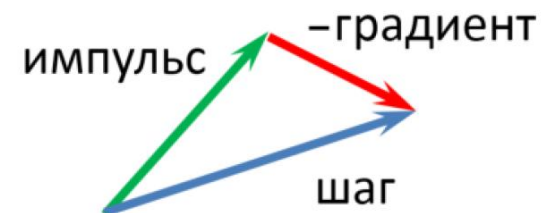
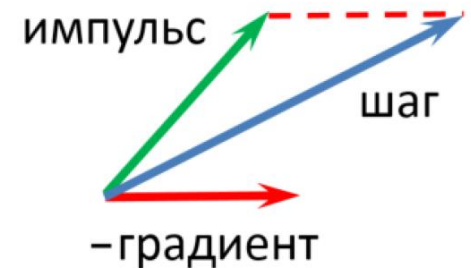
- Регуляризация (штраф за сложность):

$$\min Q(w) + \gamma R(w)$$

$R(\cdot)$ – гладкая сильно выпуклая функция, например, регуляризация L_p

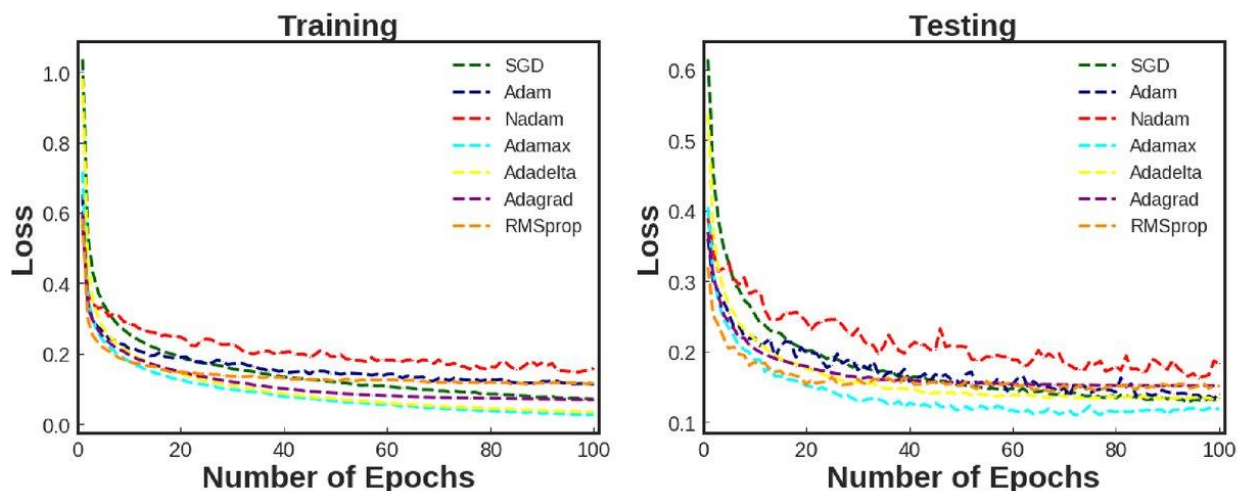
γ -константа регуляризации

$$v^k = -\eta_k [\nabla Q(w^k) + \gamma \nabla R(w^k)]$$



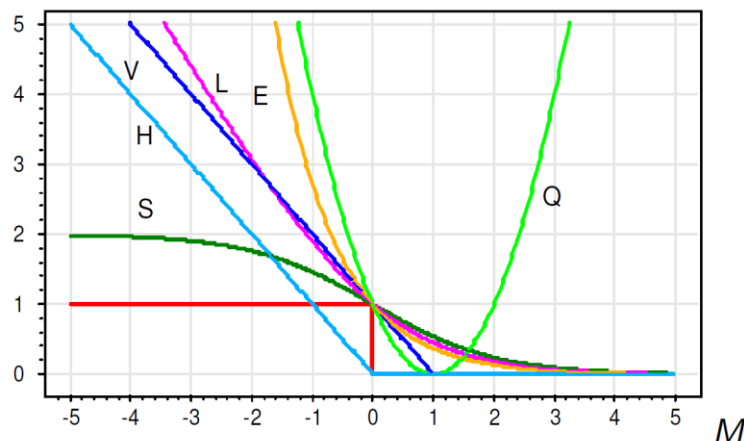
Другие популярные модификации SGD

- RProp – устойчивый к угасанию градиента (в многослойных нейросетях)
- Adam – экспоненциальное сглаживание градиентов
- AdaGrad и RMSProp – индивидуальные скорости обучения для каждого параметра «по ситуации»
- и их комбинации ...



Пример простых линейных моделей

- Линейная модель: $a(x, w) = \langle x, w \rangle$
- Регрессия ($Y = \mathbb{R}$): $L(a(x_i, w), y_i) = (\langle x_i, w \rangle - y_i)^2$
- Классификация ($Y = \{-1, 1\}$): «отступ» $M(w) = \langle x_i, w \rangle, y_i$
 $L(a(x_i, w), y_i) = F(M(w))$



$$\begin{aligned}
 V(M) &= (1 - M)_+ \\
 H(M) &= (-M)_+ \\
 L(M) &= \log_2(1 + e^{-M}) \\
 Q(M) &= (1 - M)^2 \\
 S(M) &= 2(1 + e^M)^{-1} \\
 E(M) &= e^{-M} \\
 [M < 0]
 \end{aligned}$$

- кусочно-линейная (SVM);
- кусочно-линейная (Hebb's rule);
- логарифмическая (LR);
- квадратичная (FLD);
- сигмоидная (ANN);
- экспоненциальная (AdaBoost);
- пороговая функция потерь.

- Эмпирический риск: $Q(w, Z) = \frac{1}{l} \sum_i L(a(x_i, w), y_i)$
- Регуляризация: $L_2: R(w) = \sum w^2$ или $L_1: R(w) = \sum |w|$

Пример классификации

```
import pandas as pd
import numpy as np

from sklearn.metrics import hinge_loss, log_loss

import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

```
def get_iter(size):
    np.random.seed(0)
    for i in range(50):
        yield np.random.randint(size)
```

```
def plot_sgd_decision(model, X):
    X1_max, X2_max = X.max(axis=0)
    X1_min, X2_min = X.min(axis=0)
    xx1 = np.linspace(X1_min - 0.5, X1_max + 0.5, 20)
    xx2 = np.linspace(X2_min - 0.5, X2_max + 0.5, 20)

    X1, X2 = np.meshgrid(xx1, xx2)
    Z = np.empty(X1.shape)
    for (i, j), val in np.ndenumerate(X1):
        x1 = val
        x2 = X2[i, j]
        p = model.decision_function([[x1, x2]])
        Z[i, j] = p[0]

    levels = [0.0]
    colors = "k"
    plt.contour(X1, X2, Z, levels, colors=colors)

def plot_trace(X, y, wei):
    w1_max, w2_max = wei.max(axis=0)
    w1_min, w2_min = wei.min(axis=0)
    w0 = np.linspace(w1_min - 0.5, w1_max + 0.5, 20)
    w1 = np.linspace(w2_min - 0.5, w2_max + 0.5, 20)

    X1, X2 = np.meshgrid(w0, w1)

    vals = np.zeros(shape=(w0.size, w1.size))

    for i, value1 in enumerate(w0):
        for j, value2 in enumerate(w1):
            w_temp = np.array([[value1, value2]])
            vals[i, j] = hinge_loss(y, np.dot(X, w_temp.T)[:0])

    cp = plt.contour(X1, X2, vals.T, colors='black', linestyle='dashed', linewidths=1)
    plt.clabel(cp, inline=1, fontsize=10)
    cp = plt.contourf(X1, X2, vals.T, alpha=0.7)
    plt.plot(wei[:, 0], wei[:, 1], linewidth=2.0, marker='.', color="red")
    plt.scatter(wei[[-1], 0], wei[[-1], 1], marker='*', color="red", s=100)
    plt.xlabel("w0")
    plt.ylabel("w1")
    plt.show()
```

Пример классификации

