Шпаргалка по линейным моделям / Концепции Cheatsheet (XeLaTeX)

Краткий справочник

Содержание

1 Линейная Регрессия (`Linear Regression`)

Задача

Предсказание **непрерывного** числового значения (целевой переменной y) на основе одного или нескольких признаков (X).
Примеры: прогнозирование цены дома, температуры, спроса.

Модель Линейной Регрессии

Предполагается линейная зависимость между признаками и целевой переменной. Предсказание модели ($\hat{y}$) вычисляется как взвешенная сумма признаков объекта (**x**) с добавлением свободного члена (w_0).

$$\hat{y} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^T\mathbf{x}$$

- $\mathbf{w} = (w_0, w_1, \ldots, w_n)$: Вектор **весов** (параметров) модели, которые нужно найти в процессе обучения. w_0 называется **свободным членом** или **bias term**.
- $\mathbf{x} = (1, x_1, \ldots, x_n)$: Вектор **признаков** объекта, дополненный фиктивным признаком $x_0 = 1$ для удобства матричной записи.

Интерпретация Весов

Коэффициент w_j при признаке x_j показывает, на сколько *в среднем* изменится предсказание $\hat{y}$, если значение признака x_j увеличить на единицу, при условии, что все остальные признаки остаются неизменными (*ceteris paribus*).

Функция Потерь: MSE (`Mean Squared Error`)

Для оценки качества модели и её обучения используется **Среднеквадратичная Ошибка (MSE)**. Она измеряет средний квадрат разности между реальными значениями (y_i) и предсказаниями модели ($\hat{y}_i$) по всем m объектам обучающей выборки.

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

Цель обучения: Найти такие веса **w**, которые минимизируют значение MSE. **Преимущества MSE:**
- Штрафует большие ошибки сильнее, чем маленькие.
- Функция выпуклая и дифференцируемая, что удобно для оптимизации (например, градиентным спуском).

1.1. Обучение Модели

Идея Градиентного Спуска (GD)

Градиентный спуск — это итеративный алгоритм оптимизации для поиска минимума функции потерь (например, MSE). **Метафора:** Представьте MSE как рельеф местности. Цель — найти самую низкую точку. GD делает это, двигаясь шагами в направлении, противоположном самому крутому склону (антиградиенту).

1. Вычислить **градиент** функции потерь ($\nabla_{\mathbf{w}}MSE$) — вектор, указывающий направление наискорейшего роста функции.
2. Сделать шаг в **противоположном** направлении (антиградиент): $\mathbf{w} := \mathbf{w} - \alpha\nabla_{\mathbf{w}}MSE(\mathbf{w})$.
3. α — **скорость обучения** (*learning rate*), гиперпараметр, контролирующий размер шага.
4. Повторять шаги 1-3 до сходимости (пока веса не перестанут значительно изменяться или не будет достигнуто макс. число итераций).

Варианты Градиентного Спуска

Различаются тем, на каком объеме данных вычисляется градиент на каждом шаге:

- **Batch GD (Пакетный):** Градиент считается по **всей** обучающей выборке. Точные шаги к минимуму, но может быть очень медленным на больших датасетах.
- **Stochastic GD (Стохастический, SGD):** Градиент считается по **одному случайно выбранному примеру**. Шаги быстрые, но "шумные", траектория может колебаться вокруг минимума. Подходит для очень больших датасетов и онлайн-обучения.
- **Mini-batch GD (Мини-пакетный):** Градиент считается по небольшой **случайной подвыборке** (пакету, *batch*) данных (например, 32-512 примеров). Золотая середина: сочетает скорость SGD и стабильность Batch GD. Наиболее часто используемый вариант на практике.

Аналитическое Решение (Normal Equation)

Для задачи минимизации MSE в линейной регрессии существует **точное аналитическое решение**, позволяющее найти оптимальные веса $\mathbf{w}^*$ без итераций градиентного спуска.

