Лекция 3: метрический подход к прогнозированию, проклятие размерности, переобучение

Что прогнозирует модель?

- Если есть несколько наблюдений с одинаковым вектором признаков x_*, но разными откликами $y_{*1}, y_{*2}, \dots y_{*k}$, то какой прогноз минимизирует эмпирический риск (ошибку)? Ответ:

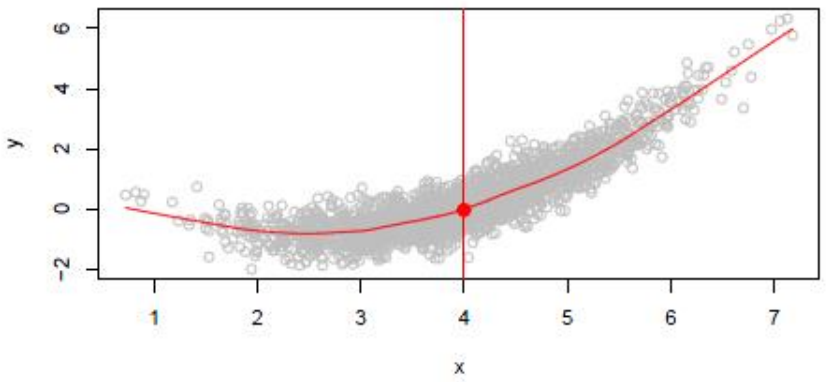
$$y_* = \arg\min_y Q(y, \{y_{*1}, y_{*2}, \dots y_{*k}\}) = \arg\min_y \frac{1}{k} \sum_{*i} L(y_{*i}, y),$$

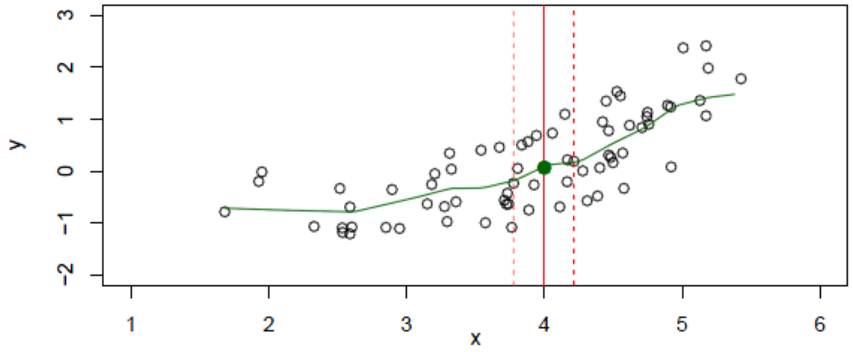
например, для квадратичной функции потерь: $y_* = E(y|x = x_*)$

- Если повторяющихся наблюдений нет, то можно «приблизить»:

$y_* = E(y|x = x_*) \approx mean[y|x \in N(x_*)]$, где $N(x_*)$ - «окрестность» x_*

- Но должна выполняться гипотеза о компактности (или о непрерывности) и может быть проблема проклятия размерности





Обобщенный метрический классификатор

- Для выбранного x_*, ранжируем выборку $\{x^{(1)}, x^{(2)}, \ldots, x^{(l-1)}\}$, так чтобы:

$$d(x_*, x^{(1)}) \leq d(x_*, x^{(2)}) \leq \ \ldots \leq d(x_*, x^{(l-1)})$$

$x^{(i)}$ - i-й сосед x_*, а $y^{(i)}$ - отклик i-го соседа

- Метрический алгоритм классификации:

$$a(x, Z) = \arg\max_{y \in Y} \Gamma_y(x)$$

$\Gamma_y(x) = \sum_{i=1}^{l} [y = y^{(i)}] w(i, x)$ - оценка близости объекта x к классу y,

$w(i, x)$ – важность i-го соседа, не возрастает по i, неотрицательна

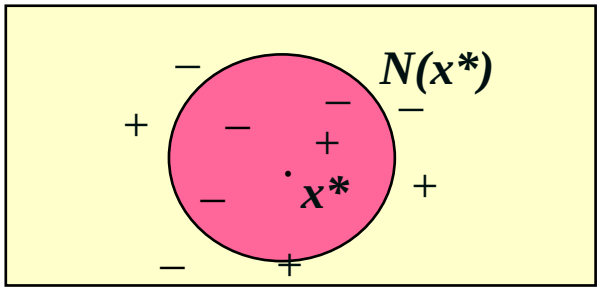
- Методы:
  - «ближайшего соседа» $w(i, x) = [i \leq 1]$
  - «k ближайших соседей» $w(i, x) = [i \leq k]$

