Машинное обучение

Разинков Е.В.

25 апреля 2015 г.

Оглавление

Введение в машинное обучение

Определение 1 (Tom Mitchell, "Machine Learning 1997). *Говорят, что компьютерная программа обучается на основе опыта E по отношению к некоторому классу задач T и некоторой мере эффективности P, если эффективность программы при решении задач из T, измеряемая с помощью P, повышается с опытом E.*

Краткая история развития машинного обучения:

- 1940-1950: Зарождение
  - Alan Turing, Computing Machinery and Intelligence, 1950.
- 1950-1970: Воодушевление
  - Minky, McCarthy, Shannon, Rochester: Dartmouth Conference, 1956
  - Программа, играющая в шашки
  - Компьютер, осуществляющий перевод с русского на английский
- 1970-1990: Системы, основанные на знаниях (Knowledge-Based Systems)
  - 1969-1979: Зарождение систем, основанных на знаниях
  - 1980-1988: Расцвет систем, основанных на знаниях
  - 1988-1993: Закат систем, основанных на знаниях
- 1990 - Настоящее время: Статистические подходы
  - Принятие неопределенности
  - Значительное увеличение вычислительных ресурсов

Классы алгоритмов машинного обучения:

- Обучение с учителем (Supervised learning)
  - Ассоциативное обучение
  - Классификация
    - Классификация и предсказание
    - Распознавание образов:
      - Распознавание печатного и рукописного текста
      - Обнаружение и распознавание лиц
      - Постановка медицинских диагнозов
      - Распознавание речи
      - Спам-фильтры
      - Машинный перевод
      - Биометрическая аутентификация: физиологические и поведенческие признаки
    - Извлечение знаний
    - Сжатие информации
    - Системы обнаружения вторжений, антивирусные техники
  - Регрессия

- Обучение без учителя (Unsupervised learning)
  - Алгоритмы кластеризации
    * Сегментирование клиентской базы
    * Сжатие изображений
    * Кластеризация документов
    * Задачи биоинформатики
- Обучение с подкреплением (Reinforcement learning)
  - Игры
    * Шашки: решены
    * Шахматы: в 1997 Каспаров проиграл компьютеру Deep Blue.
    * Го: скоро.
  - Робототехника

Регрессия

Определение 2. *Задача предсказания значения одной или более целевой переменной (*англ. targer variable*) на основе D-мерного вектора x входных переменных (*англ. input variables*) называется задачей регрессии (*англ. regression*).*

Пусть обучающая выборка (*англ. training set, training data set*) состоит из N наблюдений $\{\mathbf{x}_n\}$, где $n = 1, ..., N$, и соответствующих целевых значений $\{t_n\}$. Задача регрессии – предсказать значение t для нового вектора $\mathbf{x}$. Решение задачи заключается в построении модели – условного распределения $p(t|\mathbf{x})$.

Линейные модели регрессии

Предположим, что значение целевой переменной t для входного вектора $\mathbf{x}$ определяется значением детерминированной функции $y(\mathbf{x}, \mathbf{w})$ с аддитивным гауссовым шумом:

$$t = y(\mathbf{x}, \mathbf{w}) + \varepsilon,$$

где $\mathbf{w}$ – параметры модели, а ε – нормально распределенная случайная величина с нулевым средним и дисперсией σ^2.
Тогда

$$p(t|\mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \sigma^2). \tag{1}$$

Линейная регрессия

Линейной регрессией (*англ. linear regression*) называется простейшая линейная модель:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + ... + w_D x_D,$$

где $\mathbf{x} = (x_1, ..., x_D)^T$. Функция y линейна как относительно параметров модели $w_0, ..., w_D$, так и относительно входных переменных $x_1, ..., x_D$.

Базисные функции в линейных моделях

Интерес представляет более широкий класс моделей, в которых рассматривается линейная комбинация нелинейных функций $\phi_j(\mathbf{x})$ от входных переменных:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}). \tag{2}$$

Функции $\phi_j(\mathbf{x})$ называются *базисными функциями* (*англ. basis functions*).
Удобно определить базисную функцию $\phi_0(\mathbf{x}) = 1$, тогда (2) можно переписать следующим образом:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}),$$

где $\mathbf{w} = (w_0, ..., w_{M-1})^T$ и $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x}))^T$.
Примеры:

- Линейная регрессия: $\phi_1(\mathbf{x}) = x_1, \phi_2(\mathbf{x}) = x_2, ..., \phi_D(\mathbf{x}) = x_D$
- Полиномиальная регрессия от одной переменной: $\phi_1(x) = x, \phi_2(x) = x^2, ..., \phi_K(x) = x^K$.

