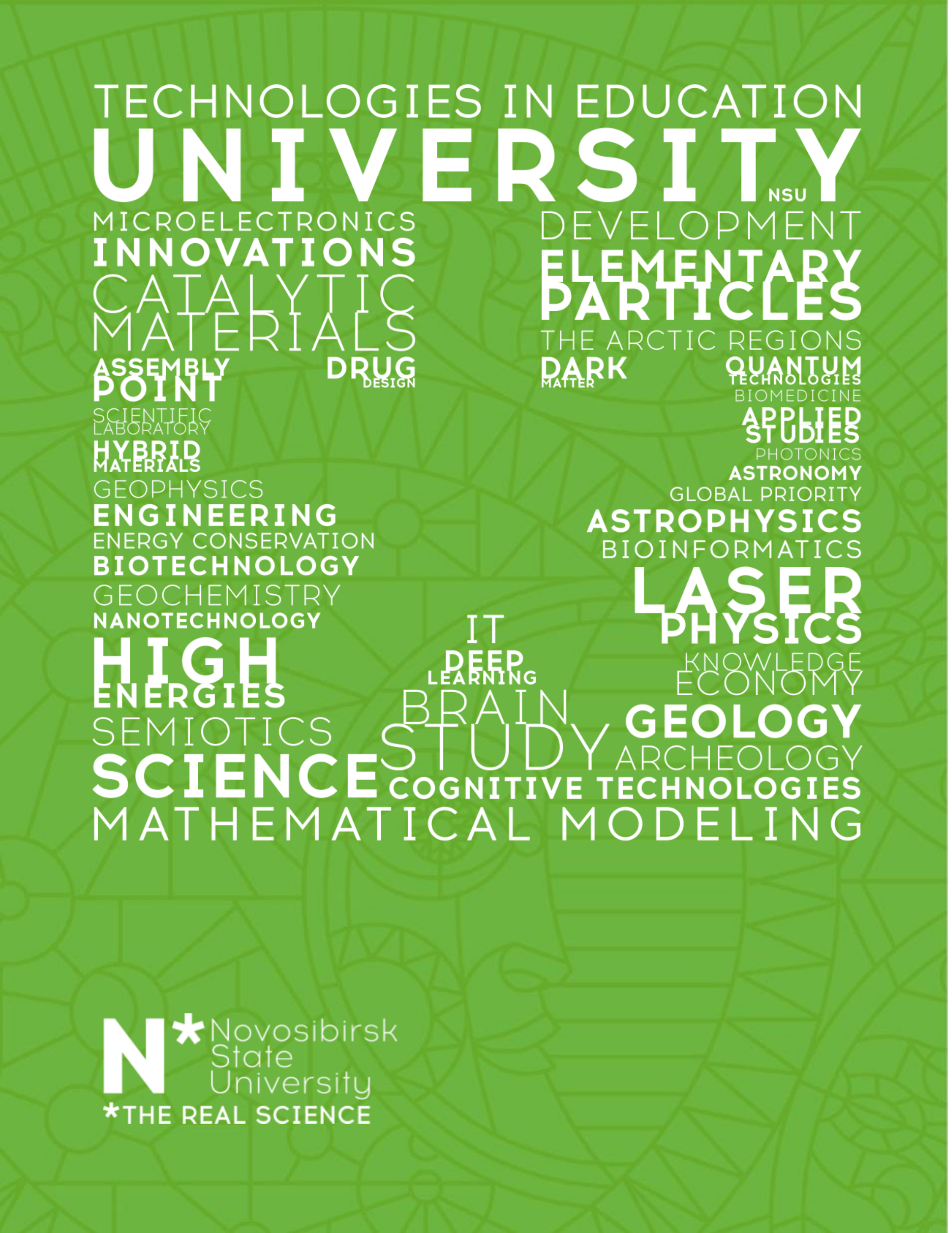


Машинное обучение
Семинар 3

Глушенко Андрей Валерьевич
ФФ НГУ

Практические аспекты

Оптимальное решение

Регуляризация

Логистическая регрессия

Алгоритм градиентного спуска

1

Оптимальное решение

$$Z = 10x_1 + 1000x_2 \rightarrow \max$$

$1x_1 + 20x_2 \leq 200$ - Ограничение по детали 1

$0x_1 + 1x_2 \leq 4$ - Ограничение по детали 2

$x_1 \geq 0$- число машин за 1000 рублей, которое можем произвести

$x_2 \geq 0$- число машин за 1000 рублей, которое можем произвести