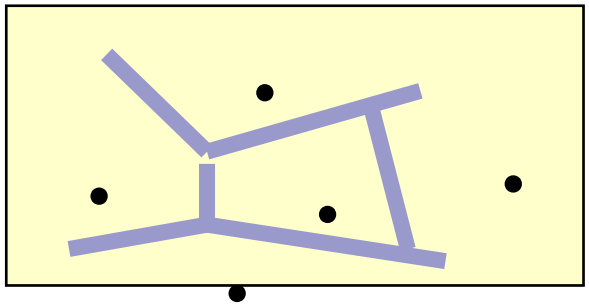
Простые методы K ближайших соседей

- Общая схема работы:
  - Каждый пример – точка в пространстве, все примеры хранятся
  - Вводится метрика расстояния с учетом нормирования координат
  - Ищется K (от 1 до …) ближайших соседей
  - Прогноз вычисляется как агрегирующая функция от откликов найденных соседей
- Метод KNN:
  - Для задачи регрессии отклик считается как среднее по откликам всех соседей:
  - Для классификации выбирается самый частый класс:

$$y^* = \frac{1}{K} \sum_{x_i \in N(x^*)} y_i$$

$$y^* = \arg\max_{c \in C, x_i \in N(x^*)} \left[|y_i = c| \right]$$





Метод «взвешенных» K ближайших соседей

- На базе KNN, но:
  - помимо распределения «отклика» учитываются и расстояния до соседей $w(i,x) = [i \le k]w_i$, где w_i зависит от близости до x
- Примеры весов:
  - линейный вес $w_i = \frac{k+1-i}{k}$, экспоненциальный $w_i = q^i, 0 < q < 1$
  - ядра, например, $w_i = \exp\left(-d_i^2/h\right)$
- Усреднение отклика за счет:
  - «взвешенного» голосования для классификации и «взвешенного» среднего для регрессии (далее покажем откуда это следует)

$$y^* = \arg\max_{c \in C, x_i \in N(x^*)} \left[\frac{w_i \left| y_i = c \right|}{\sum_{x_j \in N(x^*)} w_i} \right]$$

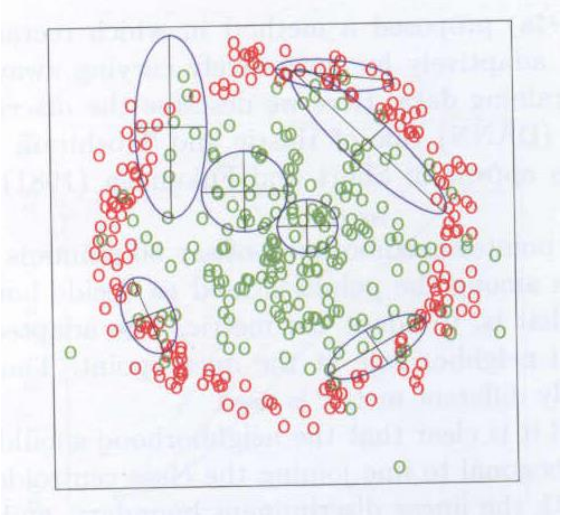
$$y^* = \frac{\sum_{x_i \in N(x^*)} w_i y_i}{\sum_{x_i \in N(x^*)} w_i}$$

Метод K ближайших соседей с дискриминантным адаптивным расстоянием

- Метод DANN:
  - На базе KNN, но используется итеративный (m-номер итерации) расчет локального расстояния Махаланобиса с учетом структуры распределения соседей в окрестности:

$$d^{(m)}(x^*, x_i) = (x^* - x_i)^T \sum_{(m)}^{-1} (x^* - x_i)$$

- Параметры алгоритма:
  - K_M – число соседей для оценки метрики (нужно побольше)
  - K – число соседей для прогноза (лучше поменьше)



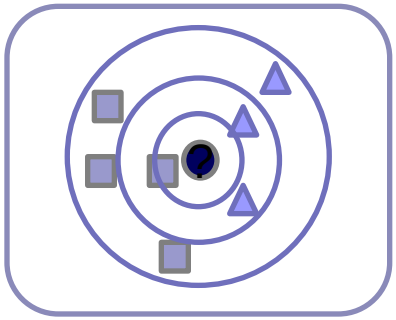
Выбор параметра K

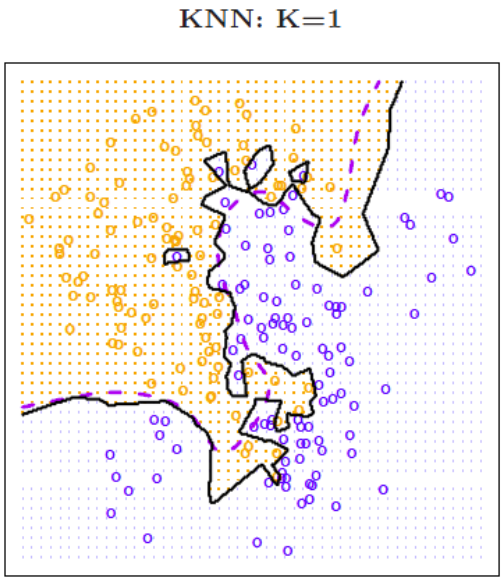
Важность K:

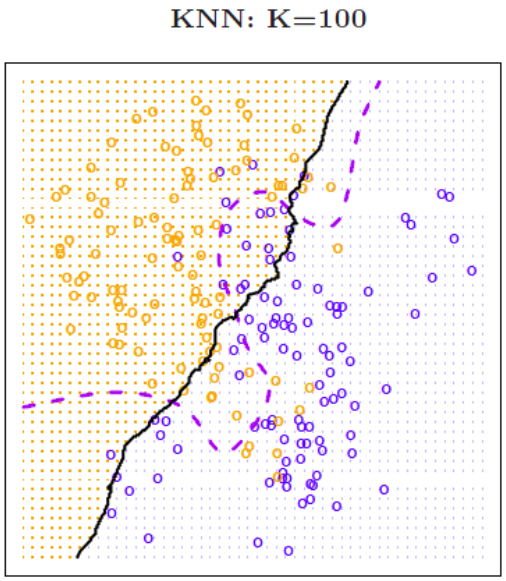
k = 1: Результат = квадрат

k = 5: Результат = треугольник

k = 7: Снова квадрат







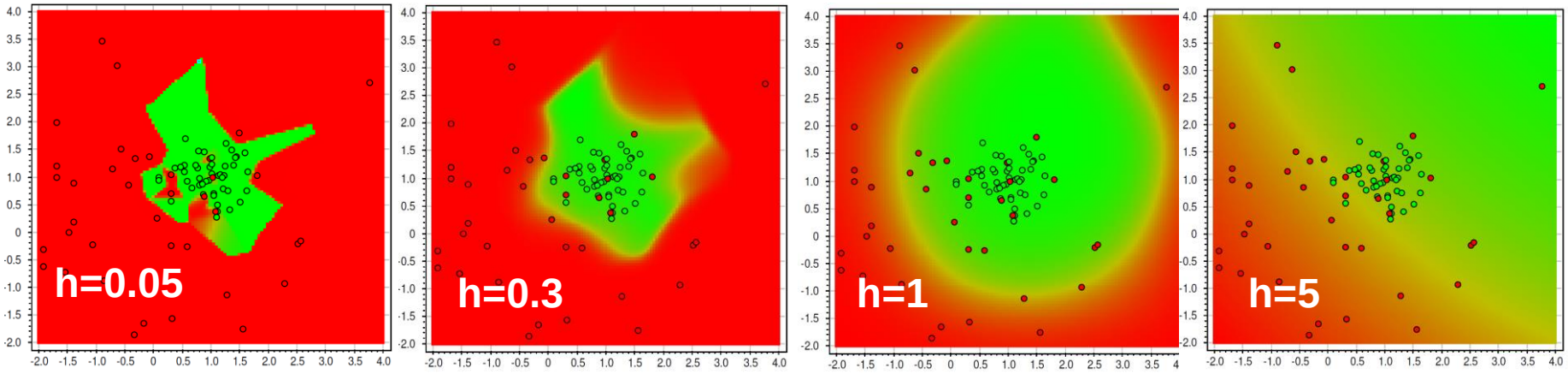
- Выбор *k*:
  - Если *k* мал, то чувствительность к шуму, и негладкие границы классов (или линии уровня для регрессий)
  - Если *k* велико, то окрестность может сильно «задеть» соседний класс, зато гладкие границы
  - При классификации следует нечетный *k*, чтобы не было «ничьей»
  - Выбирается кросс-валиадцией или на валидационном наборе (далее)
  - Стандартная эвристика k=sqrt(n)

Метод окна Парзена (для классификации)

- Вес задается радиально-базисным ядром (h –«ширина ядра»):

$$\mathrm{w(i},x) = K\left(\frac{d\left(x,x^{(i)}\right)}{h}\right)$$

- h – фиксировано $a(x,Z,h) = \arg\max_{y\in Y} \sum_{i=1}^{l} \left[y = y^{(i)}\right] K\left(\frac{d\left(x,x^{(i)}\right)}{h}\right)$
- k – фиксировано $a(x,Z,k) = \arg\max_{y\in Y} \sum_{i=1}^{l} \left[y = y^{(i)}\right] K\left(\frac{d\left(x,x^{(i)}\right)}{d\left(x,x^{(k)}\right)}\right)$



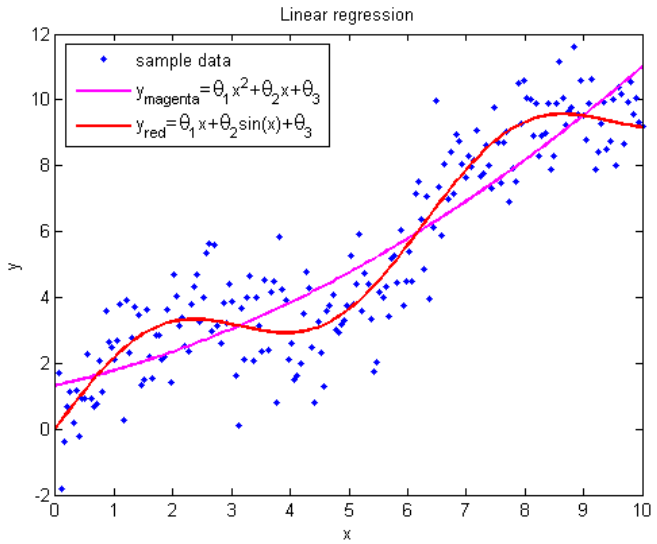
Параметрические модели

- Параметрическое семейство функций

$$A = \{g(x,\theta) \mid \theta \in \Theta\}, g: X \times \Theta \to Y$$

Θ – множество допустимых параметров

- Линейная модель:
  - Классификация $g(x,\theta) = sign\left[\sum_{i=1}^{q} \theta_i f_i(x)\right], Y = \{-1,1\}$
  - Регрессия $g(x,\theta) = \sum_{i=1}^{q} \theta_i f_i(x), Y = \mathbb{R}$
    - θ-вектор параметров
    - f_i-не обязательно линейная
    - если f_i-зависит от одного признака,
    - то модель аддитивная
- Пример:
  - Признаки $\{1, x, x^2\}$ vs $\{1, x, \sin(x)\}$