```
from sklearn.datasets import make_blobs, make_classification
# X, y = make_blobs(n_samples=200, centers=2, n_features=2, random_state=0)
X, y = make_classification(n_samples=200, n_classes=2, n_features=2, n_redundant=0, random_state=0)
```

```
from sklearn.linear_model import SGDClassifier
```

```
clf = SGDClassifier(loss="hinge", penalty="l1", alpha=0.5)
```

```
coefs = []
```

```
for i in get_iter(X.shape[0]):
```

```
    clf.partial_fit(X[[i]], y[[i]], classes=np.unique(y))
```

```
    coefs.append(clf.coef_.copy())
```

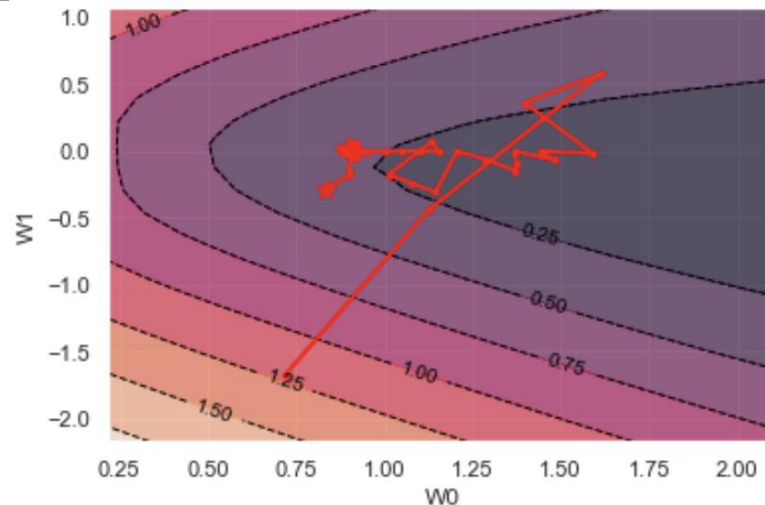
```
plot_sgd_decision(clf, X)
```

```
plt.scatter(X[:, 0], X[:, 1], c=np.where(y > 0, "blue", "orange"))
```

```
plt.scatter(X[[i], 0], X[[i], 1], c=np.where(y[[i]] > 0, "blue", "orange"), edgecolor="red", linewidth=3.0, s=200)
```

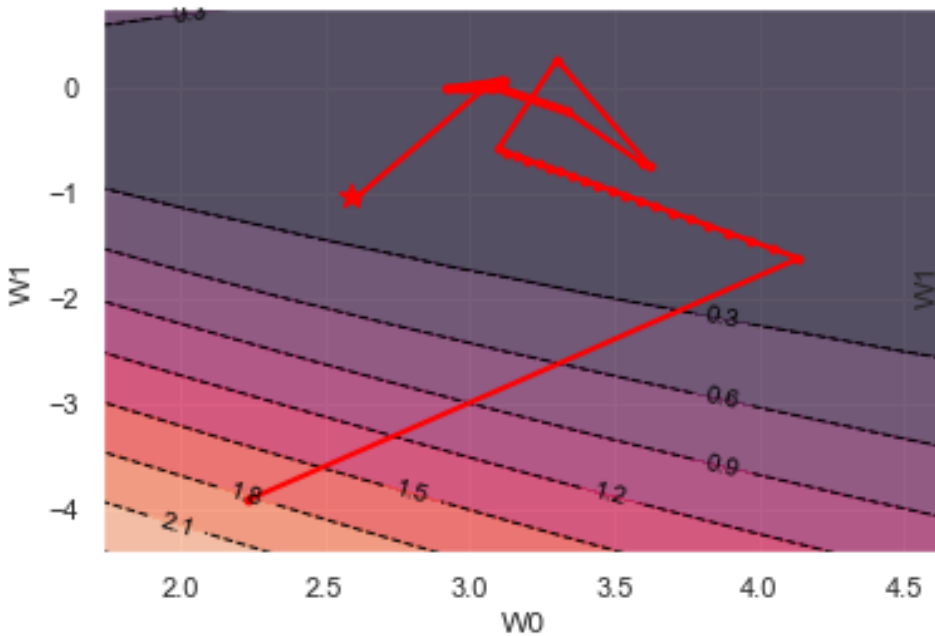
```
plt.show()
```

