Обучение с учителем

Санкт-Петербургский государственный университет
Кафедра статистического моделирования

16 сентября 2025, Санкт-Петербург

Введение

Машинное обучение — это раздел искусственного интеллекта, в котором разрабатываются методы и алгоритмы, позволяющие компьютерам обнаруживать закономерности в данных и делать прогнозы.

Обучение с учителем — один из способов машинного обучения, в ходе которого для каждого наблюдения в обучающем наборе известно, какой результат является правильным.

Пример задач:

- Регрессия: предсказание стоимости недвижимости, количества продаж некоторого товара, погоды.
- Классификация: предсказание ценовой категории товара, типа изображения, болеет ли человек или нет.

Постановка задачи

Дано:

1. Пространство объектов $\mathbb{X}$ — множество описаний объектов (например, фотографии, тексты, таблицы с признаками).
2. Пространство ответов $\mathbb{Y}$ — множество меток или значений, которые нужно предсказывать (например, классы «кот»/«собака», цена товара).
3. Доступная выборка $D = \{(x_i, y_i)\}_{i=1}^n$, где $x_i \in \mathbb{X}$, $y_i \in \mathbb{Y}$.

Модель:

$$y = f(x) + \varepsilon,$$

где $f(x)$ — некоторая фиксированная (но неизвестная) функция, ε — шум, $\mathsf{E}\varepsilon = 0$ и ε не зависит от x.

Предположение: $f(x)$ лежит в некотором классе функций (например, в классе линейных функций).

Задача: по выборке D построить оценку $\hat{f}(x)$ функции $f(x)$ в выбранном классе функций.

Функция потерь и ее минимизация

Чтобы оценить, насколько хорошо модель предсказывает ответы, используется **функция потерь** для набота данных $L(Y, \hat{Y})$. Она показывает, насколько велико расхождение между истинными значениями Y и его предсказаниями $\hat{Y} = \hat{f}(X)$. Такая функция потерь обычно состоит из **функций потерь для одного индивида** $l(y, \hat{y})$, то есть это может быть сумма, максимум и т.д. по всем таким функциям.

Тогда задача машинного обучения — минимизация выбранной функции потерь:

$$L(Y, \hat{Y}) \longrightarrow \min .$$

В большинстве случаев вычислить точку минимума функции потерь аналитически не представляется возможным, поэтому для его нахождения прибегают к методам **детерменированной и стохастической оптимизации** (например, перебор значений по сетке, метод Ньютона и квазиньютоновские методы, (стохастический) градиентный спуск, случайный поиск).

Градиентный спуск

Градиентный спуск является наиболее распространенным алгоритмом оптимизации в машинном обучении.

Пусть g — некоторая гладкая функция, у которой необходимо найти минимум. Обозначим $p_n = -\nabla g(x_n)$ — направление антиградиента в точке x_n. Тогда

$$x_{n+1} = x_n + \alpha p_n,$$

где α — гиперпараметр, отвечающий за скорость обучения.

Условия сходимости: выпуклость g, липшицевость ∇g, ...

Критерий остановки: достижение определенного числа итераций, малая норма градиента, малое изменение значения функции.

Модификации градиентного спуска

В машинном обучении минимизируем функцию потерь:

$$g(x) = L(Y, \hat{Y}(x)) = \frac{1}{n} \sum_{i=1}^{n} l_i(y_i, \hat{y}_i(x)).$$

Варианты градиентного спуска:

1. (Batch) Gradient Descent (GD):

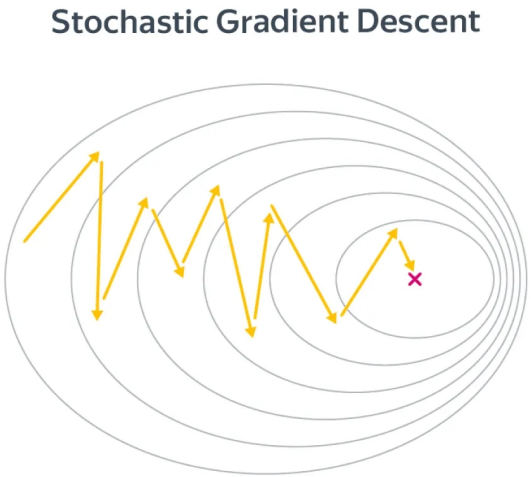
$$p_n = -\nabla g(x_n) = -\sum_{i=1}^{n} \nabla l_i(x_n).$$