Определение 3. *Матрицей плана (*англ. design matrix*) называется матрица* $\mathbf{\Phi}$*, имеющая следующий вид:*

$$\mathbf{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

Вычисление параметров модели

Пусть обучающая выборка состоит из множества входных векторов $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ и соответствующих этим векторам целевых переменных $t_1, ..., t_N$. Объединим целевые переменные $t_1, ..., t_N$ в вектор

$$\mathbf{t} = (t_1, ..., t_N)^T.$$

Предполагается, что все элементы обучающей выборки независимо получены из распределения (1), тогда

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n), \sigma^2). \tag{3}$$

Напомним, что функция плотности нормального распределения со средним μ и дисперсией σ^2 имеет следующий вид:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Запишем логарифм функции правдоподобия (3) по отношению к $\mathbf{w}$:

$$\ln p(\mathbf{t}|\mathbf{X}.\mathbf{w}, \sigma^2) = \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n), \sigma^2) = \frac{N}{2}\ln\frac{1}{\sigma^2} - \frac{N}{2}\ln(2\pi) - \frac{1}{\sigma^2}E_D(\mathbf{w}), \tag{4}$$

где

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)^2. \tag{5}$$

Таким образом, максимизация (4) сводится к минимизации (5) отнсительно $\mathbf{w}$. Вычислим градиент (5):

$$\nabla E_D(\mathbf{w})^T = -\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)\boldsymbol{\phi}(\mathbf{x}_n)^T$$

и приравняем к нулевому вектору:

$$\mathbf{0}^T = -\sum_{n=1}^{N} t_n\boldsymbol{\phi}(\mathbf{x}_n)^T + \mathbf{w}^T\left(\sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^T\right) = -\mathbf{t}^T\mathbf{\Phi} + \mathbf{w}^T(\mathbf{\Phi}^T\mathbf{\Phi}),$$

откуда получаем

$$\mathbf{w} = \left(\mathbf{t}^T\mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\right)^T = \left((\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\right)^T\left(\mathbf{t}^T\mathbf{\Phi}\right)^T = \left((\mathbf{\Phi}^T\mathbf{\Phi})^T\right)^{-1}\mathbf{\Phi}^T\mathbf{t} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}.$$

Определение 4. *Матрица* $\mathbf{\Phi}^+ = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T$ *называется псевдообратной матрицей Мура-Пенроуза для матрицы* $\mathbf{\Phi}$ *(*англ. Moore-Penrose pseudoinverse matrix*).*

Альтернативными способами минимизации (5) являются метод градиентного спуска (*англ. gradient descent*) и метод покоординатного спуска.

Регуляризация

Регуляризация (*англ. regularization*) – способ борьбы с оверфиттингом, когда вектор входных переменных имеет высокую размерность, а обучающая выборка невелика. Регуляризация заключается в добавлении еще одного слагаемого в целевую функцию E:

$$E_r = E_D + \frac{\lambda}{2}\sum_{i=0}^{M-1}|w_i|^q,$$

где λ – коэффициент регуляризации (*англ. regularization coefficient*). Выбор показателя степени q зависит от особенностей решаемой задачи, обычно используется значение $q = 2$. Рассмотрим этот случай подробнее:

$$E_r = E_D + \frac{\lambda}{2}\sum_{i=0}^{M-1}w_i^2 = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n})\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}.$$

Вычислим градиент

$$\nabla E_r^T = -\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n})\right)\boldsymbol{\phi}(\mathbf{x}_n)^T + \lambda\mathbf{w}^T$$

и приравняем к нулевому вектору:

$$\mathbf{0}^T = -\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n})\right)\boldsymbol{\phi}(\mathbf{x}_n)^T + \lambda\mathbf{w}^T = -\sum_{i=1}^{N}t_n\boldsymbol{\phi}(\mathbf{x}_n)^T + \mathbf{w}^T\sum_{i=1}^{N}\boldsymbol{\phi}(\mathbf{x_n})\boldsymbol{\phi}(\mathbf{x_n})^T + \lambda\mathbf{w}^T =$$
$$= -\mathbf{t}^T\boldsymbol{\Phi} + \mathbf{w}^T(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{I}),$$

откуда получаем:

$$\mathbf{w} = \left(\mathbf{t}^T\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\right)^T = \left((\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\right)^T\left(\mathbf{t}^T\boldsymbol{\Phi}\right)^T = \left(\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^T + \lambda\mathbf{I}^T\right)^{-1}\boldsymbol{\Phi}^T\mathbf{t} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^T\mathbf{t},$$

где $\mathbf{I}$ – единичная матрица $M \times M$.

Кросс-валидация

Кросс-валидация (*англ. cross-validation*) – процесс выбора модели на основе независимого набора данных, называемого кросс-валидационной выборкой (*англ. cross-validation set*).
В случае применения линейной модели для решения регрессионной задачи под выбором модели понимается определение набора базисных функций и значения коэффициента регуляризации.
Таким образом, имеющееся множество данных разбивается на три непересекающихся множества:

- Обучающая выборка (*англ. training set*).
- Кросс-валидационная выборка (*англ. cross-validation set*).
- Тестовая выборка (*англ. testing set*).

Процесс построения модели состоит из трех этапов:

- *Обучение.* В процессе обучения для каждой модели параметры модели оцениваются на основе обучающей выборки.
- *Кросс-валидация.* Выбирают модель, которой соответствует наименьшее значение функции ошибки на кросс-валидационной выборке.
- *Тестирование.* Точность выбранной модели оценивается на тестовой выборке.