```
coefs = np.vstack(coefs)
plot_trace(X, y, coefs)
```

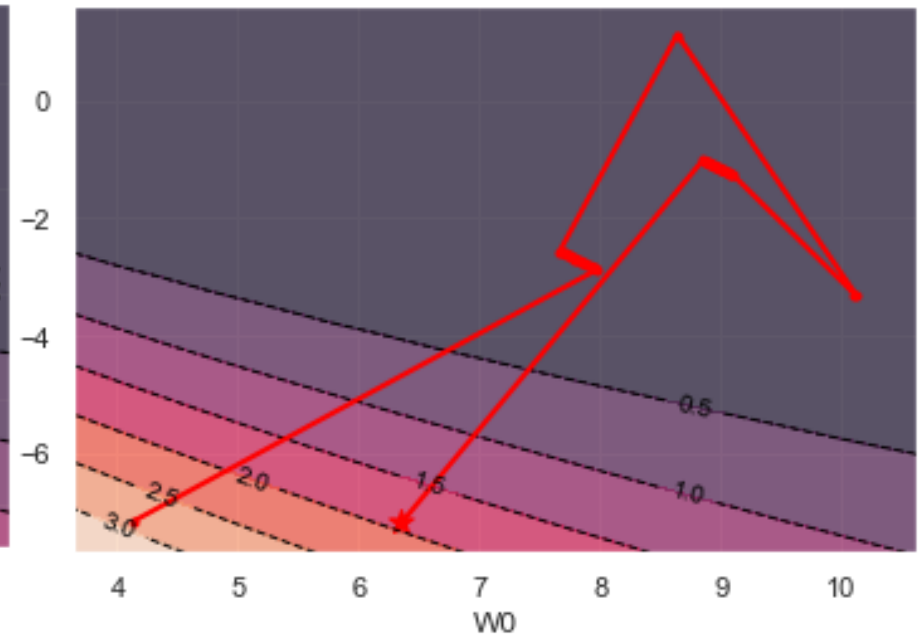


Пример классификации

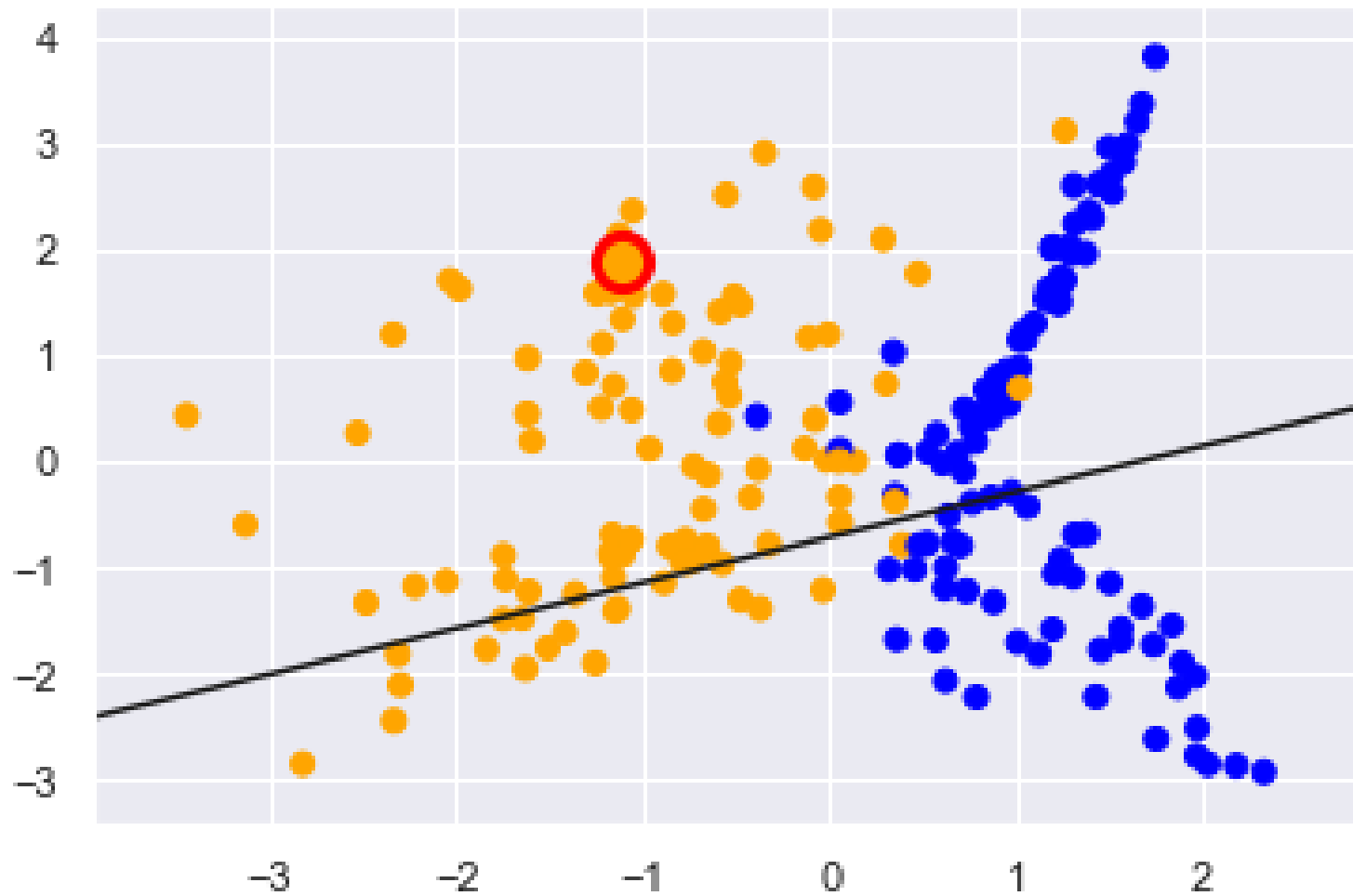
```
from sklearn.linear_model import SGDClassifier  
clf = SGDClassifier(loss="hinge", penalty="l1", alpha=0.05)
```



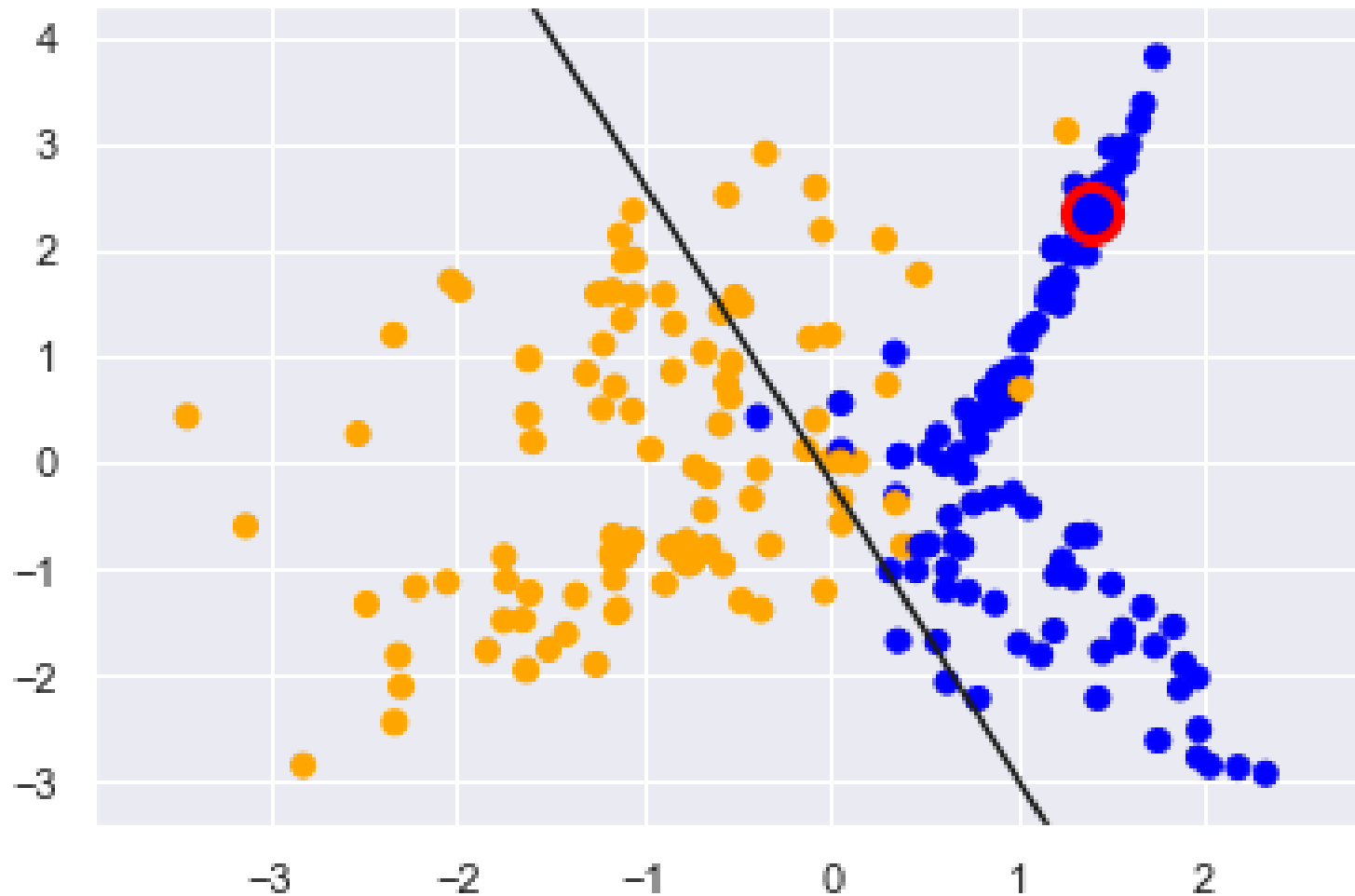
```