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

- **X**: Матрица признаков (размер $m \times (n+1)$), где строки — объекты, столбцы — признаки (включая столбец единиц для w_0).
- **y**: Вектор целевых значений (размер $m \times 1$).

Преимущества:
- Находит точный минимум MSE за один шаг.
- Не требует подбора гиперпараметра learning rate (α).

Недостатки:
- Вычисление обратной матрицы $(\mathbf{X}^T\mathbf{X})^{-1}$ имеет сложность $O(n^3)$, где n — число признаков. Становится непрактичным при очень большом количестве признаков (например, $n > 10,000$).
- Матрица $\mathbf{X}^T\mathbf{X}$ может быть **вырожденной** (сингулярной), если признаки линейно зависимы (мультиколлинеарность) или если число признаков n больше числа объектов m. В этом случае обратной матрицы не существует.

1.2. Предположения и Расширения

Основные Предположения Классической Линейной Регрессии

Для корректной интерпретации результатов и построения статистических выводов (например, доверительных интервалов для весов) модель опирается на ряд предположений относительно данных и ошибок модели $\epsilon_i = y_i - \hat{y}_i$:

1. **Линейность:** Среднее значение целевой переменной y линейно зависит от признаков X.
2. **Независимость ошибок:** Ошибки ϵ_i для разных наблюдений независимы друг от друга.
3. **Гомоскедастичность:** Ошибки ϵ_i имеют одинаковую постоянную дисперсию ($\text{Var}(\epsilon_i) = \sigma^2$) для всех уровней признаков X. (Отсутствие гомоскедастичности называется *гетероскедастичностью*).
4. **Нормальность ошибок:** Ошибки ϵ_i распределены нормально с нулевым средним: $\epsilon_i \sim N(0, \sigma^2)$. (Важно в основном для построения доверительных интервалов и проверки гипотез).
5. **Отсутствие сильной мультиколлинеарности:** Признаки x_j не должны быть сильно линейно зависимы друг от друга.

Примечание: Нарушение этих предположений не всегда делает предсказания модели плохими, но может сделать недействительными статистические выводы о параметрах и их значимости.

Полиномиальная Регрессия

Стандартная линейная регрессия моделирует только линейные зависимости. Для моделирования нелинейных связей можно использовать **полиномиальную регрессию**. **Идея:** Создать новые признаки, являющиеся степенями исходных признаков ($x^2, x^3, \dots$) или их произведениями ($x_1x_2, \dots$). Затем применить обычную линейную регрессию к этому расширенному набору признаков. Пример для одного признака x:

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$$

Важно: Несмотря на нелинейную зависимость от x, модель остается **линейной по параметрам w**, поэтому все методы обучения линейной регрессии применимы. **Риск:** Легко приводит к **переобучению** при высокой степени полинома d. Обычно требует **регуляризации**.

2 Логистическая Регрессия (`Logistic Regression`)

Задача

Модель для задач **бинарной классификации**, когда целевая переменная y принимает одно из двух значений (например, 0 или 1, "да"/"нет", "спам"/"не спам").

Основная Идея

Логистическая регрессия напрямую предсказывает не сам класс, а **вероятность** принадлежности объекта к классу 1, $P(y = 1|\mathbf{x})$.

- Сначала вычисляется линейная комбинация признаков: $z = \mathbf{w}^T\mathbf{x}$.
- Затем результат z (который может быть любым числом) преобразуется в вероятность (число от 0 до 1) с помощью **сигмоидной (логистической) функции** $\sigma(z)$.

Сигмоидная Функция

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

Свойства:

- Принимает значения строго между 0 и 1.
- Если $z \to +\infty$, то $\sigma(z) \to 1$.
- Если $z \to -\infty$, то $\sigma(z) \to 0$.
- Если $z = 0$, то $\sigma(z) = 0.5$.

Предсказание вероятности: $\hat{p} = P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$.

