# Шпаргалка по линейным моделям / Концепции

# Cheatsheet (XeLaTeX)

Краткий справочник

April 2, 2025

## Contents



## 1 Линейная Регрессия (`Linear Regression`)

### Основная Идея

Простейшая модель для предсказания **непрерывного** значения (например, цены дома, температуры). Мы пытаемся найти наилучшую прямую (или гиперплоскость в многомерном случае), которая описывает зависимость между признаками ($X$) и целевой переменной ($y$).
Модель: $y \approx \hat{y} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^T\mathbf{x}$ Где **w** - вектор весов (параметров) модели, включая свободный член $w_0$ (bias term), **x** - вектор признаков объекта (с добавленным $x_0 = 1$). **Интерпретация весов:** Коэффициент $w_j$ показывает, на сколько в среднем изменится предсказание $\hat{y}$, если признак $x_j$ увеличить на 1, при условии, что все остальные признаки остаются неизменными.

### Функция Потерь: MSE (`Mean Squared Error`)

Чтобы понять, насколько хорошо наша прямая подходит к данным, мы измеряем ошибку. Самый частый способ - **Среднеквадратичная Ошибка (MSE)**. Мы суммируем квадраты разностей между реальными значениями ($y_i$) и предсказаниями модели ($\hat{y}_i$) и делим на количество примеров ($m$).

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 = \frac{1}{m}\sum_{i=1}^{m}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

**Почему квадрат?** Он штрафует большие ошибки сильнее и делает функцию потерь дифференцируемой.

### Обучение: Идея Градиентного Спуска (GD)

Представьте, что MSE - это холмистая местность, а мы стоим где-то на склоне. Наша цель - найти самую низкую точку (минимум MSE). **Градиентный спуск (GD)** - это как катиться с горы:

- Смотрим, в каком направлении уклон самый крутой (**градиент** $\nabla MSE$).
- Делаем небольшой шаг в **противоположном** направлении (анти-градиент). Длина шага контролируется **скоростью обучения** (*learning rate*, $\alpha$).
- Повторяем, пока не дойдем до дна (или почти).

Формула обновления весов: $\mathbf{w} := \mathbf{w} - \alpha\nabla_{\mathbf{w}}MSE(\mathbf{w})$
**Варианты GD:**

- **Batch GD**: Градиент считается по **всей** обучающей выборке на каждом шаге. Точно, но медленно на больших данных.
- **Stochastic GD (SGD)**: Градиент считается по **одному** случайно выбранному примеру на каждом шаге. Быстро, шумно (может "прыгать" вокруг минимума), хорошо для очень больших данных и онлайн-обучения.
- **Mini-batch GD**: Компромисс. Градиент считается по небольшой **случайной подвыборке** (*batch*) на каждом шаге. Сочетает преимущества Batch и SGD. Самый популярный вариант.

### Аналитическое Решение (Normal Equation)

Для линейной регрессии с MSE существует **точное аналитическое решение**, позволяющее найти оптимальные веса **w** без итераций GD.

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Где **X** - матрица признаков (объекты по строкам, признаки по столбцам, с добавленным столбцом единиц), **y** - вектор целевых значений.
**Плюсы:** Точно, не требует подбора learning rate. **Минусы:** Требует вычисления обратной матрицы $(\mathbf{X}^T\mathbf{X})^{-1}$, что вычислительно сложно ($O(n^3)$, где $n$ - число признаков) и может быть невозможно, если матрица вырождена (признаки линейно зависимы). Непрактично при очень большом числе признаков.

### Основные Предположения Линейной Регрессии

Модель работает лучше всего, когда выполняются некоторые предположения (на собеседовании важно знать хотя бы названия):

- **Линейность:** Средняя зависимость $y$ от $X$ является линейной.
- **Независимость ошибок:** Ошибки предсказаний для разных наблюдений независимы.
- **Гомоскедастичность:** Разброс (варианция) ошибок одинаков для всех значений $X$. *Аналогия: толщина "облака" точек вокруг линии регрессии примерно одинакова по всей длине.*
- **Нормальность ошибок:** Ошибки распределены нормально (важно для построения доверительных интервалов).

Нарушение предположений не всегда делает модель бесполезной, но может влиять на надежность выводов.

### Полиномиальная Регрессия

Что если зависимость нелинейная? Можно добавить **полиномиальные признаки** - степени существующих признаков ($x^2, x^3$) или их взаимодействия ($x_1x_2$). Пример: $y \approx w_0 + w_1x + w_2x^2$. **Важно:** Модель все еще остается **линейной по параметрам w**! Мы просто применяем линейную регрессию к расширенному набору признаков ($1, x, x^2$). Легко переобучается, требует регуляризации.