from sklearn.linear_model import SGDClassifier  
clf = SGDClassifier(loss="hinge", penalty="l1", alpha=0.005)
```



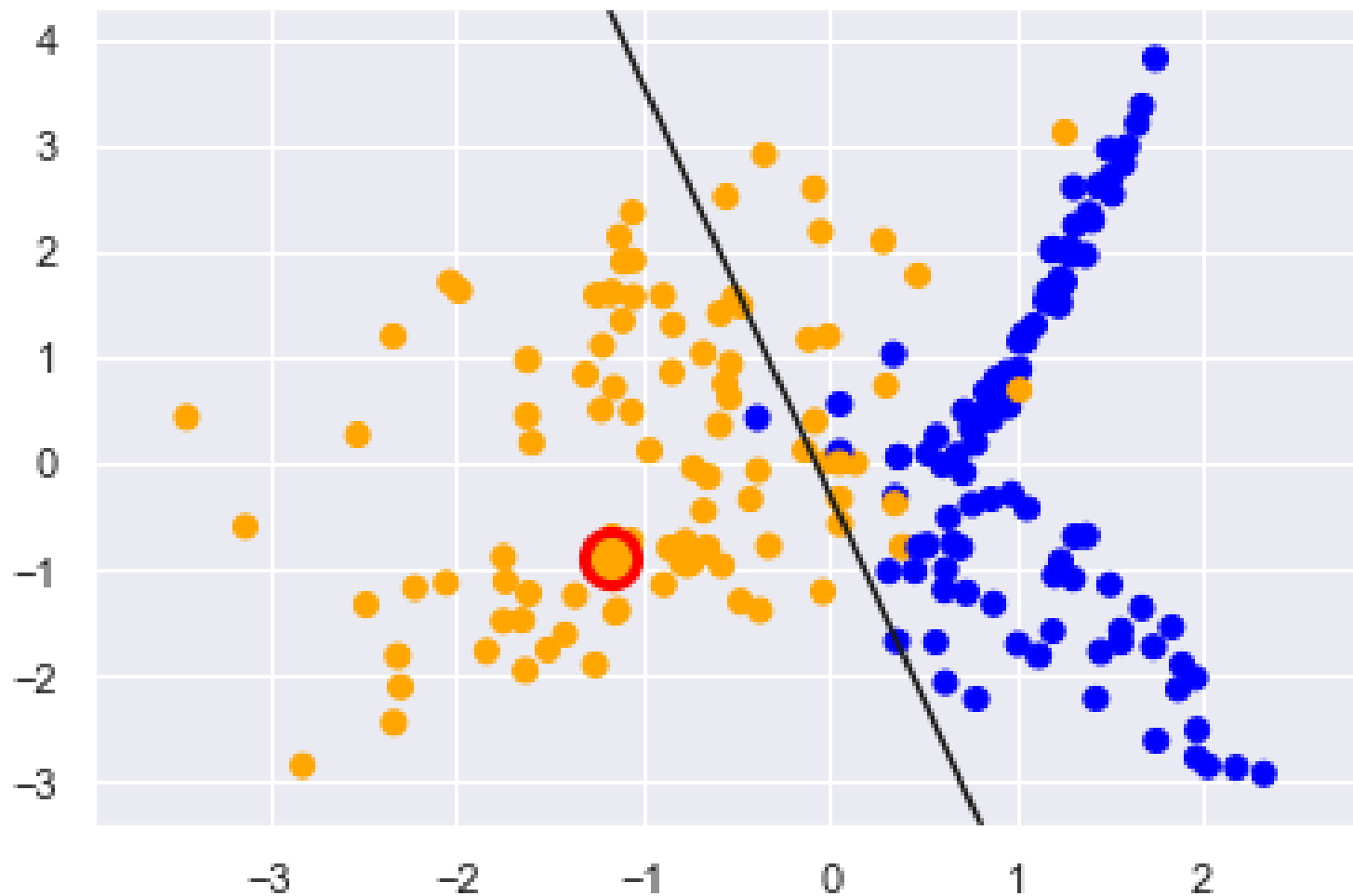
Пример классификации



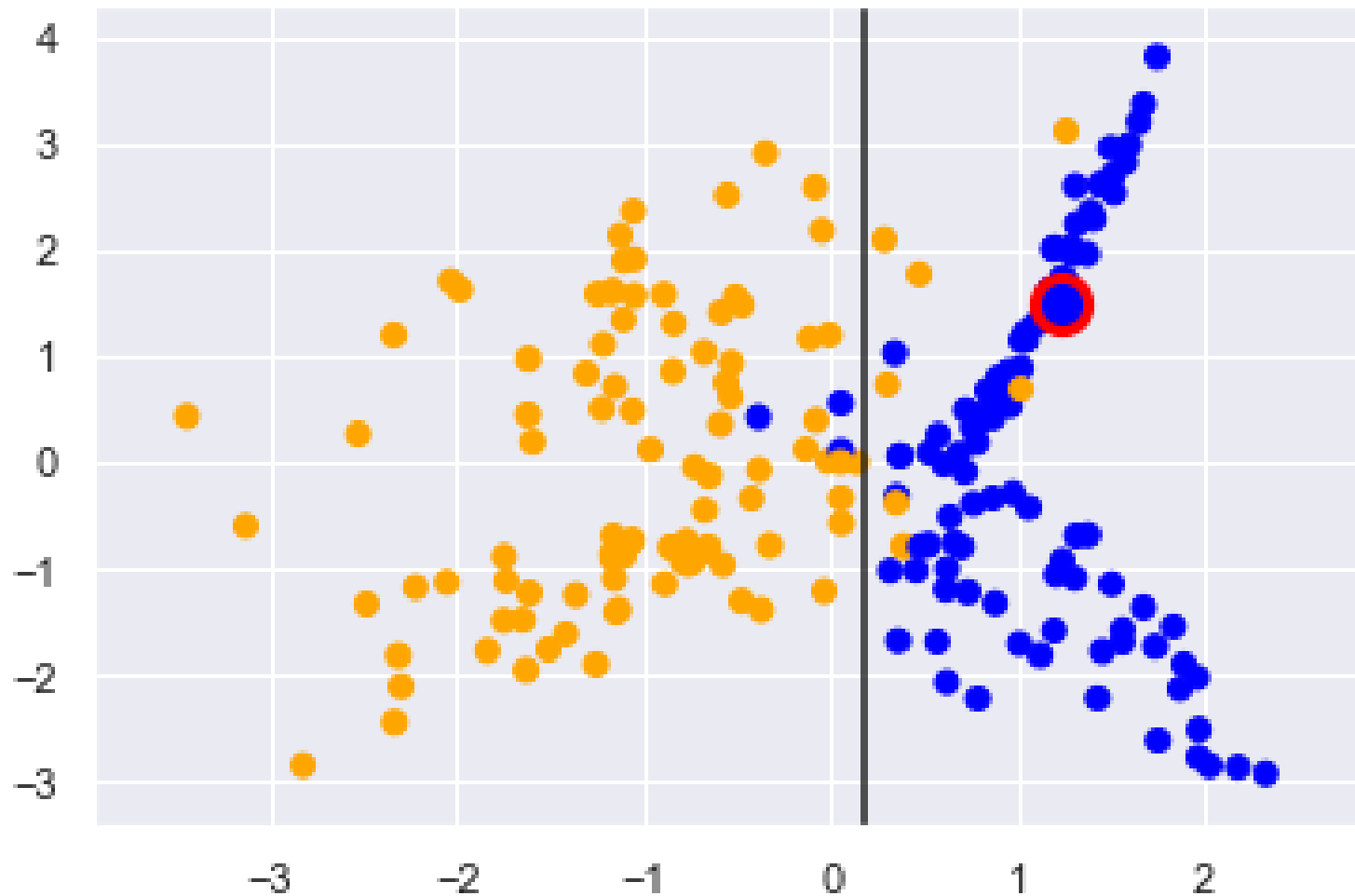
Пример классификации



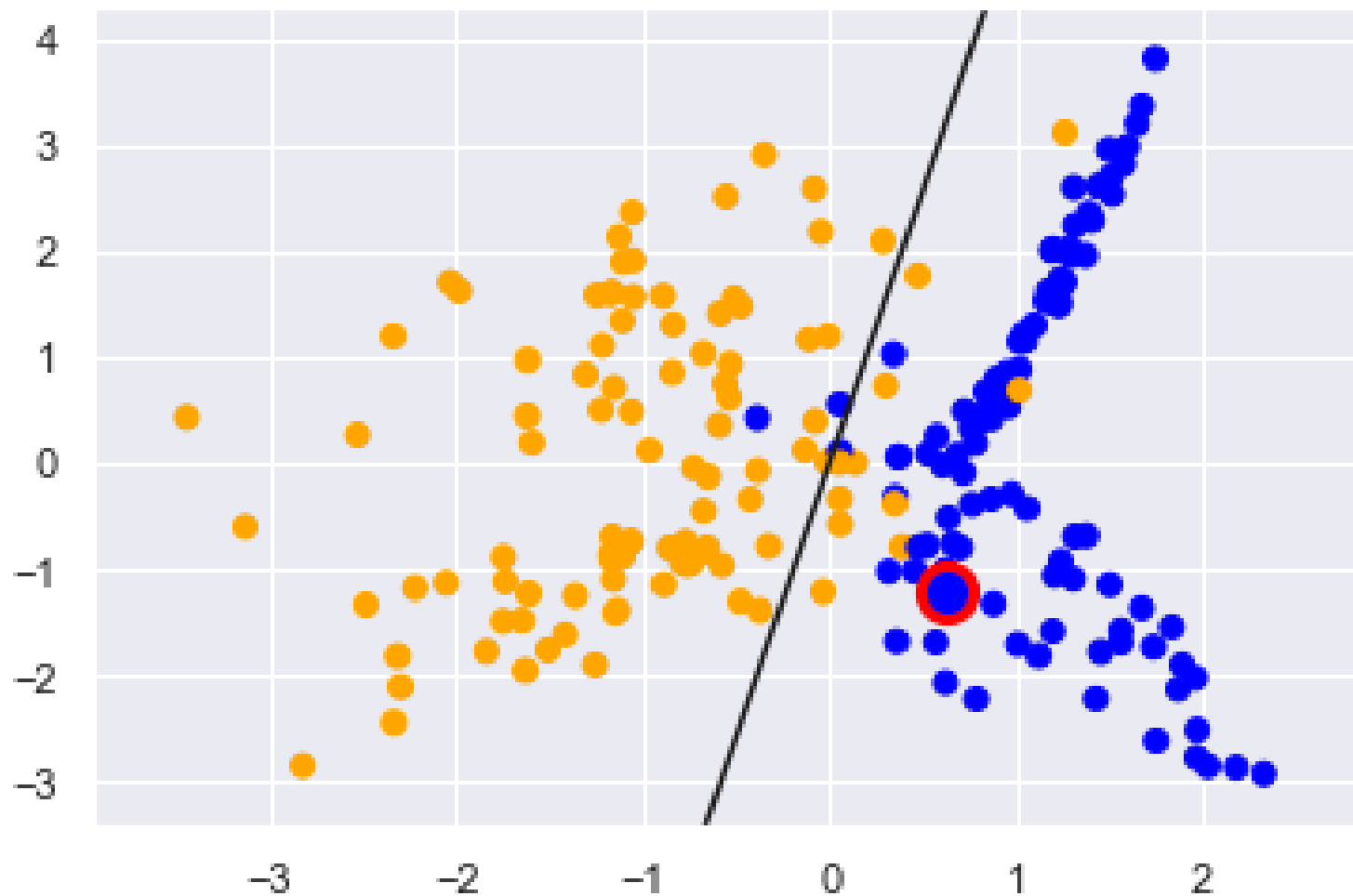
Пример классификации



Пример классификации



Пример классификации



Пример регрессии