Классификация

Пусть заданы K классов: $C_0, C_1, ..., C_{K-1}$. Пусть U – множество объектов, каждый из которых принадлежит одному из классов $C_0, C_1, ..., C_{K-1}$. Функция

$$\mu : U \to \mathbb{R}^D$$

вычисляет на основе объекта u вектор характеристик размерности D.

Определение 5. *Задача определения, какому из K классов $C_0, ..., C_{K-1}$ принадлежит объект $u \in U$ на основе его характеристик, представленных D-мерным вектором x входных переменных (*англ. input variables*), называется задачей классификации (*англ. classification*).*

Пусть обучающая выборка построена на основе данных N объектов $u_1, ..., u_N \in U$ и состоит из N наблюдений $\{\mathbf{x}_n\}$, где $\mathbf{x_n} = \mu(u_n)$, $n = 1, ..., N$. Для каждого входного вектора $\mathbf{x}_n = \mu(u_n)$ в обучающей выборке хранится номер $t_n \in \{0, ..., K-1\}$ класса, которому принадлежит объект u_n. Задача – по вектору характеристик $\mathbf{x}$ объекта u определить класс номер класса t, которому принадлежит объект u.

Подходы к решению задачи классификации

Решение задачи классификации часто можно разделить на два этапа:

- Вывод (*англ. inference stage*): вычисление условных вероятностей $p(u \in C_i|\mu(u) = \mathbf{x}), i = 0, ..., K-1$.
- Принятие решения (*англ. decision stage*): на основе условных вероятностей $p(u \in C_i|\mu(u) = \mathbf{x}), i = 0, ..., K-1$, принимается решение о принадлежности объекта u к тому или иному классу:

$$t = \arg\max_i p(u \in C_i|\mu(u) = \mathbf{x}).$$

Существует три основных подхода к решению задачи классификации:

- Генеративные модели (*англ. generative models*).
  - Определение функций плотности условных вероятностей $p(\mu(u) = \mathbf{x}|u \in C_i)$ и вероятностей $p(u \in C_i)$ для каждого класса $C_i, i = 0, ..., K-1$.
  - Применение теоремы Байеса:

$$p(u \in C_i|\mu(u) = \mathbf{x}) = \frac{p(\mu(u) = \mathbf{x}|u \in C_i)p(u \in C_i)}{p(\mu(u) = \mathbf{x})},$$

    где функция плотности $p(\mu(u) = \mathbf{x})$ может быть найдена по формуле:

$$p(\mu(u) = \mathbf{x}) = \sum_{i=0}^{K} p(\mu(u) = \mathbf{x}|u \in C_i)p(u \in C_i).$$

  - Принятие решения:

$$t = \arg\max_i p(u \in C_i|\mu(u) = \mathbf{x}).$$

- Условные модели (*англ. conditional models, discriminative models*).
  - Функции плотности условных вероятностей $p(u \in C_i|\mu(u) = \mathbf{x}), i = 0, ..., K-1$ оцениваются непосредственно.

  - Принятие решения:
$$t = \arg\max_i p(u \in C_i | \mu(u) = \mathbf{x}).$$

- Дискриминантные функции (*англ. discriminant functions*).
  - Дискриминантная функция представляет собой отображение
$$f(\mathbf{x}) \to \{0, ..., K-1\},$$
вероятности $p(u \in C_i | \mu(u) = \mathbf{x}), i = 0, ..., K-1$, не оцениваются.

Ошибки классификации

Рассмотрим задачу с двумя классами, C_0 и C_1, и алгоритм классификации $y(\mathbf{x}, \mathbf{w}) \to \{0, 1\}$. Пусть точность классификатора оценивается на основе выборки векторов $\mathbf{x}_1, ..., \mathbf{x}_N$, представляющий собой характеристики объектов $\mathbf{x}_i = \mu(u_i)$, $u_i \in C_{t_i}$.
Введем четыре показателя:

- $TruePositives = |\{u_i \in C_1 \mid y(\mathbf{x}_i, \mathbf{w}) = 1\}|$,
- $TrueNegatives = |\{u_i \in C_0 \mid y(\mathbf{x}_i, \mathbf{w}) = 0\}|$,
- $FalsePositives = |\{u_i \in C_0 \mid y(\mathbf{x}_i, \mathbf{w}) = 1\}|$,
- $FalseNegatives = |\{u_i \in C_1 \mid y(\mathbf{x}_i, \mathbf{w}) = 0\}|$.

Заметим, что
$$TruePositives + TrueNegatives + FalsePositives + FalseNegatives = N.$$

Количественные показатели, связанные с ошибками классификации:

- Ошибки первого и второго рода:
  - Ошибка первого рода, ложная положительная классификация, "ложная тревога"(*англ. Type I error, false alarm*):
$$\alpha = \frac{FalsePositives}{TrueNegatives + FalsePositives}$$
  - Ошибка второго рода, ложная отрицательная классификация, "пропуск цели"(*англ. Type I error, miss*):
$$\beta = \frac{FalseNegatives}{TruePositives + FalseNegatives}$$

- Accuracy (*достоверность*):
  - $Accuracy = \dfrac{TruePositives + TrueNegatives}{N}$

- Precision and recall (*точность и полнота*):
  - $Precision = \dfrac{TruePositives}{TruePositives + FalsePositives}$.
  - $Recall = \dfrac{TruePositives}{TruePositives + FalseNegatives}$.