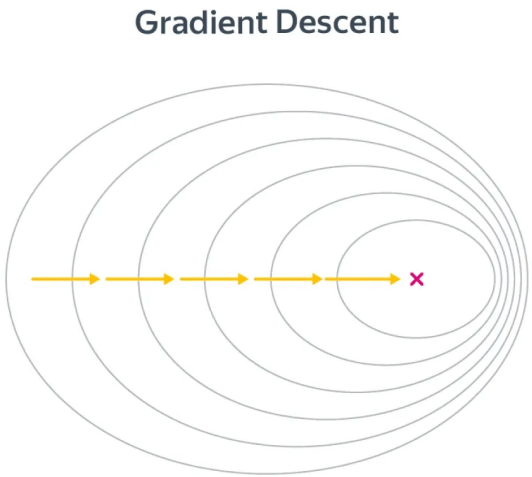
2. Mini-batch GD: случайным образом выбирается m наблюдений и делается шаг с $p_n = -\sum_{i=1}^{m} \nabla g_{j_i}(x_n)$.
3. Stochastic GD: mini-batch GD с $m = 1$.
4. Adam (Adaptive Moment Estimation): основан на GD, каждый параметр модели имеет собственную адаптивную скорость обучения, основанную на прошлых градиентах.

Преимущества стохастического градиентного спуска

Использование стохастического градиентного спуска вместо обычного обусловлено следующими преимущетвами:

1. Меньшее использование памяти: не храним промежуточные вычисления для всей выборки;
2. Обработка онлайн: не надо хранить сразу всё для обучения, можно добавлять данные по мере поступления;
3. Быстрота: одно обновление весов происходит значительно быстрее, за меньшее количество эпох можем сойтись;
4. Регуляризация: не идём сразу в минимум, что может привести к переобучению.





Процесс обучения

Процесс обучения любого алгоритма машинного обучения выглядит следующим образом:

1. Выборка D предварительно случайным образом разбивается на тренировочную и тестовую: $D = D_{\text{train}} \sqcup D_{\text{test}}$ (чтобы избежать закономерности, заложенной в порядок данных).

2. На тренировочных данных модель обучается: минимизируется выбранная функция потерь $L(y, \hat{y})$.

3. Если в модели есть гиперпараметр, то выделяют валидационную выборку D_{val} для подбора гиперпараметров модели или производится остановка оптимизации функции потерь.

Оценка качества модели

После обучения нужно оценить качество/обобщающую способность модели. Для этого на тестовом множестве вычисляются различные метрики. Выбираются они в зависимости от задачи.

Важно понимать разницу между функцией потерь и метрикой качества:

- Функция потерь возникает в тот момент, когда мы сводим задачу построения модели к задаче оптимизации.
- Метрика — внешний, объективный критерий качества, обычно зависящий не от параметров модели, а только от предсказанных меток.

Таблица: Метрики для задач регрессии и классификации

Регрессия	MSE, RMSE, MAE, MAPE, WAPE
Классификация	Accuracy, Precision, Recall, F1-score, ROC AUC, PR AUC

Линейная классификация

Пусть целевая переменная y принимает значения $\{-1, 1\}$. Хотим обучить модель так, чтобы плоскость, которую она задает, как можно лучше отделяла объекты одного класса от другого.

Линейный классификатор:

$$\hat{y} = \hat{f}(x; w) = \operatorname{sign}\langle x, w\rangle.$$

Функция потерь:

$$L(y, \hat{y}) = \sum_{i=1}^{n} \mathbb{I}[y_i \langle x_i, w\rangle < 0] \longrightarrow \min_w.$$