Метод наименьших квадратов для линейной регрессии

- Оценка ошибки - квадратичная функция потерь:

$$Q(\theta, Z) = \frac{1}{l}\sum_{i=1}^{l}(y_i - a(\bar{x}_i, \theta))^2 = \frac{1}{l}\sum_{i=1}^{l}\left(y_i - \theta_0 - \sum_{j=1}^{p} x_{ij}\theta_j\right)^2$$

- В матричной форме:

$$Q(\theta, Z) = (\bar{y} - X\theta)^T(\bar{y} - X\theta)$$

- Единственное оптимальное решение (если матрица данных не сингулярная)

$$\theta = (X^T X)^{-1} X^T \bar{y}$$

- Но не все так просто:
  - Если сингулярная матрица данных из-за коррелированных факторов
  - Большое число регрессоров – плохая точность и интерпретируемость

Непараметрическая (ядерная) регрессия Надарая-Ватсона

- Суть метода:
  - МНК (квадратичная функция потерь)
  - Локальный прогноз константой в точке
  - Ядерные веса для определения локальной окрестности
- Постановка задачи:

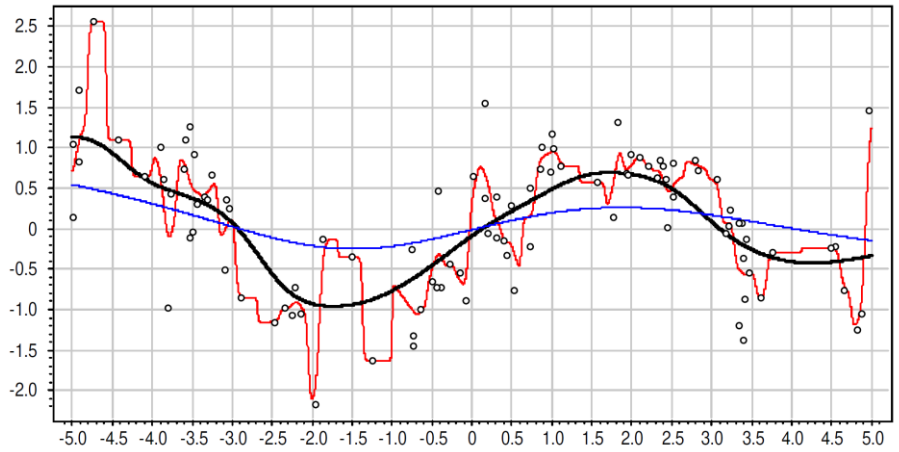
$$Q(Z,\theta) = \frac{1}{l}\sum_{i=1}^{l} K\left(\frac{d\left(x,x^{(i)}\right)}{h}\right)(y_i-\theta)^2 \to \begin{matrix} min \\ \theta \in \mathbb{R} \end{matrix}$$

- Решающая функция:

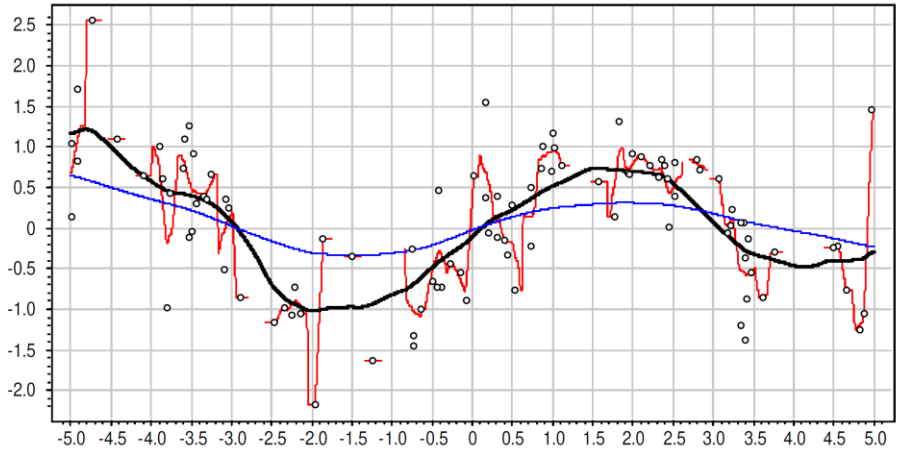
$$a(x,Z,h) = \theta = \frac{\sum_{i=1}^{l} y_i K\left(\frac{d\left(x,x^{(i)}\right)}{h}\right)}{\sum_{i=1}^{l} K\left(\frac{d\left(x,x^{(i)}\right)}{h}\right)}$$

Непараметрическая (ядерная) регрессия Надарая-Ватсона

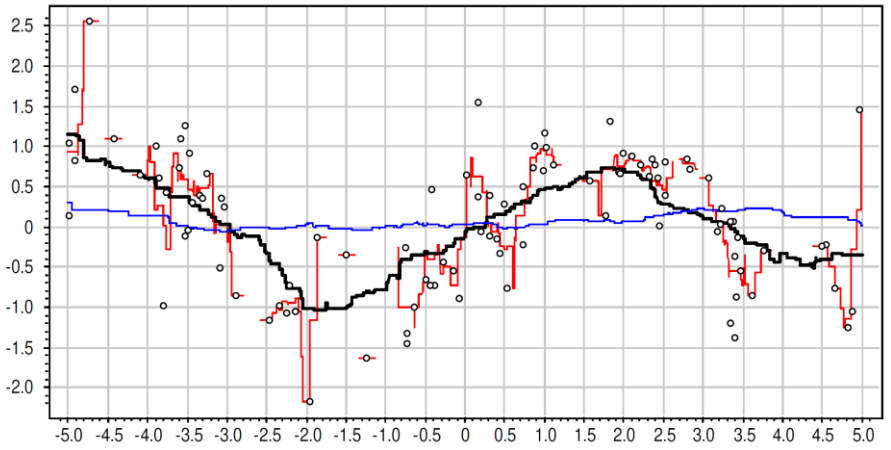
$h \in \{0.1, 1.0, 3.0\}$, гауссовское ядро $K(r) = \exp(-2r^2)$