Принятие Решения о Классе

Получив предсказанную вероятность $\hat{p}$, решение о классе принимается сравнением с **порогом** (threshold), обычно равным 0.5:

$$\hat{y} = \begin{cases} 1, & \text{если } \hat{p} \geq 0.5 \\ 0, & \text{если } \hat{p} < 0.5 \end{cases}$$

Порог можно настраивать в зависимости от задачи (например, если важнее минимизировать ложноотрицательные срабатывания).

Функция Потерь: LogLoss (Бинарная Кросс-Энтропия)

MSE не подходит для задач классификации с вероятностными выходами. Вместо неё используется **LogLoss** или **Бинарная Кросс-Энтропия**.

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

Где y_i — истинный класс (0 или 1), $\hat{p}_i = \sigma(\mathbf{w}^T\mathbf{x}_i)$ — предсказанная вероятность класса 1. **Интуиция:**

- Если истинный класс $y_i = 1$, член $(1 - y_i)\log(1 - \hat{p}_i)$ об-

нуляется, и штраф равен $-\log(\hat{p}_i)$. Штраф тем больше, чем меньше предсказанная вероятность $\hat{p}_i$.
- Если истинный класс $y_i = 0$, член $y_i \log(\hat{p}_i)$ обнуляется, и штраф равен $-\log(1 - \hat{p}_i)$. Штраф тем больше, чем больше предсказанная вероятность $\hat{p}_i$ (т.е., чем меньше $1 - \hat{p}_i$).

Модель сильно штрафуется за уверенные, но неверные предсказания. Обучение также проводится с помощью вариантов градиентного спуска для минимизации LogLoss.

Мультиклассовая Классификация: Softmax Regression

Для задач с $K > 2$ классами используется обобщение логистической регрессии — **Softmax Regression** (или Multinomial Logistic Regression).
- Для каждого класса $k \in \{1, \ldots, K\}$ вычисляется своя линейная комбинация ("оценка"): $z_k = \mathbf{w}_k^T \mathbf{x}$.
- Эти оценки преобразуются в вектор вероятностей $\hat{\mathbf{p}}$ с помощью **функции Softmax**:

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

- Функция Softmax гарантирует, что $\hat{p}_k \in (0, 1)$ и $\sum_{k=1}^{K} \hat{p}_k = 1$.
- Предсказанным классом обычно является тот, для которого вероятность $\hat{p}_k$ максимальна.
- Функция потерь — **Cross-Entropy Loss** (обобщение LogLoss на мультиклассовый случай).

3 Регуляризация L1 и L2

Проблема: Переобучение (Overfitting)

Сложные модели (например, полиномиальная регрессия высокой степени или модели с большим количеством признаков) могут **переобучаться** — идеально подгоняться под обучающие данные, включая шум, но плохо обобщаться на новые, невиданные данные. Визуально это проявляется в очень низких потерях на обучении и высоких на валидации/тесте. Одним из признаков переобучения часто являются *слишком большие значения весов* **w**.

Идея Регуляризации

Регуляризация — это метод борьбы с переобучением путем добавления **штрафа** к функции потерь модели за величину её весов. Это заставляет модель искать баланс между минимизацией исходной ошибки (MSE или LogLoss) и поддержанием весов небольшими. Общая формула регуляризованной функции потерь:

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

- $J_{original}(\mathbf{w})$: Исходная функция потерь (MSE для регрессии, LogLoss для классификации).
- $R(\mathbf{w})$: **Регуляризационный член**, зависящий от весов модели.
- $\lambda \geq 0$: **Коэффициент регуляризации** (гиперпараметр). Контролирует силу штрафа.
  - $\lambda = 0$: Нет регуляризации.
  - $\lambda \to \infty$: Веса стремятся к нулю, модель становится очень простой (может привести к недообучению).

Оптимальное значение λ подбирается с помощью кросс-валидации.

Важность Масштабирования Признаков

Перед применением регуляризации **крайне рекомендуется масштабировать** (привести к одному масштабу) признаки X. Популярные методы:
- **Стандартизация (StandardScaler):** Приведение к нулевому среднему и единичной дисперсии.
- **Нормализация (MinMaxScaler):** Приведение к диапазону [0, 1] или [-1, 1].