Линейная классификация. Отступ

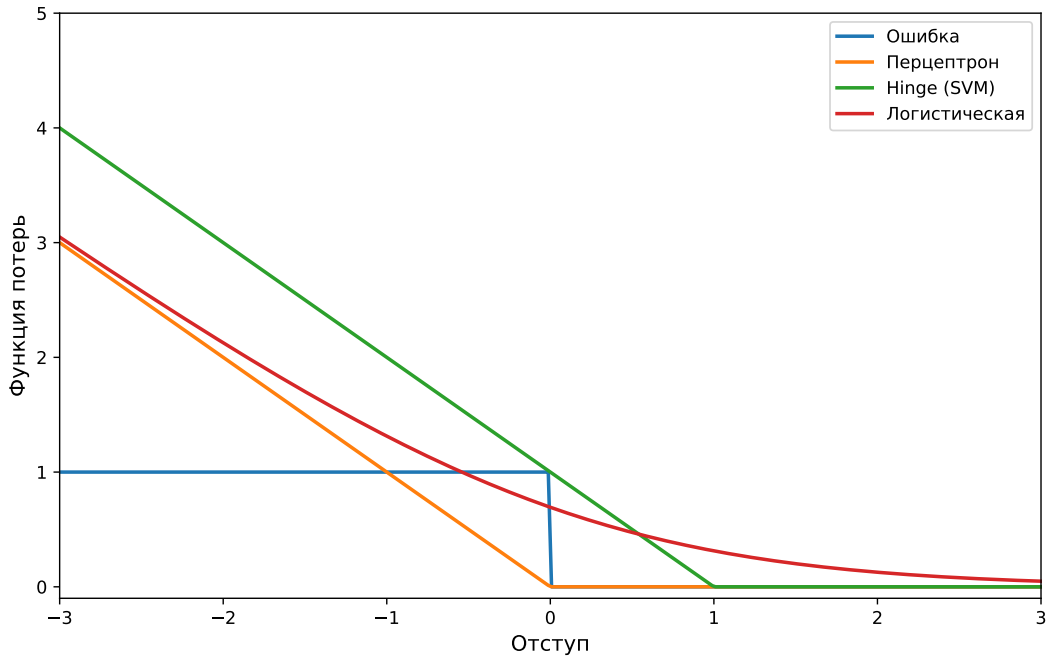
Величина $M_i = y_i \langle x_i, w \rangle$ называется **отступом** (margin) классификатора. Абсолютная величина отступа говорит о степени уверенности классификатора.

Проблема: функция $\mathbb{I}[M < 0]$ кусочно-постоянная, следовательно функцию потерь невозможно оптимизировать методами, основанными на производных, поскольку во всех точках производная равна нулю.

Решение: можно мажорировать эту функцию более гладкой функцией и минимизировать функцию потерь с этой мажорирующей функцией с помощью методов численной оптимизации.

Линейная классификация. Функции потерь

1. Перцептрон: $L(M) = \max(0, -M)$ — отступы учитываются только для неправильно классифицированных объектах пропорционально величине отступа.
2. Hinge (SVM): $L(M) = \max(0, 1 - M)$ — объекты, которые классифицированы правильно, но не очень «уверенно», продолжают вносить свой вклад в градиент.
3. Логистическая: $L(M) = \ln\left(1 + e^{-M}\right)$.



Логистическая регрессия

Посмотрим на задачу классификации как на задачу предсказания вероятностей и введём следующую модель: есть зависимая переменная y, принимающая значения только 0 и 1:

$$P(y = 1|x) = p = h(z),$$

где $h(z) \in [0, 1]$, $z = \langle x, w \rangle$ — линейная регрессия.
Преобразование h, которое нам подходит — сигмоида:

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

Функция правдоподобия для Бернулли ($y \in \{0, 1\}$):

$$p(y \mid \mathbf{X}, w) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i}.$$

Прологарифмируем:

$$\sum_{i=1}^{n} \Big[y_i \ln(\sigma(\langle x_i, w \rangle)) + (1 - y_i) \ln(1 - \sigma(\langle x_i, w \rangle)) \Big].$$

Логистическая регрессия. Связь с отступом

Теперь пусть $y \in \{-1, 1\}$. Тогда, поскольку $\sigma(z) = 1 - \sigma(-z)$, логарифм правдоподобия можно представить в следующем виде:

$$\begin{aligned}\ln p(y \mid \mathbf{X}, w) &= -\sum_{i=1}^{n} \Big[\mathbb{I}[y_i = 1]\sigma(z_i) + \mathbb{I}[y_i = -1]\,(1 - \sigma(z_i))\Big] \\ &= -\sum_{i=1}^{n} \ln \sigma(y_i \langle x_i, w \rangle) \\ &= \sum_{i=1}^{n} \ln\left(1 + e^{-M}\right)\end{aligned}$$

