AI Community

4. Logistic Regression and Support Vector Machines

План на сегодня

1. Логистическая регрессия
    a. Разделяющая гиперплоскость
    b. Сигмоида
    c. Binary Log Loss
    d. Градиентный спуск
2. Метод Опорных Векторов (Support Vector Machines)
    a. Лучшая разделяющая гиперплоскость
    b. Hinge Loss
    c. Регуляризация
3. Softmax
    a. Log Loss

Логистическая регрессия

Формулировка задачи

Мы хотим построить модель

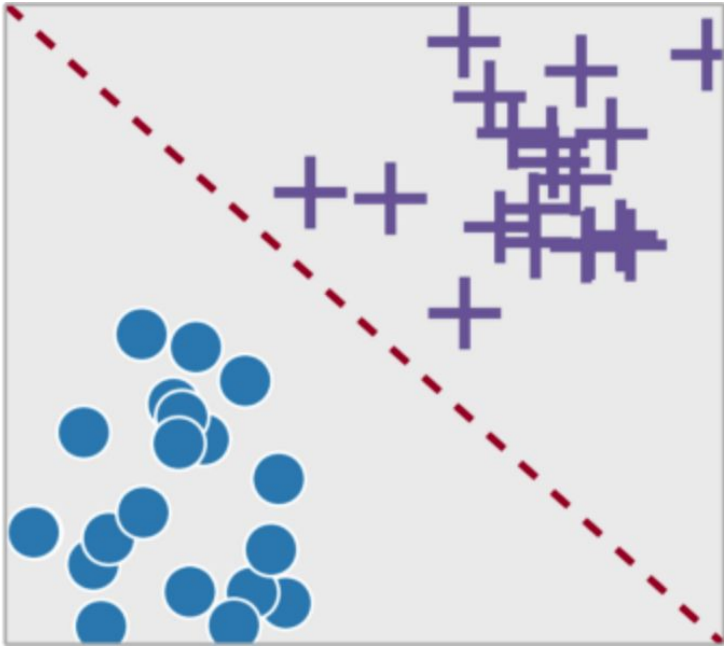
$$f(x) : R^n \to \{0, 1\}$$

По нашим тренировочным данным

$$(X_1, y_1), ..., (X_N, y_N) \in R^n \times \{0, 1\}$$

Логистическая регрессия

Логистическая регрессия -
это классификационная модель,
используемая для прогнозирования
вероятности бинарного события (1 или 0).



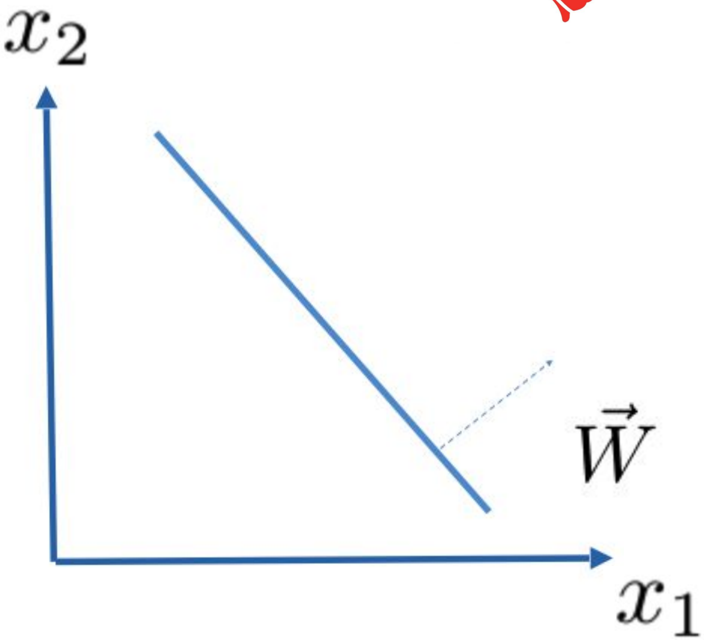
Разделяющая гиперплоскость

Уравнение плоскости задается так:

$$W_1 x_1 + \ldots + W_n x_n + b = W^T x + b = 0$$

$$W, x \in R^n \qquad b \in R$$

Если b = 0, то гиперплоскость будет проходить через центр координат $\vec{0}$

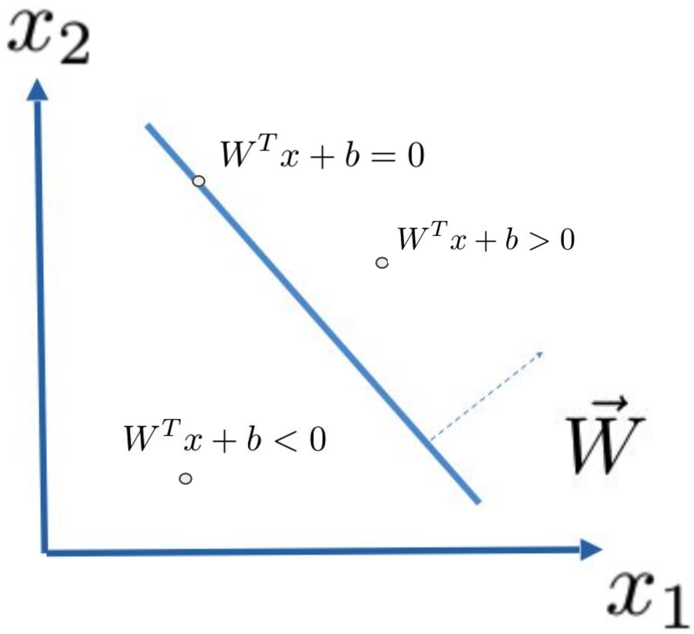


Разделяющая гиперплоскость

1) Любая точка $x \in R^n$ которая удовлетворяет уравнению $W^T x + b = 0$ лежит на этой плоскости.

2) Если $W^T x + b > 0$ то точка лежит со стороны гиперплоскости в которую смотрит вектор W