- F_1-score:
  - $F_1 Score = 2\dfrac{Precision \cdot Recall}{Precision + Recall}$

Линейные модели классификации

Рассмотрим задачу классификации с двумя классами, C_0 и C_1, $t \in \{0,1\}$. Пусть u – классифицируемый объект, $\mathbf{x} = \mu(u)$ – вектор характеристик этого объекта, $\mathbf{w}$ – вектор параметров модели. В рамках линейной модели классификации рассматривается функция следующего вида:

$$t = y(\mathbf{x}, \mathbf{w}) = f(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})),$$

где $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x}))^T$ – вектор базисных функций (*см. "Линейные модели регрессии. Базисные функции"*). Функция f называется функцией активации (*англ. activation function*).

Логистическая регрессия

По теореме Байеса,

$$p(u \in C_1 | \mu(u) = \mathbf{x}) = \frac{p(\mu(u) = \mathbf{x} | u \in C_1) p(u \in C_1)}{p(\mu(u) = \mathbf{x})} =$$
$$= \frac{p(\mu(u) = \mathbf{x} | u \in C_1) p(u \in C_1)}{p(\mu(u) = \mathbf{x} | u \in C_0) p(u \in C_0) + p(\mu(u) = \mathbf{x} | u \in C_1) p(u \in C_1)} = \frac{1}{1 + \frac{p(\mu(u) = \mathbf{x} | u \in C_0) p(u \in C_0)}{p(\mu(u) = \mathbf{x} | u \in C_1) p(u \in C_1)}}.$$

Обозначим

$$a = \ln \frac{p(\mu(u) = \mathbf{x} | u \in C_1) p(u \in C_1)}{p(\mu(u) = \mathbf{x} | u \in C_0) p(u \in C_0)},$$

тогда

$$p(u \in C_1 | \mu(u) = \mathbf{x}) = \frac{1}{1 + e^{-a}}.$$

Функция

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

называется *логистической функцией* или *сигмоидом* (*англ. logistic sigmoid*) и используется в логистической регрессии в качестве функции активации.
Таким образом, в рамках логистической регрессии модель имеет следующий вид:

$$y(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})}},$$

где $\mathbf{x}$ – входной вектор, $\mathbf{w}$ – вектор параметров модели, а $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x}))^T$ – вектор базисных функций.

Вычисление параметров модели. Запишем функцию правдоподобия

$$p(\mathbf{t} | \mathbf{w}) = \prod_{n=1}^{N} y(\mathbf{x}_n, \mathbf{w})^{t_n} (1 - y(\mathbf{x}_n, \mathbf{w}))^{1 - t_n},$$

где $\mathbf{t} = (t_1, ..., t_N)^T$. Логарифмируем функцию правдоподобия и возьмем результат с противоположным знаком:

$$E(\mathbf{w}) = -\ln p(\mathbf{t} | \mathbf{w}) = -\sum_{n=1}^{N} (t_n \ln y(\mathbf{x}_n, \mathbf{w}) + (1 - t_n) \ln(1 - y(\mathbf{x}_n, \mathbf{w}))). \quad (6)$$

В качестве параметров модели выбирается вектор $\mathbf{w}$, доставляющий минимум функции (6). Для минимизации (6) часто используется метод градиентного спуска.

Вычислим градиент целевой функции (6):

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y(\mathbf{x}_n, \mathbf{w}) - t_n)\boldsymbol{\phi}(\mathbf{x}_n).$$

Для вычисления вектора $\mathbf{w}$ используется метод градиентного спуска.

Логистическая регрессия с регуляризацией

При использовании регуляризации целевая функция принимает следующий вид:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}(t_n \ln \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)) + (1 - t_n)\ln(1 - \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)))) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w},$$

где λ – коэффициент регуляризации.
Градиент этой функции:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)) - t_n)\boldsymbol{\phi}(\mathbf{x}_n) + \lambda\mathbf{w}.$$

Метод градиентного спуска

Метод градиентного спуска (*англ. gradient descent*) – итерационный алгоритм минимизации целевой функции $E(\mathbf{w})$, состоящий из следующих шагов:

- Выбирается случайное начальное приближение $\mathbf{w}_0$.
- Итеративный процесс:
  - Приближенное решение $\mathbf{w}_k$ на очередной итерации вычисляется как разность между полученным на предыдущей итерации приближенным решением $\mathbf{w}_{k-1}$ и вектором градиента $\nabla E(\mathbf{w_{k-1}})$, умноженным на коэффициент обучения γ:

    $$\mathbf{w}_k = \mathbf{w}_{k-1} - \gamma\nabla E(\mathbf{w}_{k-1}).$$

  - Итеративный процесс продолжается до тех пор, пока не будет выполнено следующее условие:

    $$||\mathbf{w}_k - \mathbf{w}_{k-1}|| < \varepsilon(||\mathbf{w}_k|| + \varepsilon_0)$$

    для некоторых малых положительных $\varepsilon, \varepsilon_0$.
- Результат: последнее полученное приближенное решение $\mathbf{w}_k$.