## 2 Логистическая Регрессия (`Logistic Regression`)

### Основная Идея

Используется для задач **бинарной классификации** (ответ 0 или 1, "да" или "нет"). Вместо прямого предсказания класса, она моделирует **вероятность** принадлежности объекта к классу 1. *Аналогия: предсказать не "сдал/не сдал" экзамен, а вероятность сдачи в зависимости от часов подготовки.*

### Сигмоида (Логистическая Функция)

Линейная комбинация признаков ($\mathbf{w}^T\mathbf{x}$) может дать любое вещественное значение. Чтобы получить вероятность (от 0 до 1), результат пропускают через **сигмоидную функцию** $\sigma(z)$:

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad \text{где } z = \mathbf{w}^T\mathbf{x}$$

Предсказание модели: $\hat{p} = P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$. Решение о классе принимается по порогу (обычно 0.5): если $\hat{p} \geq 0.5$, то класс 1, иначе класс 0.

### Функция Потерь: LogLoss (Логарифмическая Функция Потерь)

MSE плохо подходит для вероятностей. Используется **LogLoss** (или Бинарная Кросс-Энтропия).

$$LogLoss = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{p}_i) + (1-y_i)\log(1-\hat{p}_i)]$$

**Идея:** Сильно штрафует модель, если она уверенно предсказывает неверный класс (например, дает $\hat{p} = 0.99$, когда реальный класс $y = 0$). Когда предсказание правильное, штраф маленький. Обучается также с помощью GD.

### Softmax (для Мультиклассовой Классификации)

Если классов больше двух, используется обобщение логистической регрессии - **Softmax Regression**. Для каждого класса $k$ вычисляется своя "оценка" $z_k = \mathbf{w}_k^T\mathbf{x}$. Затем эти оценки преобразуются в вероятности с помощью **функции Softmax**:

$$P(y = k|\mathbf{x}) = \hat{p}_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{для } k = 1, \ldots, K$$

Softmax гарантирует, что все $\hat{p}_k$ будут от 0 до 1 и их сумма будет равна 1. Функция потерь - **Cross-Entropy Loss**.

## 3 Регуляризация L1 и L2

### Что и Зачем?

**Регуляризация** - это техника борьбы с **переобучением** (*overfitting*) линейных (и не только) моделей. Переобучение происходит, когда модель слишком сложная и "запоминает" обучающие данные вместо того, чтобы выучить общие закономерности. **Идея:** Добавить к основной функции потерь (MSE или LogLoss) **штраф** за большие значения весов **w**. *Аналогия: Мы не просто просим модель хорошо описать данные (минимизировать MSE/LogLoss), но и говорим ей: "Будь проще! Не усложняй без необходимости!" (штрафуем за большие веса).*

$$J_{reg}(\mathbf{w}) = J_{original}(\mathbf{w}) + \lambda \cdot R(\mathbf{w})$$

Где $J_{original}$ - MSE или LogLoss, $R(\mathbf{w})$ - регуляризационный член, $\lambda$ (*lambda*) - **коэффициент регуляризации** (гиперпараметр), контролирующий силу штрафа. **Важно:** Перед применением регуляризации признаки обычно **масштабируют** (стандартизируют или нормализуют), чтобы штраф не зависел от исходного масштаба признаков.

## L2 Регуляризация (Ridge / Гребневая)

Штраф пропорционален **сумме квадратов** весов.

$$R_{L2}(\mathbf{w}) = \sum_{j=1}^{n} w_j^2$$

**Примечание:** Свободный член $w_0$ обычно **не регуляризуют**, т.к. он отвечает за общее смещение модели, а не за ее сложность взаимодействия с признаками. **Эффект:** Заставляет веса быть **маленькими**, но редко обнуляет их полностью. Уменьшает веса пропорционально их величине. Хорошо работает почти всегда. *Аналогия: Родитель, который немного урезает карманные расходы всем детям (весам), особенно тем, кто тратит больше.*

## L1 Регуляризация (Lasso)

Штраф пропорционален **сумме модулей** весов.

$$R_{L1}(\mathbf{w}) = \sum_{j=1}^{n} |w_j|$$

**Примечание:** Свободный член $w_0$ обычно **не регуляризуют** по той же причине, что и в L2. **Эффект:** Может **обнулять** некоторые веса, эффективно производя **отбор признаков** (*feature selection*). Полезна, когда есть подозрение, что многие признаки неинформативны. *Аналогия: Строгий родитель, который лишает карманных денег (обнуляет вес) некоторых детей (признаков) за провинности, а остальным может тоже немного урезать.*

## Связь с Bias-Variance Trade-off

Регуляризация - это инструмент управления компромиссом между смещением (bias) и разбросом (variance):

- **Без регуляризации ($\lambda = 0$):** Модель может иметь низкое смещение (хорошо подходит к обучающим данным), но высокий разброс (сильно меняется при изменении данных, переобучается).
- **С регуляризацией ($\lambda > 0$):** Добавляя штраф, мы **увеличиваем смещение** (модель становится "проще", может хуже подходить к обучающим данным), но **уменьшаем разброс** (модель становится стабильнее, лучше обобщается на новые данные).

Подбор оптимального $\lambda$ (через кросс-валидацию) позволяет найти баланс.