3) Если $W^T x + b < 0$ то точка лежит на противоположной стороне



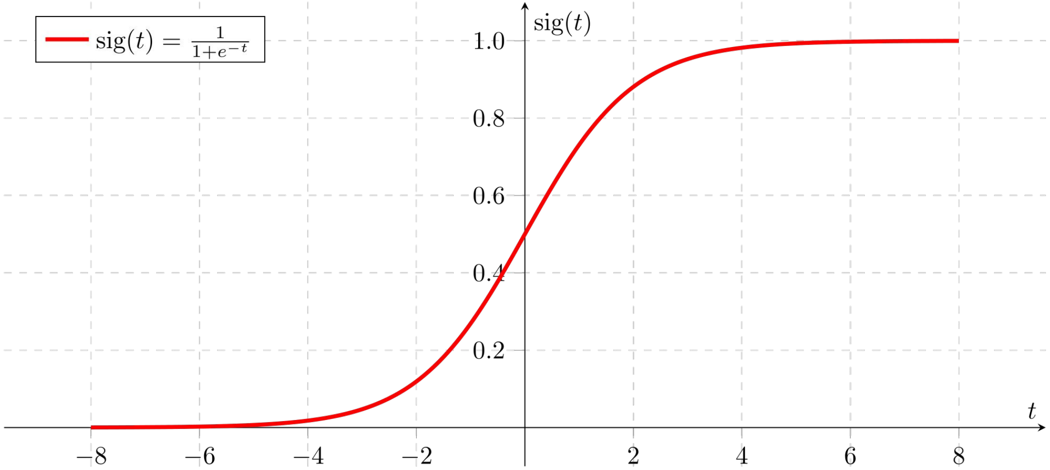
Сигмоида

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

$$\sigma(-\infty) = 0$$

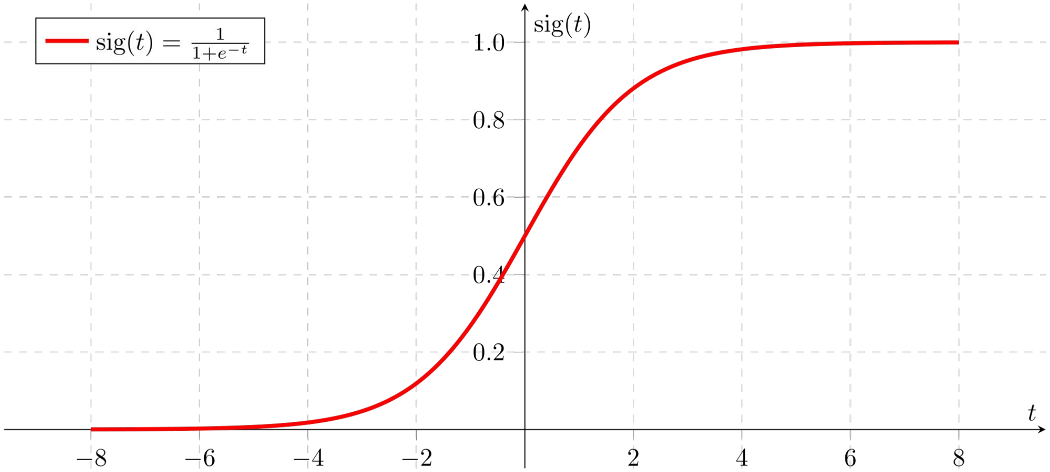
$$\sigma(0) = 0.5$$

$$\sigma(+\infty) = 1$$



Логистическая Регрессия

$$f(x) = \sigma(W^T x + b)$$



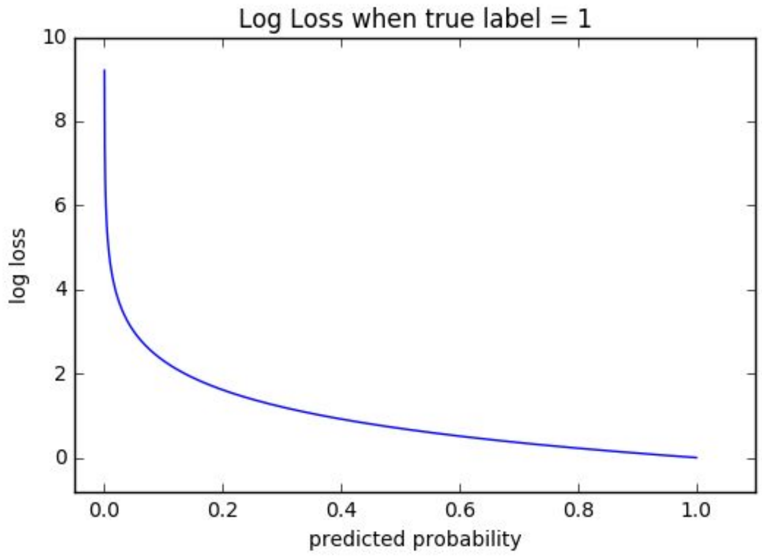
Функция Потерь (Loss)

В тренировочном датасете у нас есть метки классов y которые соответствуют своим x

Мы хотим измерять качество алгоритма по его предсказаниям y_{pred} и истинным меткам y

Log Loss

$$-L(y_{pred}, y) = \begin{cases} \log(y_{pred}) & \text{if } y = 1 \\ \log(1 - y_{pred}) & \text{if } y = 0 \end{cases}$$



$$L(y_{pred}, y) = -[y \log(y_{pred}) + (1 - y) \log(1 - y_{pred})]$$

Оптимизация

X - все сэмплы из тренировочного датасета. `X.shape == (N, n)`
Y - соответствующие метки. `Y.shape == (N, 1)`

Градиентный спуск:

```
W = random(n); b=0    # Инициализация параметров
while not_converged:
    Y_pred = f(X; W, b)
    loss = L(Y_pred, Y)
```

$$W = W - \frac{\partial loss}{\partial W}$$

$$b = b - \frac{\partial loss}{\partial b}$$

Производные

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial W} = \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial z}\frac{\partial z}{\partial W}$$

$$L(\sigma) = y\log(\sigma) + (1-y)\log(1-\sigma)$$

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

$$z = W^T x + b$$

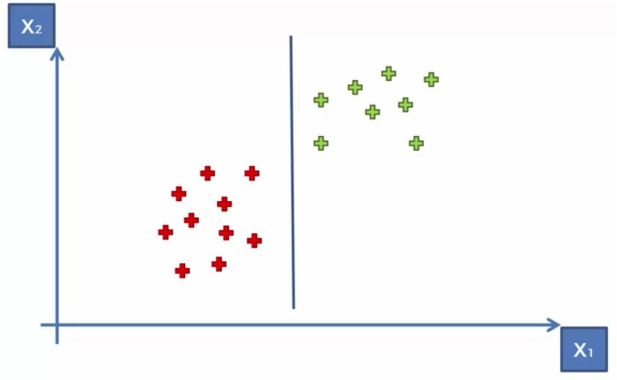
$$\frac{\partial L}{\partial \sigma} = y\frac{1}{\sigma} - \frac{1-y}{1-\sigma}$$