```
from sklearn.metrics import mean_squared_error

def plot_sgd_predict(model, X):
    xx = np.linspace(X.min() - 0.5, X.max() + 0.5, 20)

    Z = np.empty(xx.shape)
    for i, val in np.ndenumerate(xx):
        Z[i] = model.predict([[val]])[0]

    plt.plot(xx, Z, color="black")

def plot_trace(X, y, wei):
    w1_max, w2_max = wei.max(axis=0)
    w1_min, w2_min = wei.min(axis=0)
    w0 = np.linspace(w1_min - 0.5, w1_max + 0.5, 20)
    w1 = np.linspace(w2_min - 0.5, w2_max + 0.5, 20)

    X1, X2 = np.meshgrid(w0, w1)

    vals = np.zeros(shape=(w0.size, w1.size))
    for i, value1 in enumerate(w0):
        for j, value2 in enumerate(w1):
            w_temp = np.array([value1])
            vals[i, j] = mean_squared_error(y, np.dot(X, w_temp) + value2)

    cp = plt.contour(X1, X2, vals.T, colors='black', linestyles='dashed', linewidths=1)
    plt.clabel(cp, inline=1, fontsize=10)
    cp = plt.contourf(X1, X2, vals.T, alpha=0.7)
    plt.plot(wei[:, 0], wei[:, 1], linewidth=2.0, marker='.', color="red")
    plt.scatter(wei[[-1], 0], wei[[-1], 1], marker='*', color="red", s=100)
    plt.xlabel("W0")
    plt.ylabel("Intercept")
    plt.show()
```

Пример регрессии

```
# https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\_regression.html#sklearn.datasets.make\_regression
```

```
from sklearn.datasets import make_regression, load_iris
X, y = make_regression(n_samples=100, n_features=1, n_informative=1, noise=10, random_state=0)
```

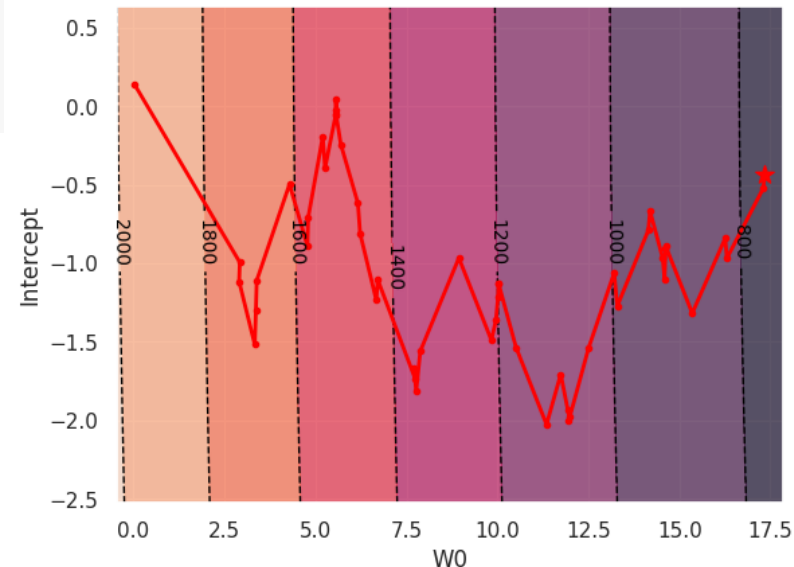
```
from sklearn.linear_model import SGDRegressor
```

```
clf = SGDRegressor(loss="squared_error", penalty="l1", alpha=0.0001, eta0=0.1)
```

```
coefs = []
for i in get_iter(X.shape[0]):
    clf.partial_fit(X[[i]], y[[i]])
    coefs.append([clf.coef_[0].copy(), clf.intercept_[0].copy()])
```