Задача: найти такие x_1, x_2, что Z максимален

1000 рублей

10 рублей

Оптимальное решение

$$Z = 10x_1 + 1000x_2 \rightarrow \max$$

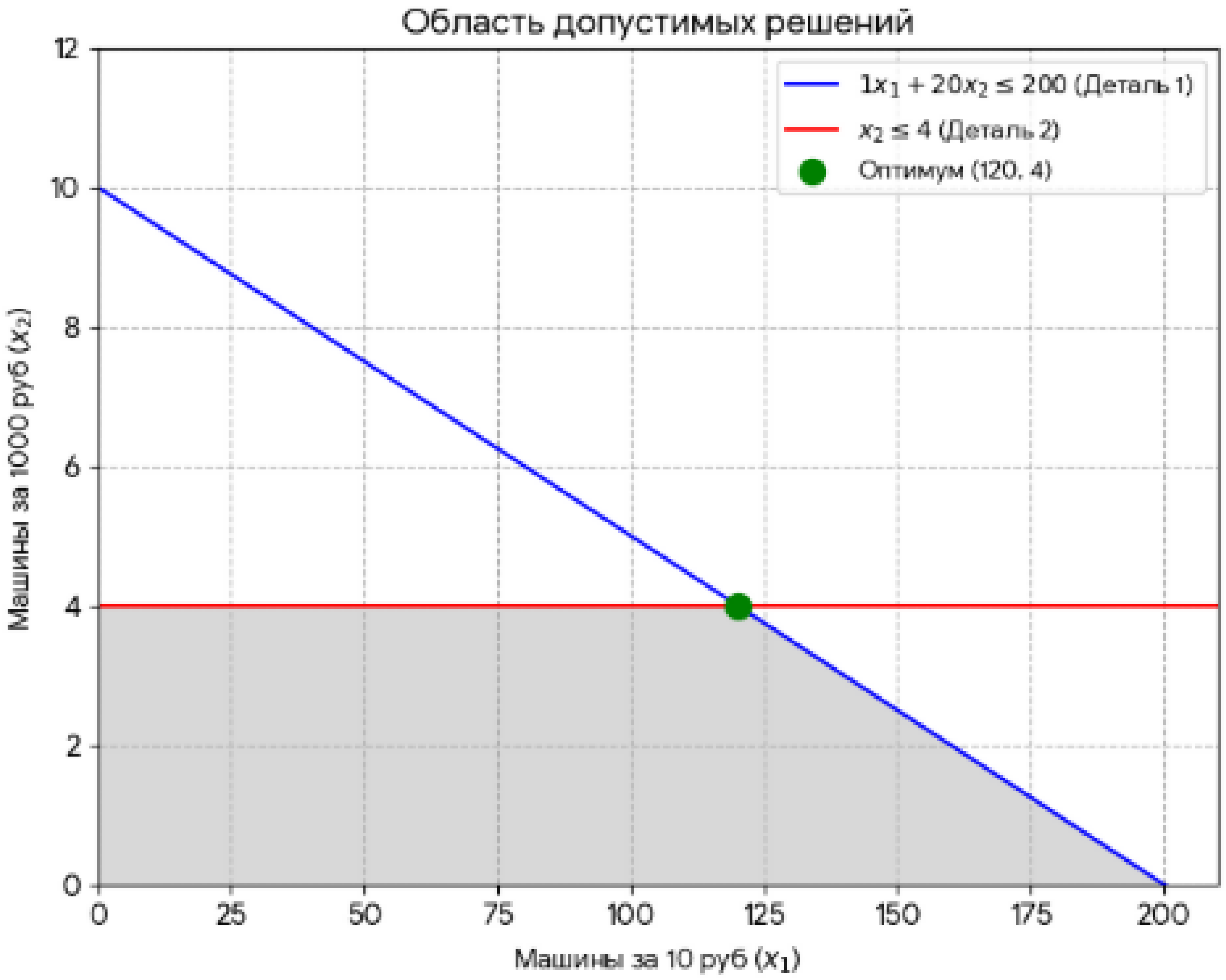
$$1x_1 + 20x_2 \leq 200$$

$$0x_1 + 1x_2 \leq 4$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

Решение: $x_1 = 120, x_2 = 4$



Оптимальное решение

В контексте регрессионного анализа понятие **оптимального решения** — это поиск таких параметров модели, которые минимизируют «ошибку» между предсказаниями алгоритма и реальными данными.