Параметры метода градиентного спуска для линейной модели регрессии

Параметры:

- Обучающая выборка: матрица плана $\mathbf{\Phi}$, вектор целевых переменных $\mathbf{t}$.
- Коэффициент регуляризации λ, коэффициент обучения γ, погрешность ε.

Целевая функция:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}.$$

Градиент целевой функкии:

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N}\left(t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x_n})\right)\boldsymbol{\phi}(\mathbf{x}_n) + \lambda\mathbf{w}.$$

Параметры метода градиентного спуска для логистической регрессии

Параметры:

- Обучающая выборка: матрица плана $\mathbf{\Phi}$, вектор целевых переменных $\mathbf{t}$.
- Коэффициент регуляризации λ, коэффициент обучения γ, погрешность ε.

Целевая функция:

$$E(\mathbf{w}) = -\sum_{n=1}^{N}(t_n \ln\sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)) + (1-t_n)\ln(1-\sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)))) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}.$$

Градиент целевой функции:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)) - t_n)\boldsymbol{\phi}(\mathbf{x}_n) + \lambda\mathbf{w}.$$

Нейронные сети

Структура нейронной сети

Нейрон

Нейрон представляет собой функцию следующего вида:

$$z = f\left(\mathbf{w}^T\mathbf{z}\right),$$

где $\mathbf{z} = (1, z_1, ..., z_M)^T$, $z_1, ..., z_M$ – выходные значения нейронов предыдущего слоя, $\mathbf{w} = (w_0, w_1, ..., w_M)$ – веса (параметры модели), f – функция активации. Значение $a = \mathbf{w}^T\mathbf{z}$ называется активацией (*англ. activation*) этого нейрона.

Многослойный персептрон

Определение 6. *Нейронная сеть, состоящая из слоев нейронов, где каждый нейрон связан со всеми нейронами предыдущего слоя и всеми нейронами следующего слоя, называется многослойным персептроном (*англ. multilayer perceptron, MLP*).*

Каждый нейрон i-го слоя принимает на вход взвешенные выходные значения всех нейронов предыдущего слоя, выход этого нейрона с соответствующими весами подается на вход всем нейронам следующего слоя. Нейроны одного слоя не связаны между собой.
Входы нейронной сети образуют входной слой (*англ. input layer*). Нейроны, выходы которых являются выходами нейронной сети, образуют выходной слой (*англ. output layer*). Слои, находящиеся между входным и выходным слоями, называются скрытыми (*англ. hidden layers*).

Нотация

Пусть нейронная сеть состоит из входного слоя, $L-1$ скрытых слоев и выходного слоя. Пусть l-й слой состоит из M_l нейронов. Во входной и каждый скрытый слой добавляется 0-й нейрон, который не имеет входов, постоянное выходное значение этого нейрона - единица.
Пусть $\mathbf{x} = (x_1, ..., x_D)$ – входной вектор. Выходные значения входного слоя обозначаются через $\mathbf{z}^{(0)}$:

$$\mathbf{z}^{(0)} = (1, x_1, ..., x_D).$$

Рассмотрим i-й нейрон l-го слоя. Выходное значение этого нейрона $z_i^{(l)}$ вычисляется следующим образом:

$$z_i^{(l)} = f\left(a_i^{(l)}\right) = f\left(\sum_{j=0}^{M_{l-1}} w_{ij}^{(l)} z_j^{(l-1)}\right),$$

где $a_i^{(l)}$ – активация этого нейрона, $w_{ij}^{(l)}$ – вес ребра, соединяющего j-й нейрон $(l-1)$-го слоя с i-м нейроном l-го слоя, $z_j^{(l-1)}$ – выходное значение j-го нейрона $(l-1)$-го слоя.
Вектор выходных значений выходного слоя (да, именно так), таким образом, обозначается через $\mathbf{z}^{(L)}$ и является выходом нейронной сети:

$$y(\mathbf{x}, \mathbf{W}) = \mathbf{z}^{(L)},$$

где $\mathbf{W} = (\mathbf{W}^{(1)}, ..., \mathbf{W}^{(L)})$ – параметры нейронной сети для скрытых и выходного слоев.
Для вычисления $y(\mathbf{x}, \mathbf{W})$ выходные значения нейронов вычисляются послойно.

Функции активации

Нейронная сеть может содержать нейроны с различными функциями активации. Функции активации нейронов выходного слоя выбираются в соответствии с решаемой задачей:

- Классификация:
$$f(a) = \frac{1}{1+e^{-a}}.$$
- Регрессия:
$$f(a) = a.$$

Целевые функции

Пусть нейронная сеть содержит M_L выходных нейронов, i-й выходной нейрон возвращает значение y_i. Нейронная сеть, таким образом, возвращает вектор

$$\mathbf{y}(\mathbf{x}, \mathbf{W}) = \{y_i\}_{i=1}^{M_L}$$

Вид целевой функции зависит от решаемой задачи.