$h \in \{0.1, 1.0, 3.0\}$, треугольное ядро $K(r) = (1 - |r|)[|r| \leqslant 1]$



$h \in \{0.1, 1.0, 3.0\}$, прямоугольное ядро $K(r) = [|r| \leqslant 1]$

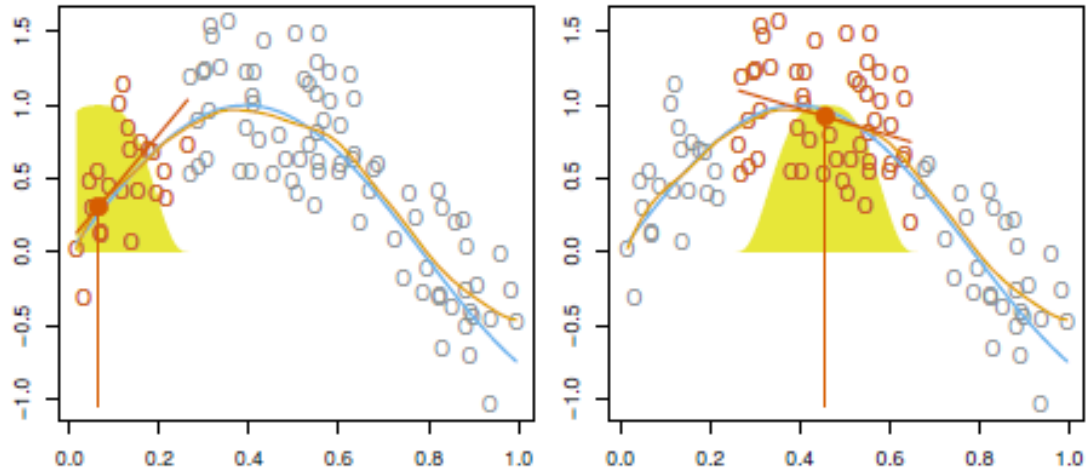


- Чем больше h тем «проще» функция
- Гладкость определяется ядром (непрерывная/кусочно линейная, кусочно постоянная аппроксимация)
- Разрыв, если нет точек в окрестности

Локальная взвешенная регрессия

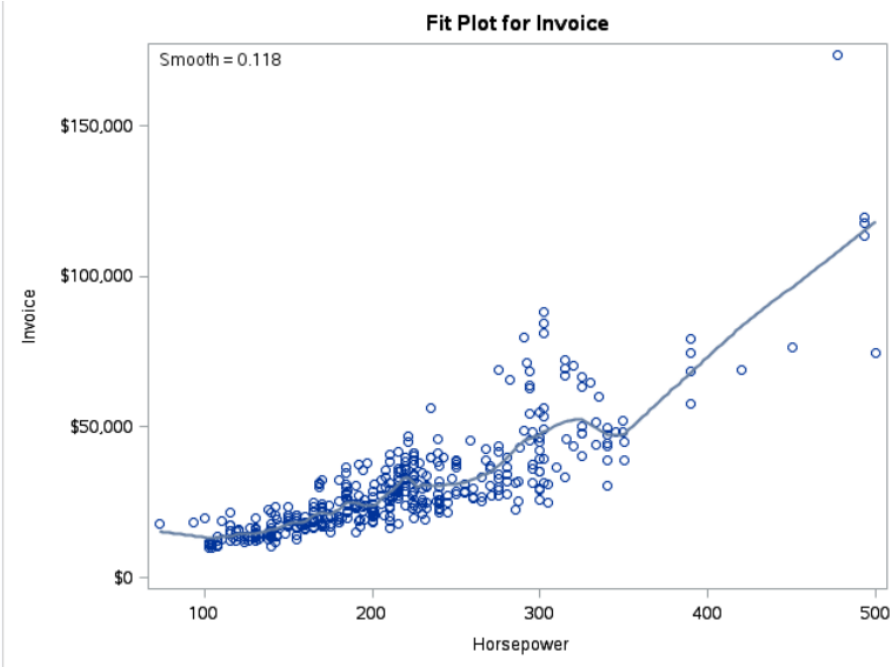
- Вместо константы (как в kernel regression) простая локальная параметрическая модель, например, линейная:

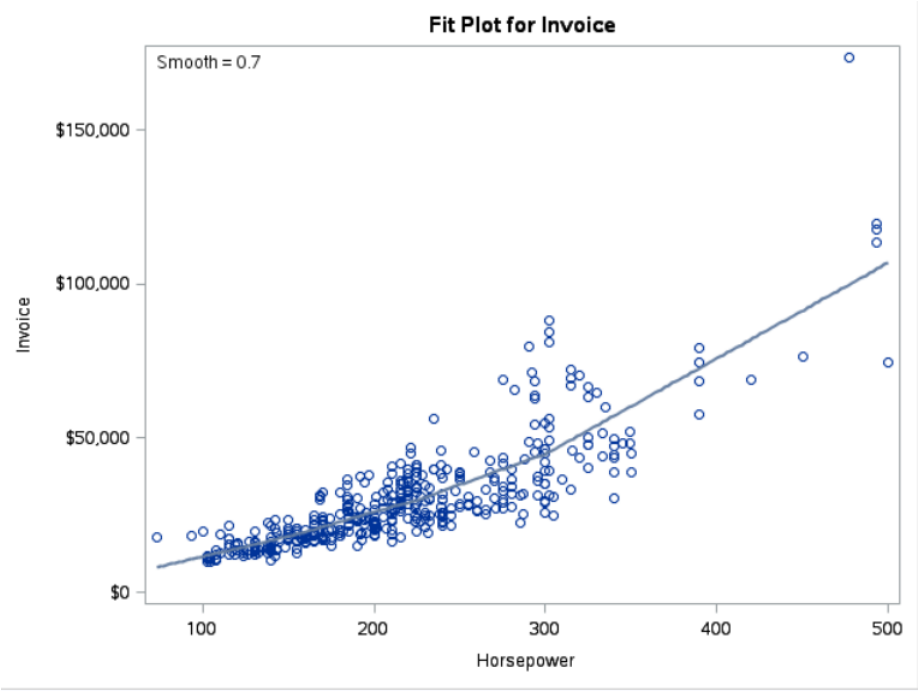
$$Q(Z,\theta) = \frac{1}{l}\sum_{i=1}^{l} K\left(\frac{d\left(x, x^{(i)}\right)}{h}\right)(y_i - \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{\theta})^2 \rightarrow \min_{\theta \in \mathbb{R}^p}$$



Локальная взвешенная регрессия

- Нужно задавать параметр сглаживания (фактически – штраф за сложность), который определяет число точек окрестности, чтобы не усложнять модель:





Пример (Python)

Data Set Characteristics:

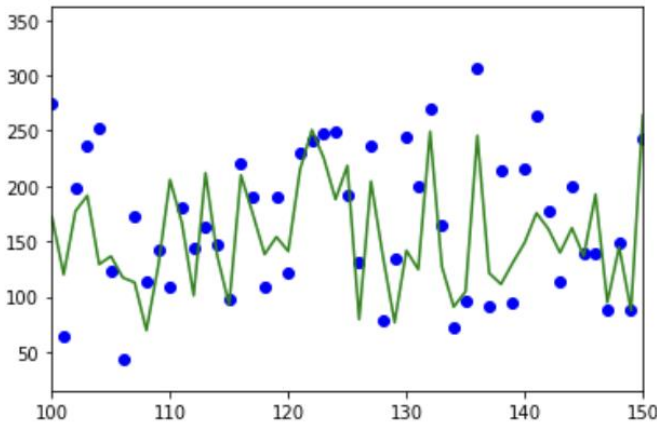
Number of Instances:	442
Number of Attributes:	First 10 columns are numeric predictive values
Target:	Column 11 is a quantitative measure of disease progression one year after baseline
Attribute Information:	• age age in years • sex • bmi body mass index • bp average blood pressure • s1 tc, total serum cholesterol • s2 ldl, low-density lipoproteins • s3 hdl, high-density lipoproteins • s4 tch, total cholesterol / HDL • s5 ltg, possibly log of serum triglycerides level • s6 glu, blood sugar level

```
plt.scatter(range(len(y_test)), y_test, color="blue")
plt.plot(KNN.predict(X_test), color="green")
plt.xlim([100, 150])
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.datasets import load_diabetes
```

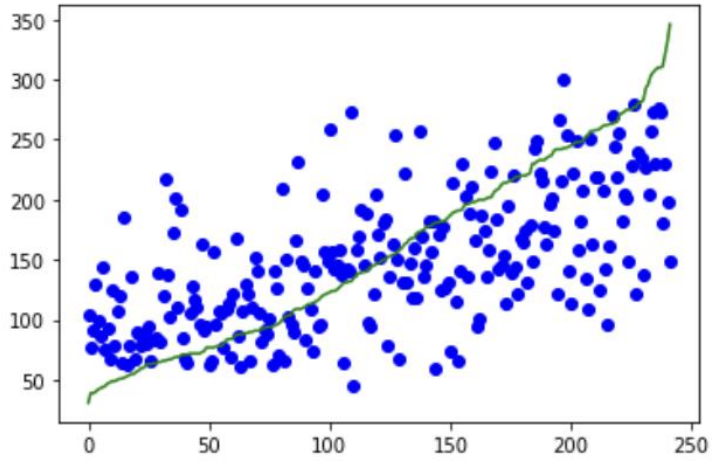
```
N = 200
data = load_diabetes()
X, X_test = data.data[:N], data.data[N:]
y, y_test = data.target[:N], data.target[N:]
```

```
# weights="uniform" is default
# weights="distance" is for KWNN
# weights as user function: distances -> weight (implement DAN
KNN = KNeighborsRegressor(n_neighbors=5, weights="distance")
KNN.fit(X, y)
pass
```