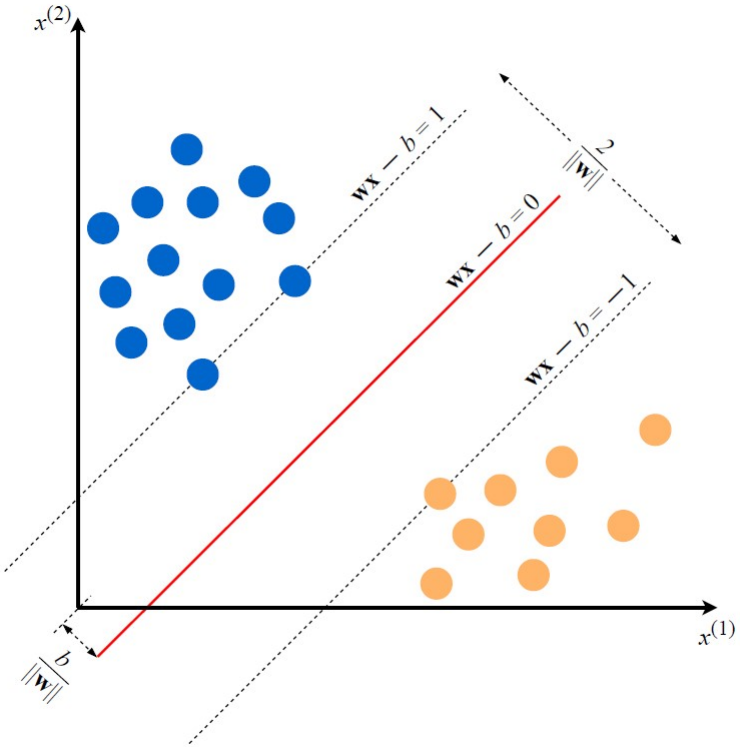
Таким образом, функцию потерь в логистической регрессии можно представить в виде функции от отступа.

Метод опорных векторов (SVM): Основная идея

Геометрическая интуиция

Цель: Найти не просто разделяющую гиперплоскость, а ту, которая максимизирует зазор (отступ) между классами.

- Гиперплоскость: $\mathbf{w}^T\mathbf{x} + b = 0$
- Отступ (Margin) — расстояние до ближайших точек классов.
- Опорные векторы — точки, лежащие на границах отступа. Именно они «определяют» положение гиперплоскости.



Задача оптимизации (для линейно разделимых данных):

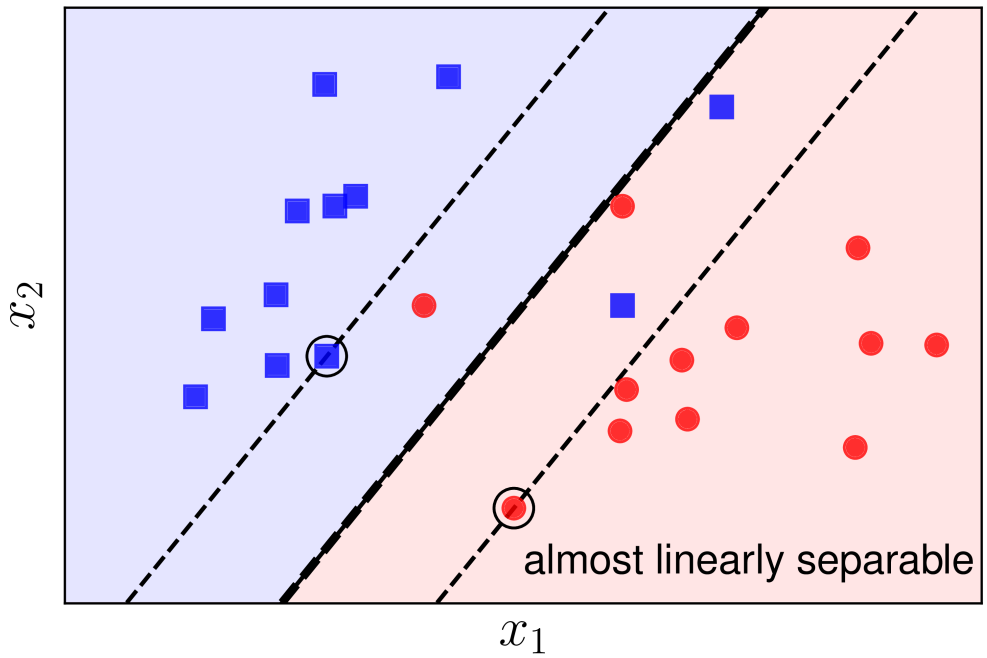
$$\begin{cases} \min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \\ \text{при условии } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n \end{cases}$$