Регрессия

Количество нейронов в выходном слое нейронной сети совпадает с количеством целевых переменных, i-й нейрон возвращает значение i-й целевой переменной:

$$t_i = y_i = \sum_{j=0}^{M_{L-1}} w_{ij}^{(L)} z_j^{(L-1)}.$$

Целевая функция имеет следующий вид:

$$E(\mathbf{W}) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}(\mathbf{x}_n, \mathbf{W}) - \mathbf{t}_n||^2.$$

Если целевая переменная одна, то выходной слой состоит из одного нейрона ($M_L = 1$) и нейронная сеть возвращает скалярное значение:

$$y(\mathbf{x}, \mathbf{W}) = \sum_{j=0}^{M_{L-1}} w_{ij}^{(L)} z_j^{(L-1)},$$

тогда целевая функция принимает вид:

$$E(\mathbf{W}) = \frac{1}{2}\sum_{n=1}^{N} (y(\mathbf{x}_n, \mathbf{W}) - t_n)^2.$$

Классификация

Если классов K, $K > 2$, выходной слой нейронной сети состоит из K нейронов, i-й нейрон возвращает уверенность классификатора в том, что объект принадлежит классу C_i:

$$y_i = \sigma\left(\sum_{j=0}^{M_{L-1}} w_{ij}^{(L)} z_j^{(L-1)}\right).$$

Пусть $u_n \in C_k$ – объект, вектор характеристик которого $\mathbf{x}_n = \mu(u_n)$ содержится в обучающей выборке. Вектор $\mathbf{t}_n$, описывающий принадлежность объекта u_n к классу C_k, имеет единицу на k-й позиции и нули на остальных позициях:

$$\mathbf{t}_n = (0, ..., 0, 1, 0, ..., 0)^T.$$

Целевая функция имеет следующий вид:

$$E(\mathbf{W}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} \left(t_{nk} \ln y_k(\mathbf{x}_n, \mathbf{W}) + (1 - t_{nk}) \ln(1 - y_k(\mathbf{x}_n, \mathbf{W}))\right),$$

где t_{nk} – значение k-й компоненты вектора $\mathbf{t}_n$.
Если класса два, C_0 и C_1, то выходной слой нейронной сети состоит из одного нейрона, возвращающего уверенность классификатора в том, что объект принадлежит классу C_1:

$$y(\mathbf{x}, \mathbf{W}) = \sigma\left(\sum_{j=0}^{M_{L-1}} w_{ij}^{(L)} z_j^{(L-1)}\right).$$

Таким образом, если $t_n \in \{0, 1\}$ – номер класса, которому принадлежит объект u_n, вектор характеристик которого $\mathbf{x}_n = \mu(u_n)$ является элементом обучающей выборки, целевая функция имеет следующий вид:

$$E(\mathbf{W}) = -\sum_{n=1}^{N} \left(t_n \ln y(\mathbf{x}_n, \mathbf{W}) + (1 - t_n) \ln(1 - y(\mathbf{x}_n, \mathbf{W}))\right).$$

Обучение нейронной сети

Стохастический градиентный спуск

Если целевая функция представляется в виде суммы функций, вычисленных для отдельных элементов обучающей выборки,

$$E(\mathbf{W}) = \sum_{i=1}^{N} E_i(\mathbf{W}),$$

возможным становится применение метода стохастического градиентного спуска (*англ. stochastic gradient descent, sequential gradient descent, on-line gradient descent*), когда изменение параметров модели на каждом шаге осуществляется на основе значения градиента, вычисленного для одного элемента обучающей выборки:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \gamma \nabla E_i(\mathbf{w}_{old})$$

Эта процедура повторяется для всех элементов обучающей выборки и, возможно, неоднократно.
Пример критерия выхода из итерационного процесса:

$$||\mathbf{w}_{sN} - \mathbf{w}_{(s-1)N}|| < \varepsilon(||\mathbf{w}_{sN}|| + \varepsilon_1),$$

где $\mathbf{w}_i$ – приближение после i-й итерации, $s \in \mathbb{N}$, $\varepsilon, \varepsilon_1 > 0$.
Метод стохастического градиентного спуска используется для обучения нейронной сети в рамках алгоритма обратного распространения ошибки.

Алгоритм обратного распространения ошибки

Рассмотрим целевую функцию следующего вида:

$$E(\mathbf{W}) = \sum_{n=1}^{N} E_n(\mathbf{W}).$$

Минимизируем эту функцию с помощью стохастического градиентного спуска, что подразумевает вычисление частных производных

$$\frac{\partial E_n}{\partial w_{ij}^{(l)}}.$$

Подадим на вход нейронной сети элемент обучающей выборки $\mathbf{x}_n$, вычислим активацию $a_i^{(l)}$ и выходное значение $z_i^{(l)}$ для каждого нейрона. На основе выходных значений $\mathbf{z}^{(L)}$ нейронов выходного слоя вычислим целевую функцию $E_n(\mathbf{W})$.
Заметим, что

$$\frac{\partial E_n}{\partial w_{ij}^{(l)}} = \frac{\partial E_n}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial w_{ij}^{(l)}}.$$