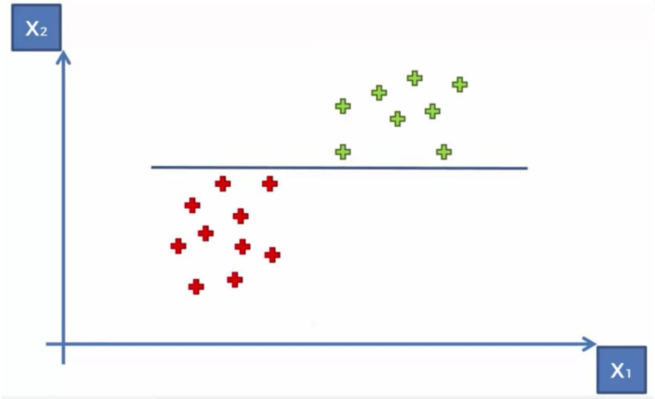
$$\frac{\partial \sigma}{\partial z} = \sigma(z)(1-\sigma(z))$$

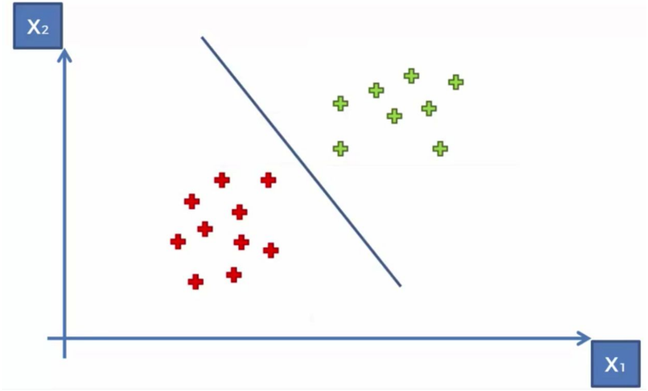
$$\frac{\partial z}{\partial W} = x$$

Support Vector Machines

Бесконечное количество решений







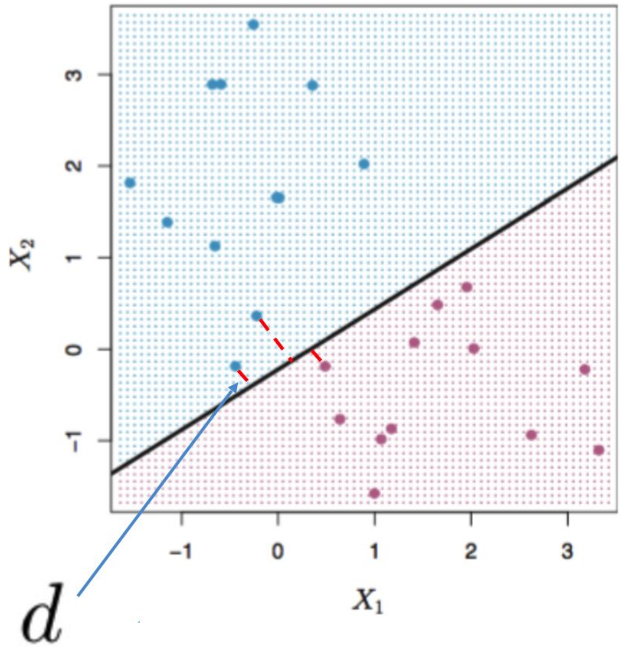
Отклонение от плоскости

$d = y(W^T x + b)$ - отклонение от разделяющей гиперплоскости.

$M_h = \min\limits_{i=1..N} d_i$ - минимальное отклонение для плоскости h

Мы хотим найти такую гиперплоскость, в которой M_h максимально:

$$h_{\max} = \operatorname*{argmax}_h M_h$$



SVM оптимизация

Есть 2 основных подхода для нахождения разделяющей плоскости.

Hinge Loss:

- Аппроксимация сложной математики
- Функция потерь для градиентного спуска
- Просто написать и использовать

Метод множителей Лагранжа:

- Выпуклая оптимизация со множеством ограничений
- Больно
- Интересно

SVM оптимизация

Есть 2 основных подхода для нахождения разделяющей плоскости.