Оптимальное решение

Для поиска оптимального решения нам необходим математический критерий — функция потерь **(Loss Function)**.

Она показывает, насколько сильно мы ошиблись.

Чаще всего в регрессии используется Метод наименьших квадратов (МНК). В этом случае оптимальным считается решение, при котором сумма квадратов отклонений минимальна:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \to \min$$

- y_i — реальное значение.
- $\hat{y}_i$ — предсказанное значение.

Оптимальное решение | Регуляризация

Иногда, чтобы найти по-настоящему оптимальное решение, в формулу ошибки добавляют «штраф» за слишком большие веса (L1 или L2 регуляризация). Это заставляет модель быть более устойчивой и не переобучаться.

L2-регуляризация (Ridge / Гребневая регрессия)

$$Loss = \sum (y_i - \hat{y}_i)^2 + \lambda \sum \omega_j^2$$

L2-регуляризация заставляет веса быть маленькими, стремящимися к нулю, но никогда не делая их ровно нулевыми.

- Модель становится более «гладкой» и устойчивой к выбросам.
- Используем, когда много признаков, и каждый из них по чуть-чуть вносит вклад в результат.

L1-регуляризация (Lasso / Лассо)

$$Loss = \sum (y_i - \hat{y}_i)^2 + \lambda \sum |w_j|$$

L1-регуляризация способна занулять коэффициенты перед неважными признаками.

- Модель сама «выбирает» самые важные переменные и отбрасывает лишние (Feature Selection).
- Используем, когда признаков очень много, но подозреваем, что реально на результат влияют лишь немногие из них.

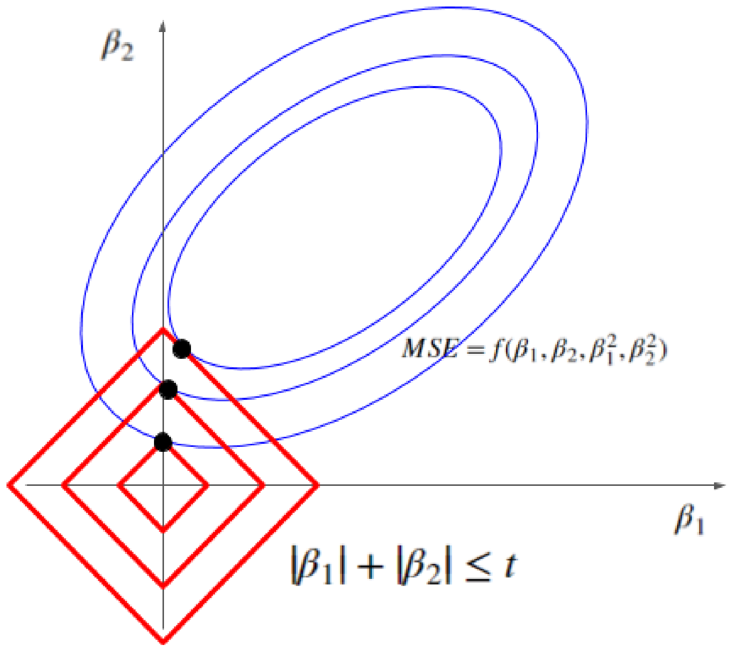
Что такое лямбда?