Введем обозначение:

$$\delta_i^{(l)} = \frac{\partial E_n}{\partial a_i^{(l)}}.$$

В силу того, что

$$\frac{\partial a_i^{(l)}}{\partial w_{ij}^{(l)}} = \frac{\partial \left(\sum_{j=0}^{M_{l-1}} w_{ij}^{(l)} z_j^{(l-1)} \right)}{\partial w_{ij}^{(l)}} = z_j^{(l-1)},$$

получим

$$\frac{\partial E_n}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} z_j^{(l-1)}.$$

Пусть вычислены $\delta_i^{(l)}$, $i = 1, ..., M_l$. Заметим, что

$$\delta_j^{(l-1)} = \frac{\partial E_n}{\partial a_j^{(l-1)}} = \sum_{i=1}^{M_l} \frac{\partial E_n}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial a_j^{(l-1)}} = \sum_{i=1}^{M_l} \delta_i^{(l)} \frac{\partial a_i^{(l)}}{\partial a_j^{(l-1)}}.$$

Так как

$$\frac{\partial a_i^{(l)}}{\partial a_j^{(l-1)}} = \frac{\partial \left(\sum_{m=0}^{M_{l-1}} w_{im}^{(l)} f\left(a_m^{(l-1)}\right)\right)}{\partial a_j^{(l-1)}} = w_{ij}^{(l)} f'\left(a_j^{(l-1)}\right),$$

истинно следующее равенство:

$$\delta_j^{(l-1)} = \sum_{i=1}^{M_l} \delta_i^{(l)} w_{ij}^{(l)} f'\left(a_j^{(l-1)}\right) = f'\left(a_j^{(l-1)}\right) \sum_{i=1}^{M_l} \delta_i^{(l)} w_{ij}^{(l)}.$$

Повторение этой процедуры для всех слоев нейронной сети позволяет эффективно вычислить частные производные

$$\frac{\partial E_n}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} z_j^{(l-1)}. \tag{7}$$

Начальные значения параметров нейронной сети выбираются случайным образом из отрезка $[-\varepsilon, \varepsilon]$ для некоторого малого $\varepsilon > 0$.

Регуляризация и нейронные сети

При построении нейронной сети с целевой функцией $E(\mathbf{W})$ возможно использование регуляризации в целях борьбы с оверфиттингом, в этом случае целевая функция принимает следующий вид:

$$E_r(\mathbf{W}) = E(\mathbf{W}) + \frac{\lambda}{2} \sum_{l=1}^{L} \sum_{i=1}^{M_l} \sum_{j=1}^{M_{l-1}} \left(w_{ij}^{(l)}\right)^2.$$

Тогда (7) принимает вид:

$$\frac{\partial E_n}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} z_j^{(l-1)} + \lambda w_{ij}^{(l)}.$$

Деревья решений

Определение 7. *Дерево – связный ориентированный граф без циклов, в котором каждая вершина имеет не более одного входящего ребра.*

Определение 8. *Дерево решений (*англ. decision tree, randomized tree, randomized decision tree*) – алгоритм машинного обучения с учителем, представляющий собой набор слабых классификаторов, организованных в иерархическую структуру – дерево.*

Дерево решений содержит узлы двух видов:

- Внутренний узел (*англ. split node, internal node, decision node, branch node*) – узел дерева, имеющий два или более потомка. В дереве решений нет узлов с одним потомком.

- Терминальный (листовой) узел (*англ. terminal node, leaf node*) – узел дерева, не имеющий потомков.

Определение 9. *Бинарным деревом решений называется дерево решений, каждый разделяющий узел которого имеет ровно два потомка.*

Деревья решений могут применяться для решения следующих задач:

- Классификация.
- Регрессия.
- Кластеризация.

Ниже рассматривается применение деревьев решений для классификации.

Нотация

Пусть имеется K классов: $C_1, C_2, ..., C_K$. Через $\mathbf{x} \in \mathbb{R}^D$ обозначим вектор входных переменных. Пусть обучающая выборка содержит N элементов, через S_i обозначим множество элементов обучающей выборки, достигших i-го узла при обучении, через S_{ij} – множество элементов обучающей выборки, достигших j-го узла, являющегося потомком i-го узла. Через $S^{(k)}$ обозначим множество элементов обучающей выборки, соответствующих объектам, принадлежащим классу C_k.
Функция выбора характеристик (*англ. feature selector function*) – это функция, получающая на вход вектор входных переменных $\mathbf{x}$ и возвращающая вектор, содержащий подмножество входных переменных вектора $\mathbf{x}$. Функция выбора характеристик, таким образом, имеет следующий вид:

$$\boldsymbol{\psi}(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^{D'},$$

где $D' \leq D$.
Введем обозначение:

$$[a] = \begin{cases} 1, & \text{если } a \text{ истинно,} \\ 0, & \text{если } a \text{ ложно.} \end{cases}$$

Внутренние узлы

Типы разделяющих функций

Разделяющая функция (*англ. split function*) – функция, определяющая следующий узел (разделяющую функцию) из m потомков j-го внутреннего узла для элемента $\mathbf{x}$:

$$h(\mathbf{x}, \boldsymbol{\theta}_j) : \mathbb{R}^D \times \mathcal{T} \to \{0, m-1\},$$