Hinge Loss: ✅

- Аппроксимация сложной математики
- Функция потерь для градиентного спуска
- Просто написать и использовать

Метод множителей Лагранжа:

- Выпуклая оптимизация со множеством ограничений
- Больно
- Интересно

Постановка задачи

Мультикласс классификация с m количеством классов.

$$f(x; W) = Wx \qquad W \in R^{m \times n}, x \in R^n$$

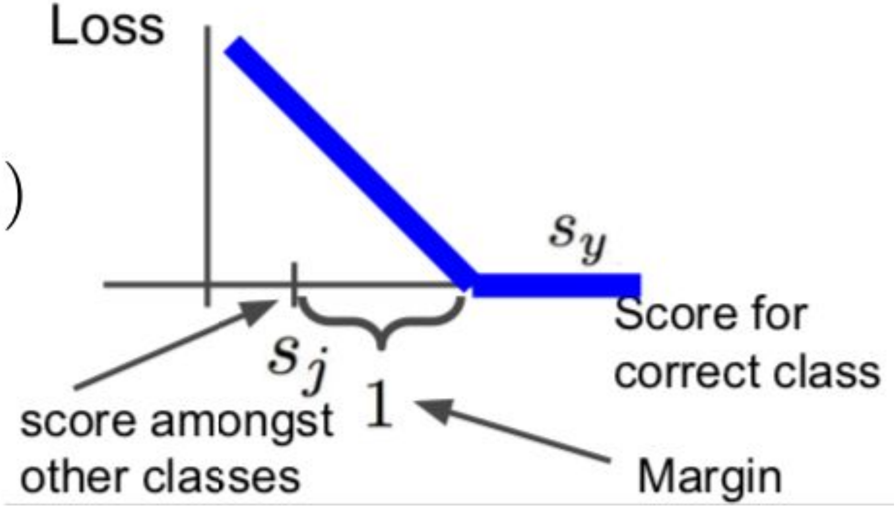
Выход модели это вектор s размера m

s_i это score класса i

Hinge Loss

$$L = \sum_{j!=y} \begin{cases} 0 & \text{if } s_y >= s_j + 1 \\ s_j - s_y + 1 & \text{otherwise} \end{cases}$$

$$L = \sum_{j!=y} \max(0, s_j - s_y + 1)$$



Reference: http://cs231n.stanford.edu/syllabus.html

Example

$$L = \sum_{j!=y} \max(0, s_j - s_y + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

Reference: http://cs231n.stanford.edu/syllabus.html

Example

$$L = \sum_{j!=y} \max(0, s_j - s_y + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9		

Reference: http://cs231n.stanford.edu/syllabus.html

Example

$$L = \sum_{j!=y} \max(0, s_j - s_y + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	

Reference: http://cs231n.stanford.edu/syllabus.html

Example

$$L = \sum_{j != y} \max(0, s_j - s_y + 1)$$

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

Reference: http://cs231n.stanford.edu/syllabus.html

Регуляризация

Представьте, что мы нашли W такое, что L = 0
Это решение не уникально!

Мы можем подставить 2*W и получить ту же самую гиперплоскость.
Но большие веса W делают решение численно нестабильным.

Для того чтобы этого не допустить, мы можем штрафовать модель за большие веса.

Регуляризация

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data Loss}} + \underbrace{\lambda R(W)}_{\text{Регуляризация}}$$

L2: $R(W) = \sum_{i,j} W_{ij}^2$

L1: $R(W) = \sum_{i,j} |W_{ij}|$

Softmax

Мотивация

Мы хотим каждую компоненту вектора $s = f(x; W)$ интерпретировать, как вероятность i-го класса.

Должны соблюдаться следующие свойства:

$$0 \leq s_i \leq 1$$

$$\sum_i s_i = 1$$

Softmax

Вероятность k-го класса находится по формуле:

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

cat	**3.2**
car	5.1
frog	-1.7

Reference: http://cs231n.stanford.edu/syllabus.html

Softmax

Вероятность k-го класса находится по формуле:

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Probabilities must be >= 0

cat	**3.2**		**24.5**
car	5.1	$\xrightarrow{\text{exp}}$	164.0
frog	-1.7		0.18

unnormalized probabilities

Reference: http://cs231n.stanford.edu/syllabus.html

Softmax

Вероятность k-го класса находится по формуле:

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

			Probabilities must be >= 0		Probabilities must sum to 1
cat	**3.2**		**24.5**		**0.13**
car	5.1	exp →	164.0	normalize →	0.87
frog	-1.7		0.18		0.00
			unnormalized probabilities		probabilities

Reference: http://cs231n.stanford.edu/syllabus.html

Log Loss

Мы хотим иметь функцию потерь с такими же свойствами, как и бинарный Log Loss.

Нужно максимизировать вероятность правильного класса p_i

Это эквивалентно минимизации отрицательного логарифма от вероятности:

$$L = -\log P(Y = y|X = x) = -\sum_{j}^{m} y_j \log P(Y = y|X = x)$$