Это гиперпараметр, который настраивает мощность регуляризации:

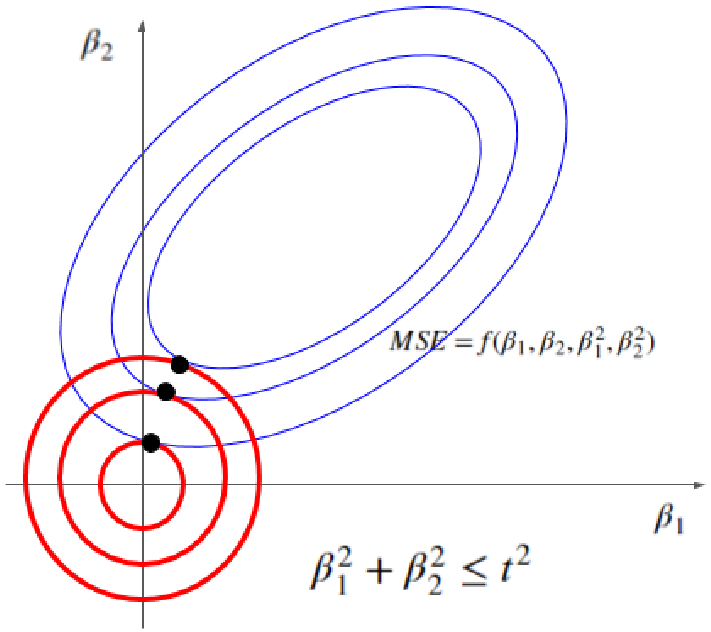
- Если $\text{лямбда} = 0$ – регуляризации нет (обычная регрессия).
- Если лямбда очень большая – все веса станут почти нулевыми (модель станет слишком простой, «недообученной»).

Регуляризация

Shrink t, and check how the corresponding β_1 β_2 behave



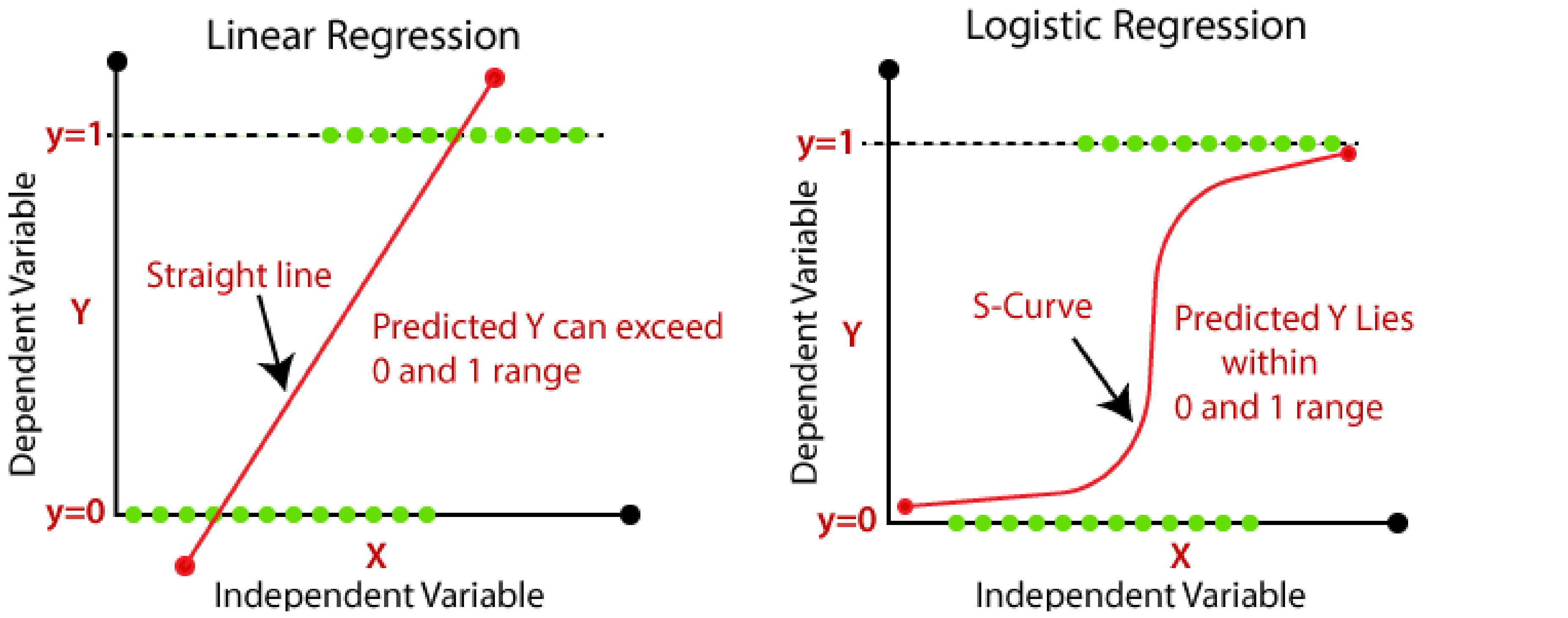
L1 regularization



L2 regularization

Логистическая регрессия

Логистическая регрессия — это метод машинного обучения, который используется для предсказания **вероятности** того, что объект относится к определенной категории.