где $\boldsymbol{\theta}_j \in \mathcal{T}$ – вектор параметров разделяющей функции j-го узла.
Для бинарного дерева решений разделяющая функция имеет следующий вид:

$$h(\mathbf{x}, \boldsymbol{\theta}_j) : \mathbb{R}^D \times \mathcal{T} \to \{0, 1\}.$$

Для бинарного дерева решений обычно используется одна из нижеперечисленных разделяющих функций:

- Разделение гиперплоскостью, параллельной осям координат:

  $$h(\mathbf{x}, \boldsymbol{\theta}_j) = [\tau_1 > \psi(\mathbf{x}) > \tau_2],$$

  где $\boldsymbol{\theta}_j = (\psi, \tau_1, \tau_2)$, $\psi(\mathbf{x}) = x_i$ для некоторого i. Обычно $\tau_1 = \infty$ или $\tau_2 = -\infty$.

- Линейное разделение:
$$h(\mathbf{x}, \boldsymbol{\theta}_j) = \left[\tau_1 > \boldsymbol{\psi}(\mathbf{x})^T \mathbf{w} > \tau_2\right],$$
где $\boldsymbol{\theta}_j = (\boldsymbol{\psi}, \mathbf{w}, \tau_1, \tau_2)$.

- Нелинейное разделение:
$$h(\mathbf{x}, \boldsymbol{\theta}_j) = \left[\tau_1 > \boldsymbol{\varphi}(\boldsymbol{\psi}(\mathbf{x}))^T \mathbf{w} > \tau_2\right],$$
где $\boldsymbol{\theta}_j = (\boldsymbol{\psi}, \mathbf{w}, \tau_1, \tau_2)$, а $\boldsymbol{\varphi}$ – вектор базисных функций.

Критерии качества разделения

Для определения того, какую разделяющую функцию выбрать для данного узла, необходимо оценить качество разделения выборки для каждой из функций. Обычно в качестве критерия качества разделения используется прирост информации (*англ. information gain*):

$$I = H(S_i) - \sum_j \frac{N_{ij}}{N_i} H(S_{ij}),$$

где $H(S_i)$ – энтропия:

$$H(S_i) = -\sum_k \frac{N_i^{(k)}}{N_i} \log \frac{N_i^{(k)}}{N_i},$$

а $N_{ij} = |S_{ij}|$, $N_i = |S_i|$, $N_i^{(k)} = \left|S_i^{(k)}\right|$.
Вектор параметров $\boldsymbol{\theta}$ разделяющей функции выбирается следующим образом:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}} I.$$

Иногда используются другие критерии качества разделения.

Терминальные узлы

Критерии прекращения роста дерева

В зависимости от особенностей решаемой задачи в качестве критерия остановки роста дерева может быть выбрано достижение одного или нескольких из нижеперечисленных условий:

- Часть обучающей выборки, достигшей узла, достаточно чиста: энтропия ниже заданного значения H_{min}.
- Достигнута максимальная глубина дерева d.
- Количество элементов обучающей выборки, достигших узла при обучении дерева, меньше заданного значения n_{min}.

Если при создании очередного узла дерева выполняется одно из выбраных условий, созданный узел создается терминальным.

Создание терминального узла

При создании терминального узла определяется значение, которое дерево будет возвращать при достижении этого узла вектором входных переменных $\mathbf{x}$.
Как правило, в качестве возвращаемого значения i-го терминального узла используется вектор $\mathbf{t} = \{t\}_{k=0}^{K-1}$, где

$$t_k = \frac{N_i^{(k)}}{N_i}.$$

Таким образом, значение t_k можно интерпретировать как уверенность дерева решений в том, что объект, которому соответствует вектор входных переменных $\mathbf{x}$, принадлежит классу C_k.

Random Forest

Определение 10. *Random Forest (decision forest) – алгоритм машинного обучения с учителем, представляющий собой набор деревьев решений.*

Все деревья решений должны быть разными.
Значение, возвращаемое классификатором Random Forest, содержащим n деревьев решений, вычисляется следующим образом:

$$\bar{\mathbf{t}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{t}^{(i)},$$

где $\mathbf{t^{(i)}}$ – вектор, возвращаемый i-м деревом решений.
Значение $\bar{t}_k$ можно интерпретировать как уверенность классификатора Random Forest в том, что объект, которому соответствует вектор входных переменных $\mathbf{x}$, принадлежит классу C_k.

Как вырастить деревья разными?

Бэггинг

Определение 11. *Бэггинг (*англ. bagging – bootstrap aggregating*) – метод генерации различных деревьев решений, заключающийся в обучении каждого дерева решений на случайном подмножестве элементов обучающей выборки.*

Применение этого метода не рекомендуется.

Рандомизированная оптимизация узлов

Выбирается значение $r \in \{1, ..., |\mathcal{T}|\}$, характеризующее случайность при обучении дерева решений.
Выбор параметров разделяющей функции в каждом узле осуществляется следующим образом:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}\in\mathcal{T}_i} I,$$

где $\mathcal{T}_i \in \mathcal{T}$ – случайно выбранное подмножество множества параметров, $|\mathcal{T}_i| = r$.