```
plot_sgd_predict(clf, X)
plt.scatter(X, y)
plt.scatter(X[[i]], y[[i]], color='C0', edgecolor="red", linewidth=3.0, s=200)
plt.show()
```

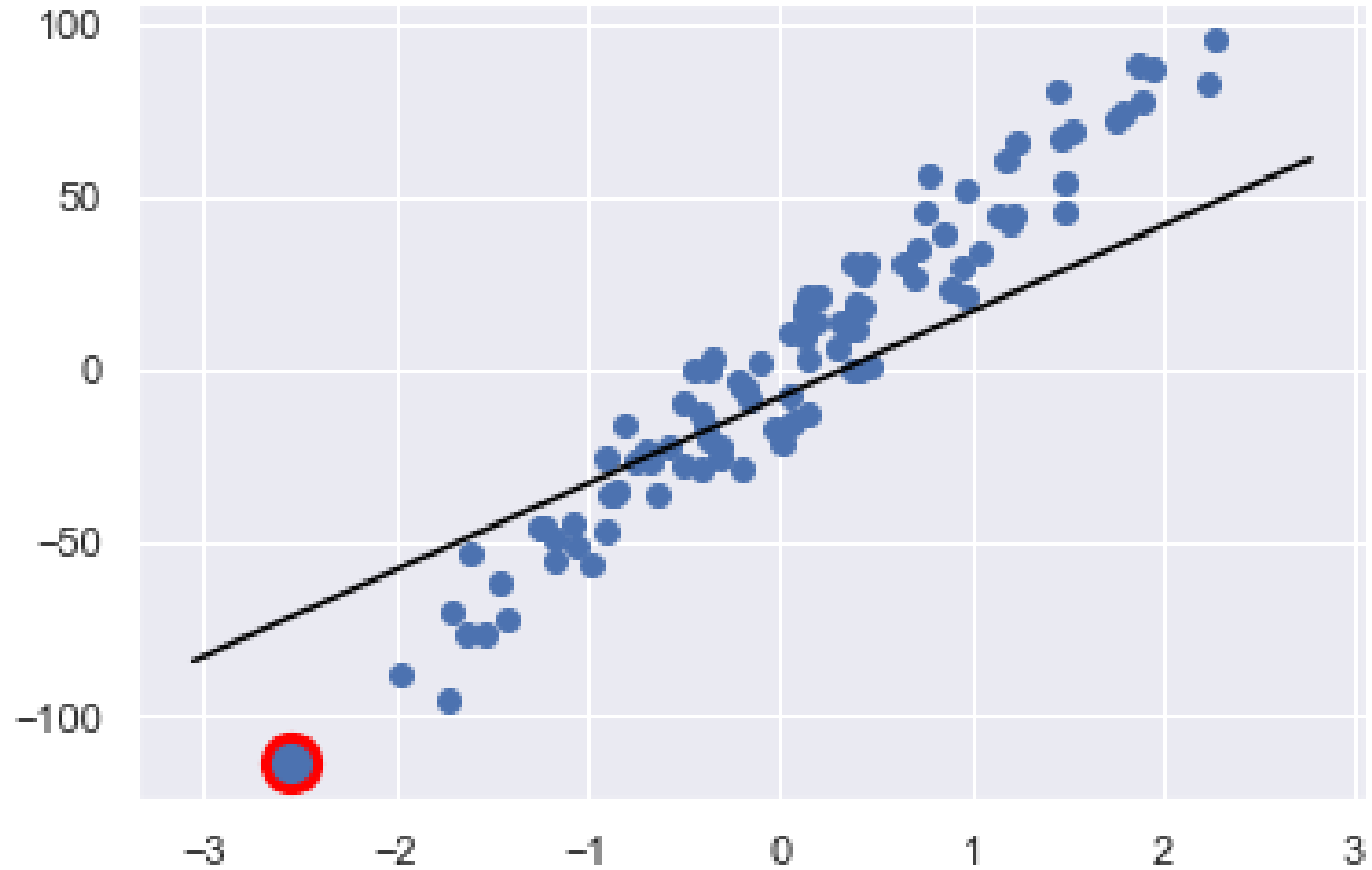
```
coefs = np.vstack(coefs)
plot_trace(X, y, coefs)
```



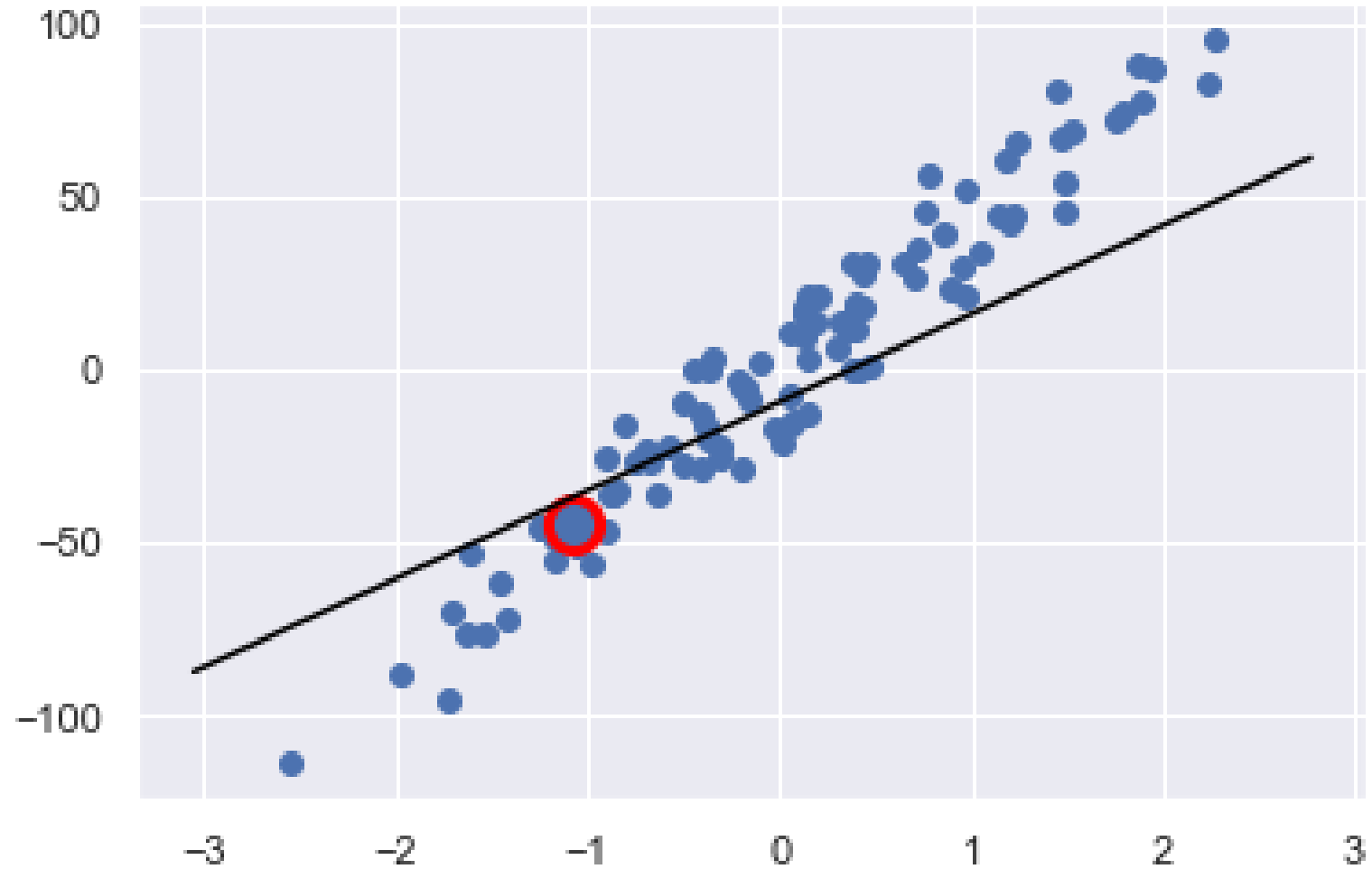
Пример регрессии



Пример регрессии



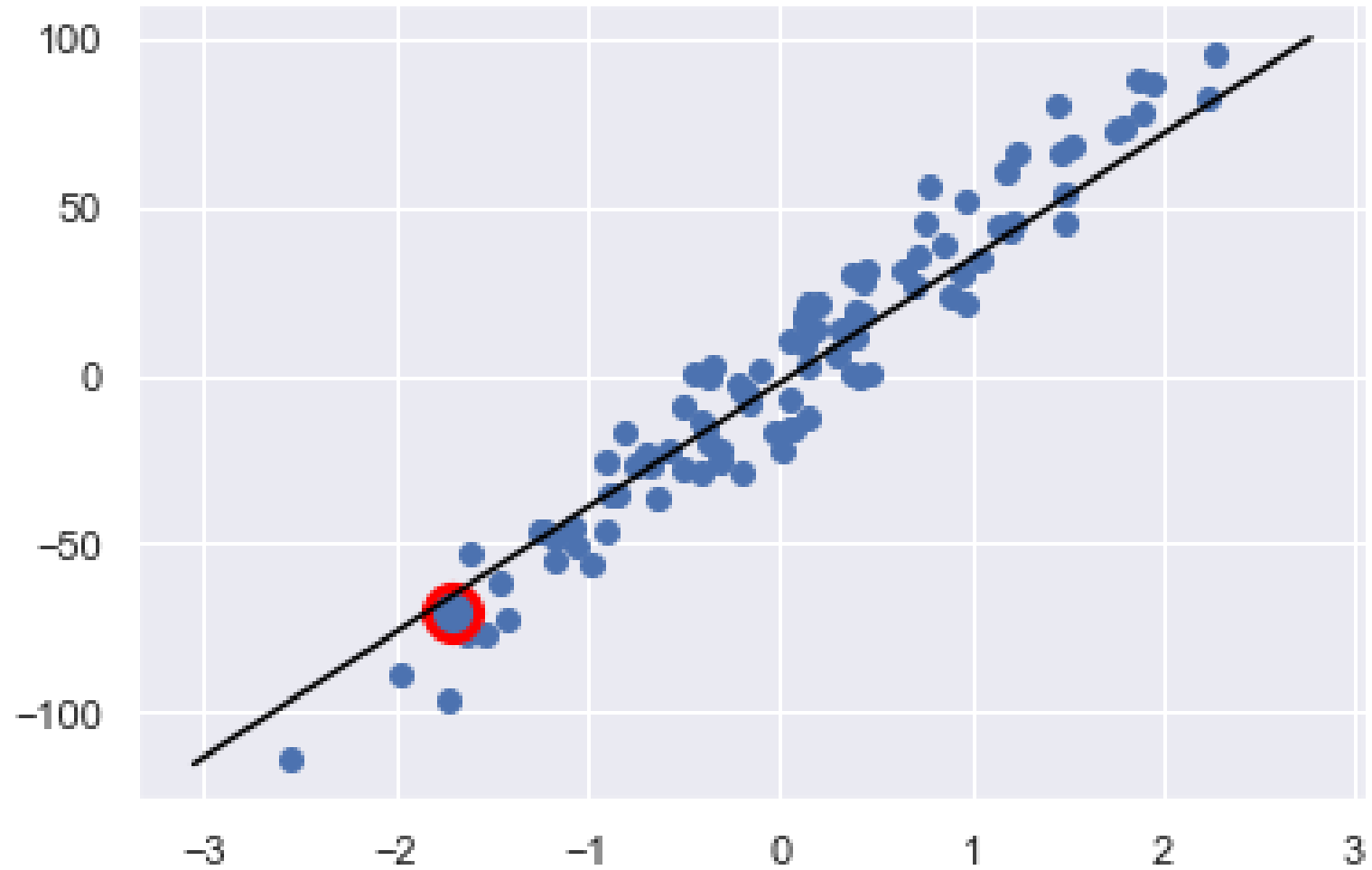
Пример регрессии



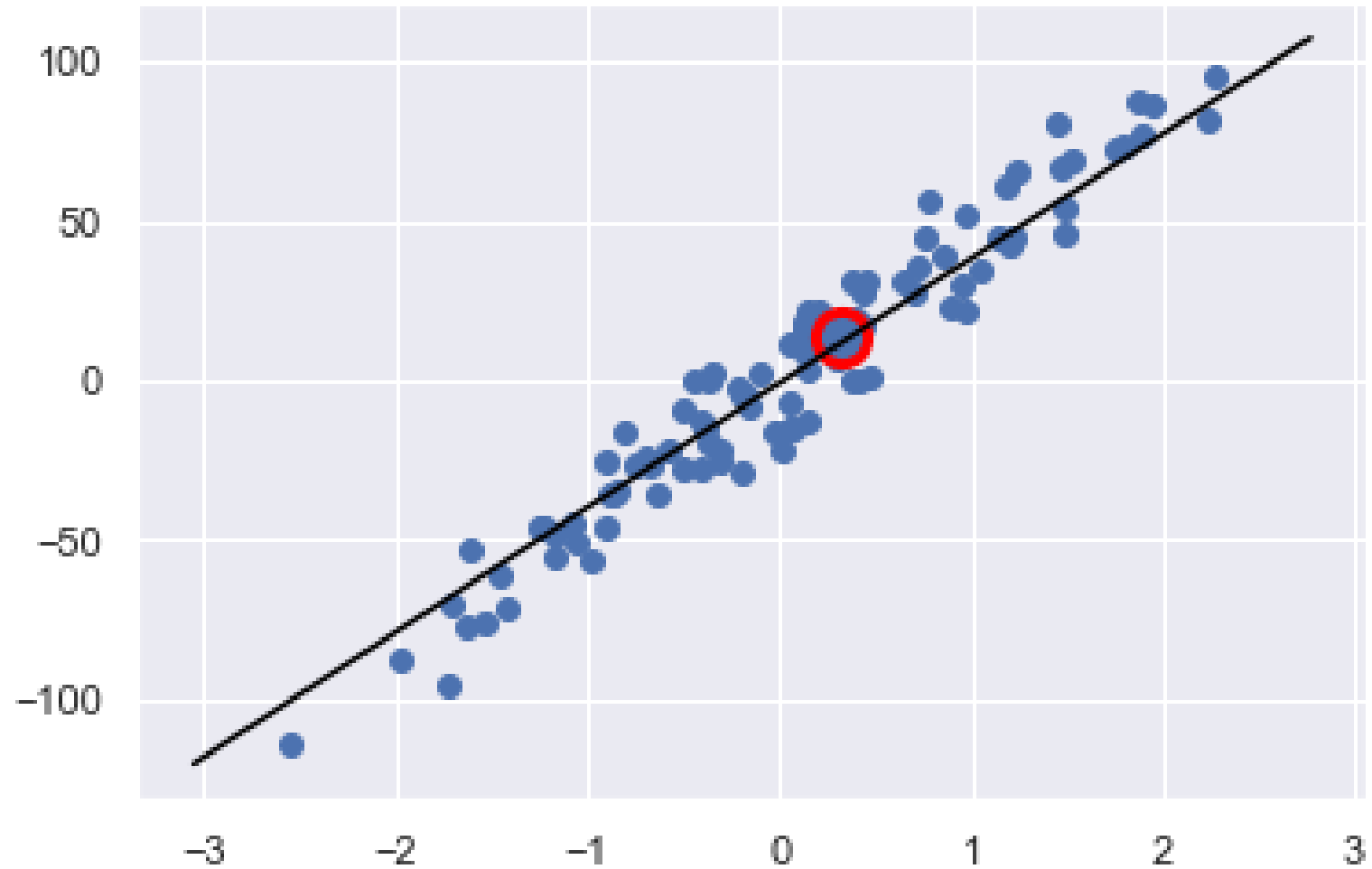
Пример регрессии



Пример регрессии

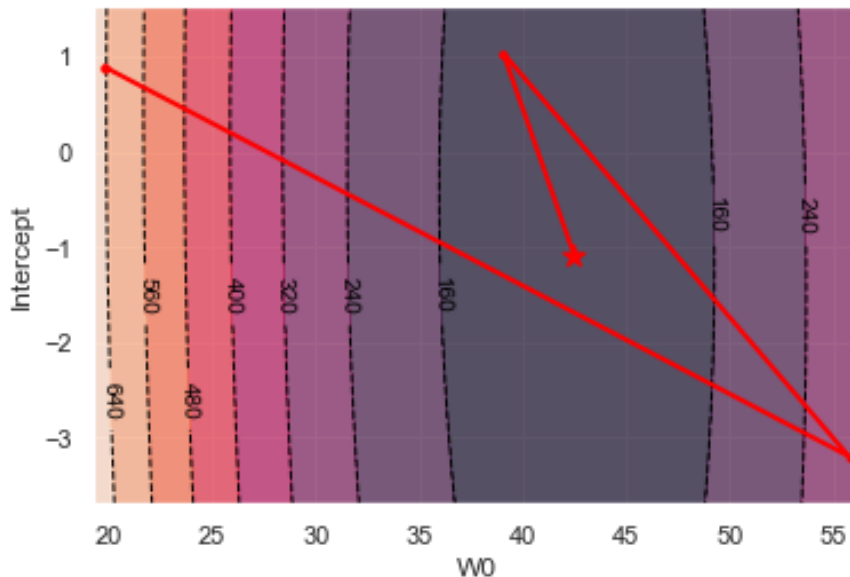


Пример регрессии

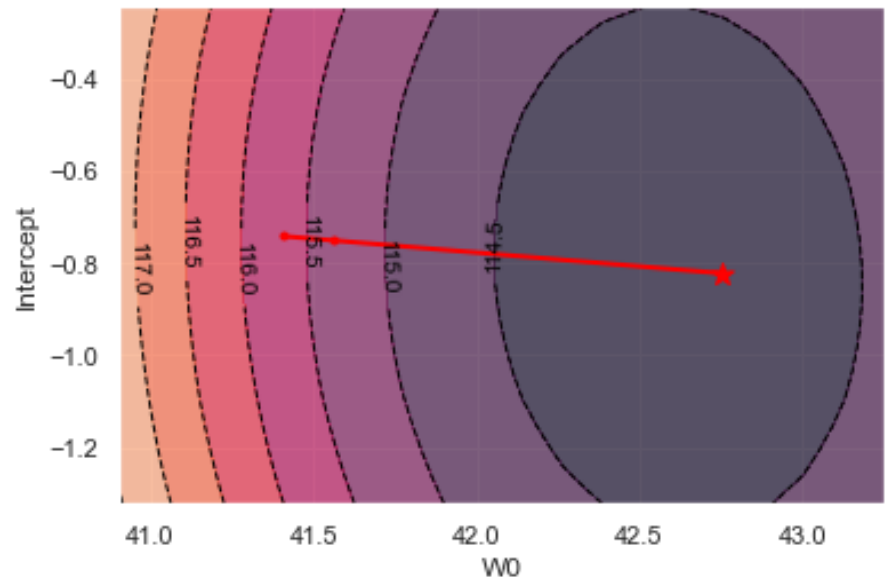


Пример регрессии

Квазиньютоновский метод (BFGS)
с робастной кусочно-линейной
функцией потерь



Условная оптимизация с
регуляризацией L2



Регуляризация

- В оптимизации – «улучшение» целевой функции

$$\min Q(w) + \gamma R(w)$$

$R(\cdot)$ – гладкая сильно выпуклая функция, γ -константа регуляризации
есть методы меняющие γ или $R(\cdot)$ «на ходу»

- В машинном обучении – контроль сложности модели для борьбы с переобучением.

- Обычно γ и $R(\cdot)$ фиксированы и $R(w) = L_p(w)$, тогда градиенты легко модифицируются, но необходимо «нормализовывать» параметры

- LASSO регуляризация $L_1(w) = \sum |w|$, градиент $\nabla Q(w) + \gamma \text{sign}(w)$

- Ridge («квадратичная» или «гребневая») регуляризация

$$L_2(w) = \sum w^2, \text{ градиент } \nabla Q(w) + 2\gamma w$$