Минимизируем норму вектора весов $\Rightarrow$ максимизируем зазор $\gamma = \frac{2}{||\mathbf{w}||}$.
Это задача квадратичной оптимизации, которая решается путём составления двойственной задачи (по теореме Куна-Таккера).

SVM для линейно неразделимых данных

Мягкий зазор (Soft Margin)

- В реальных данных идеальная линейная разделимость — редкость.
- Вводятся ослабляющие переменные (slack variables) $\xi_i \geq 0$.
- Они позволяют точкам нарушать границу отступа.
- ξ_i — штраф за неправильную классификацию или нахождение в полосе зазора.



Новая задача оптимизации (Soft Margin SVM):

$$\begin{cases} \min_{\mathbf{w},b,\xi_i} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{при условии:} \\ \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1,\ldots,n \\ \quad \xi_i \geq 0, \quad i = 1,\ldots,n \end{cases}$$

Параметр C управляет компромиссом:

- **Большой** C: Больший штраф за ошибки $\Rightarrow$ уже разделение, риск переобучения.
- **Малый** C: Меньший штраф за ошибки $\Rightarrow$ шире зазор, больше обобщающая способность.

SVM vs. Логистическая регрессия

- **Логистическая регрессия**: Строит вероятностную модель $P(y = 1|\mathbf{x})$. Минимизирует логистическую функцию потерь по всем объектам. Все точки влияют на решение.
- **SVM**: Строит разделяющую гиперплоскость. Фокусируется на максимизации отступа; решение зависит только от опорных векторов. Более устойчив к выбросам.

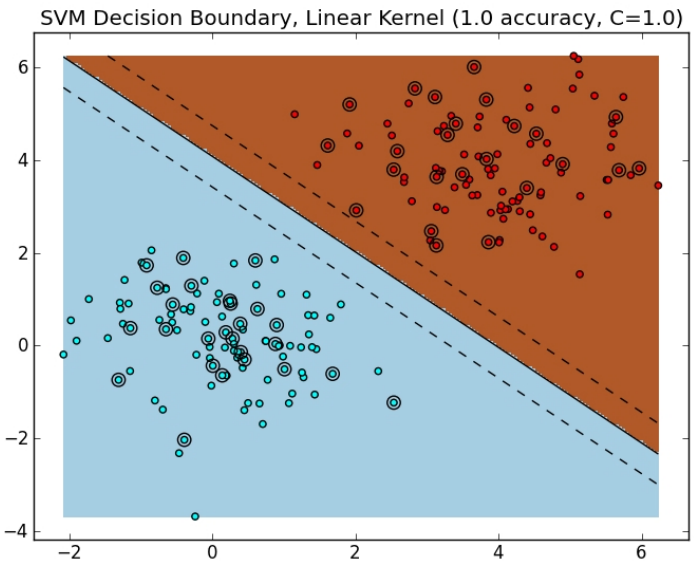
Нелинейность: Kernel Trick

Что если данные нелинейно разделимы?
Идея: Отображаем данные в пространство большей размерности ($\phi(\mathbf{x}) : \mathbb{R}^p \to \mathbb{R}^m$), где они становятся линейно разделимыми.
Ядро (Kernel): $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. Позволяет работать в высокомерном пространстве без явного вычисления $\phi(\mathbf{x})$.

Линейное ядро

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

- Исходное признаковое пространство.
- Быстрое обучение.



Нелинейные ядра

- **RBF (Gaussian)**: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2)$
- **Полиномиальное**: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \cdot \mathbf{x}_i^T \mathbf{x}_j + r)^d$

Гибкие сложные границы решений. Требуют подбора параметров (γ, d).