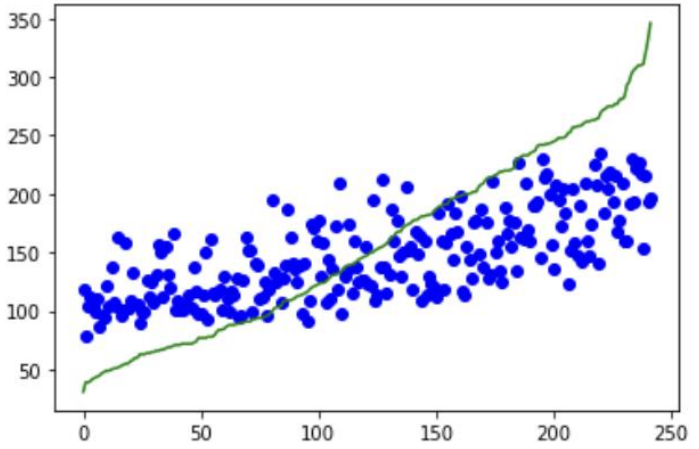


Пример (Python)

```
KNN = KNeighborsRegressor(n_neighbors=3, weights="distance")
KNN.fit(X, y)
y_pred=KNN.predict(X_test)
rs=pd.DataFrame([y_pred, y_test]).T
rs.sort_values(1,inplace=True)
plt.scatter(range(len(rs[0])), [rs[0]], color="blue")
plt.plot(range(len(rs[1])),rs[1], color="green")
```




```
KNN = KNeighborsRegressor(n_neighbors=30, weights="distance")
KNN.fit(X, y)
y_pred=KNN.predict(X_test)
rs=pd.DataFrame([y_pred, y_test]).T
rs.sort_values(1,inplace=True)
plt.scatter(range(len(rs[0])), [rs[0]], color="blue")
plt.plot(range(len(rs[1])),rs[1], color="green")
```



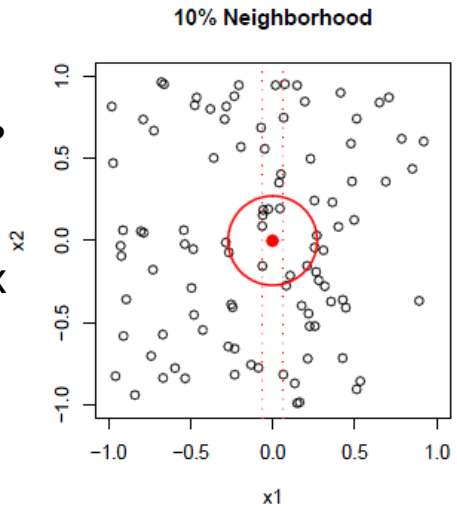
Свойства метрических методов

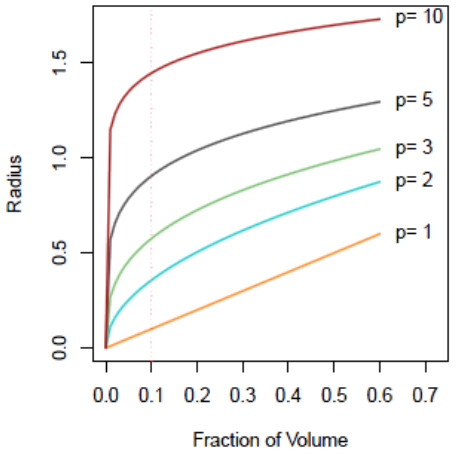
- Основные свойства:
  - «Ленивая модель» - не надо ничего обучать
  - Обязательно нужна хорошая метрика и значимые признаки
  - Есть критические **метапараметры**, определяющие сложность модели (гладкость границы или изолиний)
- Достоинства:
  - Простота реализации
  - Один из самых точных методов (при корректной настройке)
  - Легко адаптируется под сложные типы «откликов», включая ранжирование, многотемность и т.д.
  - Можно интегрировать экспертные знания, задавая веса у примеров, или параметры у метрики
- Недостатки:
  - «черный ящик» - результат не интерпретируемый совсем
  - Достаточно вычислительно трудоемкие
  - **«Проклятие размерности»**

Проклятие размерности

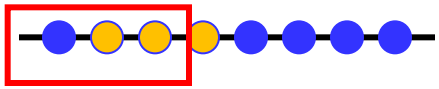
- Суть проблемы: экспоненциальный рост числа необходимых наблюдений при линейном росте размерности пространства
- Пример: ближайшие соседи как правило расположены далеко при больших размерностях пространства признаков.

Например, нам нужно получить значительную часть выборки, чтобы сгладить границу и снизить случайность в усредненном прогнозе, пусть 10%.
10% соседей для случая больших размерностей не может быть локализована, так что мы уже не можем оценить отклик на основе локального усреднения.