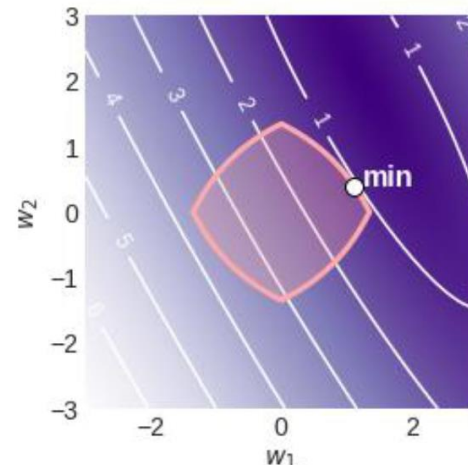
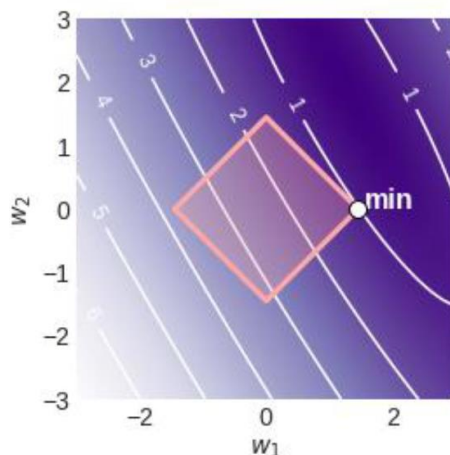
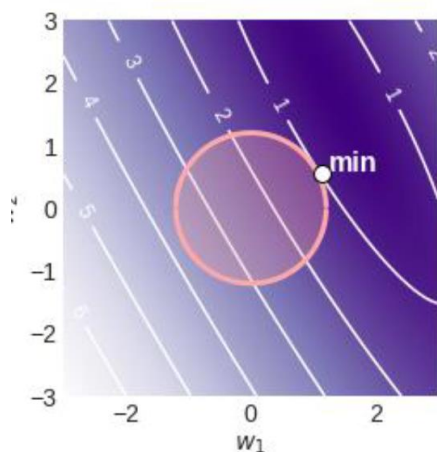
- Elastic Net= LASSO+Ridge

$$L_1(w) = \gamma_1 \sum |w| + \gamma_2 \sum w^2, \text{ градиент } \nabla Q(w) + 2\gamma_2 w + \gamma_1 \text{sign}(w)$$

Регуляризация для выпуклых целевых функций

- Выпуклые целевые функции – достаточно часто встречаются в задачах машинного обучения (линейные регрессии с регуляризацией, SVM, обобщенные линейные модели и др.)
- Для выпуклых целевых функций, можно показать, что:

$$\min Q(w) + \gamma L_p(w) \Leftrightarrow \begin{cases} \min Q(w) \\ L_p \leq C \end{cases}$$



- Чем меньше γ тем более модель настроена на «отбор» признаков

Связь регуляризации с вероятностными моделями

■ Пусть:

- $X \times Y$ вероятностное пространство с неизвестной плотностью распределения
- $Z = \{(x_i, y_i)\}$ – i.i.d. (все наблюдения независимы и одинаково распределены) выборка из $X \times Y$ размера l
- Искомая модель $P(y|x, w)$

■ Формула Байеса:

$$P(w|Z) = \frac{P(Z|w)P(w)}{P(Z)}$$

- $P(Z|w) = \prod_{i=1}^l P(x_i, y_i|w) = \prod_{i=1}^l P(y_i|w, x_i)P(x_i)$ - правдоподобие
- $P(Z), P(x)$ – не зависит от параметров нашей модели, поэтому
- Оценка max правдоподобия $\prod_{i=1}^l P(y_i|w, x_i) \rightarrow \max$
- Логарифмическое правдоподобие