Логистическая регрессия

Примеры использования:

- Банки: Одобрить кредит или нет (вернет ли клиент деньги?).
- Медицина: Есть ли у пациента заболевание на основе симптомов и анализов.
- Маркетинг: Купит ли пользователь товар, кликнет ли по рекламе.
- Безопасность: Является ли транзакция мошеннической.

Алгоритм

1. Линейное предсказание

Сначала вычисляется линейная комбинация признаков x и весов w с добавлением смещения (bias) b:

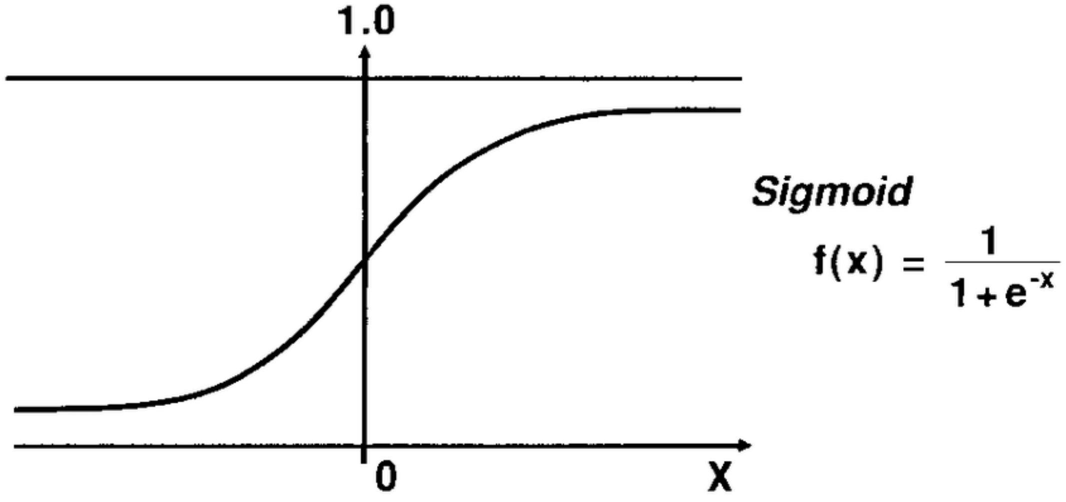
$$z = w_1x_1 + w_2x_2 + ... + w_nx_n + b = w^Tx + b$$

2. Функция активации (Сигмоида)

Чтобы превратить любое число z в вероятность от 0 до 1, используется логистическая функция (сигмоида):

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Результат $y' = \sigma(z)$ — это предсказанная вероятность того, что объект относится к классу «1».



Алгоритм

3. Функция потерь (Log Loss)

Для оценки ошибки используется логистическая функция потерь (Binary Cross-Entropy). Она сильно штрафует модель за уверенные, но неверные прогнозы:

$$L(y, y') = -\frac{1}{N}\sum_{i=1}^{N}[y_i \ln(y'_i) + (1 - y_i)\ln(1 - y'_i)]$$

Где y — реальная метка (0 или 1), а y' — предсказание модели.

Алгоритм

4. Обучение (Градиентный спуск)

Веса обновляются итеративно, чтобы минимизировать функцию потерь. На каждом шаге веса корректируются в сторону, противоположную градиенту:

$$w_{new} = w_{old} - \alpha \cdot \frac{\partial L}{\partial w}$$

Где α — скорость обучения (learning rate), а производная $\frac{\partial L}{\partial w}$ для каждого веса j выглядит на удивление просто:

$$\frac{\partial L}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} (y_i' - y_i) x_{ij}$$