Зачем? Регуляризационный штраф $R(\mathbf{w})$ зависит от абсолютных значений весов. Если признаки имеют разный масштаб (например, возраст в годах и доход в рублях), то веса при признаках с большим масштабом будут искусственно занижаться, а при признаках с малым — завышаться, что исказит эффект регуляризации. **Примечание:** Свободный член w_0 обычно **не регуляризуют**, так как он отвечает за общее смещение модели, а не за взаимодействие с признаками.

L2 Регуляризация (Ridge / Гребневая Регрессия)

Использует в качестве штрафа **сумму квадратов** весов (L2-норму вектора весов).

$$R_{L2}(\mathbf{w}) = \sum_{j=1}^{n} w_j^2 = ||\mathbf{w}||_2^2$$

(Сумма идет от $j = 1$, так как w_0 не регуляризуется). **Функция потерь (пример для MSE):**

$$J_{Ridge}(\mathbf{w}) = MSE(\mathbf{w}) + \lambda \sum_{j=1}^{n} w_j^2$$

Эффект:
- "Сжимает" веса **w**, делая их значения **меньше**.
- Уменьшает веса пропорционально их величине (большие веса штрафуются сильнее).
- Редко приводит к обнулению весов (веса становятся маленькими, но не нулевыми).
- Делает решение более устойчивым, особенно при наличии мультиколлинеарности.
- Является хорошим выбором по умолчанию.

L1 Регуляризация (Lasso / Least Absolute Shrinkage and Selection Operator)

Использует в качестве штрафа **сумму модулей** весов (L1-норму вектора весов).

$$R_{L1}(\mathbf{w}) = \sum_{j=1}^{n} |w_j| = ||\mathbf{w}||_1$$

(Сумма идет от $j = 1$, так как w_0 не регуляризуется). **Функция потерь (пример для MSE):**

$$J_{Lasso}(\mathbf{w}) = MSE(\mathbf{w}) + \lambda \sum_{j=1}^{n} |w_j|$$

Эффект:

- Также "сжимает" веса **w** к нулю.
- Из-за использования модуля, может **обнулять** некоторые веса (при достаточно большом λ).
- Эффективно производит **отбор признаков** (*feature selection*), оставляя только наиболее важные.
- Полезна, когда есть основания полагать, что многие признаки являются неинформативными.

Примечание: Функция потерь с L1-штрафом не дифференцируема в точке 0, что требует использования специальных методов оптимизации (например, Subgradient descent или Proximal gradient descent).

Elastic Net

Комбинация L1 и L2 регуляризации. Штраф является взвешенной суммой L1 и L2 норм.

$$R_{ElasticNet}(\mathbf{w}) = \alpha \sum_{j=1}^{n} |w_j| + \frac{1-\alpha}{2} \sum_{j=1}^{n} w_j^2$$

(Здесь α - это *другой* гиперпараметр, смешивающий L1 и L2, часто обозначаемый как 'l1_ratio', не путать со скоростью обучения. Общий коэффициент регуляризации λ также присутствует). Сочетает преимущества обоих подходов: отбирает признаки (как Lasso) и стабилизирует решение при коррелирующих признаках (как Ridge).

Регуляризация и Дилемма Смещения-Разброса

Регуляризация является явным способом управления компромиссом Bias-Variance:

- **Увеличение λ (усиление регуляризации):**
  - Увеличивает **смещение (Bias)**: Модель становится проще, может хуже описывать обучающие данные.
  - Уменьшает **разброс (Variance)**: Модель становится менее чувствительной к шуму в данных и лучше обобщается.
- **Уменьшение λ (ослабление регуляризации):**
  - Уменьшает **смещение (Bias)**: Модель становится сложнее, лучше подгоняется под обучающие данные.
  - Увеличивает **разброс (Variance)**: Повышается риск переобучения.

Цель — найти такое λ, которое минимизирует ошибку на *новых* (валидационных) данных, достигая оптимального баланса между смещением и разбросом.