$$\text{loglik}(Z, w) = \log P(Z|w) = \sum_i \log(P(y_i|w, x_i)) \rightarrow \max$$

Связь регуляризации с вероятностными моделями

- Если взять логарифмическую функцию потерь

$$L(x_i, y_i, w) = -\log(P(y_i, |x_i, w))$$

- То максимизация правдоподобия эквивалентно минимизации эмпирического риска

$$Q(w) = -\frac{1}{l} \sum_{i=1}^l \log(P(y_i, |x_i, w)) \rightarrow \min$$

- Но мы можем учесть $P(w)$ из формулы Байеса, если «параметризуем» априорное распределение $P(w; \gamma)$, тогда максимизируем $P(Z|w)P(w) = \prod_{i=1}^l P(y_i|w, x_i)P(x_i)P(w; \gamma) \rightarrow \max$

- И получаем:

$$Q(w) = -\frac{1}{l} \sum_{i=1}^l \log(P(y_i, |x_i, w)) - \log(P(w; \gamma)) \rightarrow \min$$

$-\log(P(w; \gamma))$ - регуляризатор

Связь регуляризации с вероятностными моделями

- Если предположить $E(w) = 0$ и $D(w) = C$, то:
- Квадратичный регуляризатор L_2 эквивалентен нормальному априорному распределению параметров модели

$$-\log(P(w, C)) = -\log \left[\frac{1}{(2\pi C)^{\frac{n}{2}}} \exp \left(-\frac{\|w\|^2}{2C} \right) \right] = \frac{\|w\|^2}{2C} + const$$

- А абсолютный регуляризатор L_1 эквивалентен априорному распределению Лапласа параметров модели

$$-\log(P(w, C)) = -\log \left[\frac{1}{(2C)^n} \exp \left(-\frac{|w|}{C} \right) \right] = \frac{|w|}{2C} + const$$

- C – параметр регуляризации