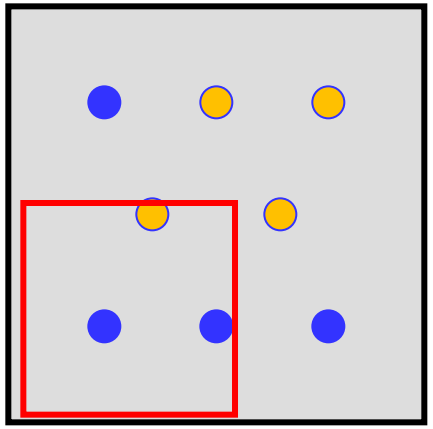




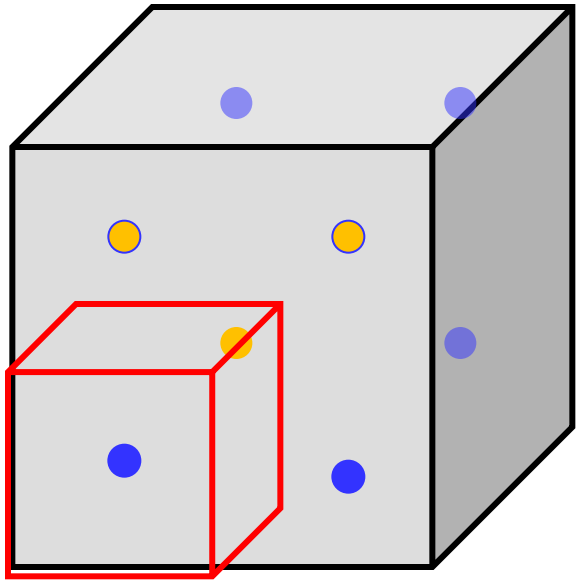
Модельный пример, демонстрирующий проклятие размерности



1–D



2–D



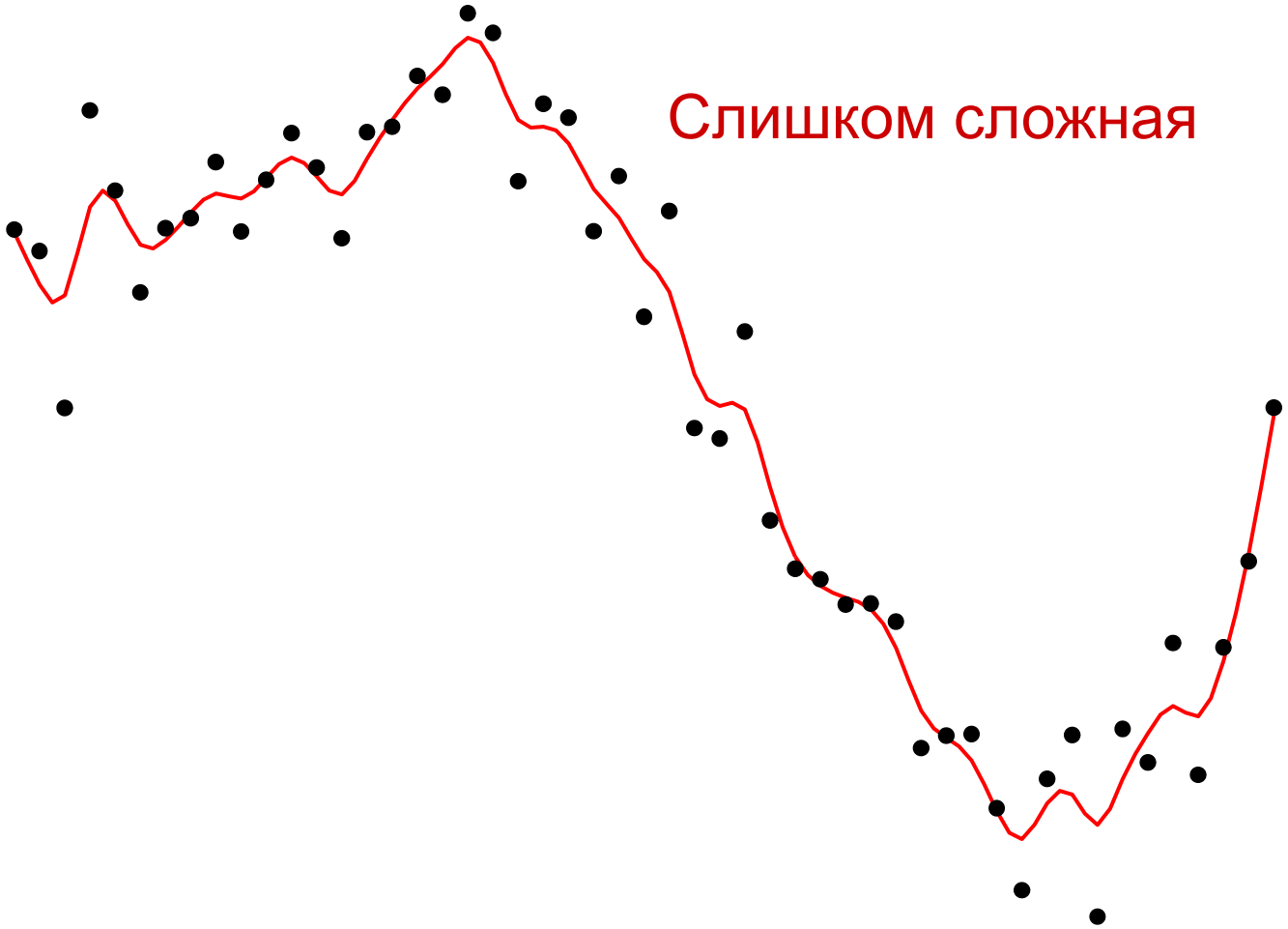
3–D

- $r=K/N$
- $E_p(r)=r^{1/p}$
- $E_{10}(0.01)=0.63$
- $E_{10}(0.1)=0.8$

Сложность модели

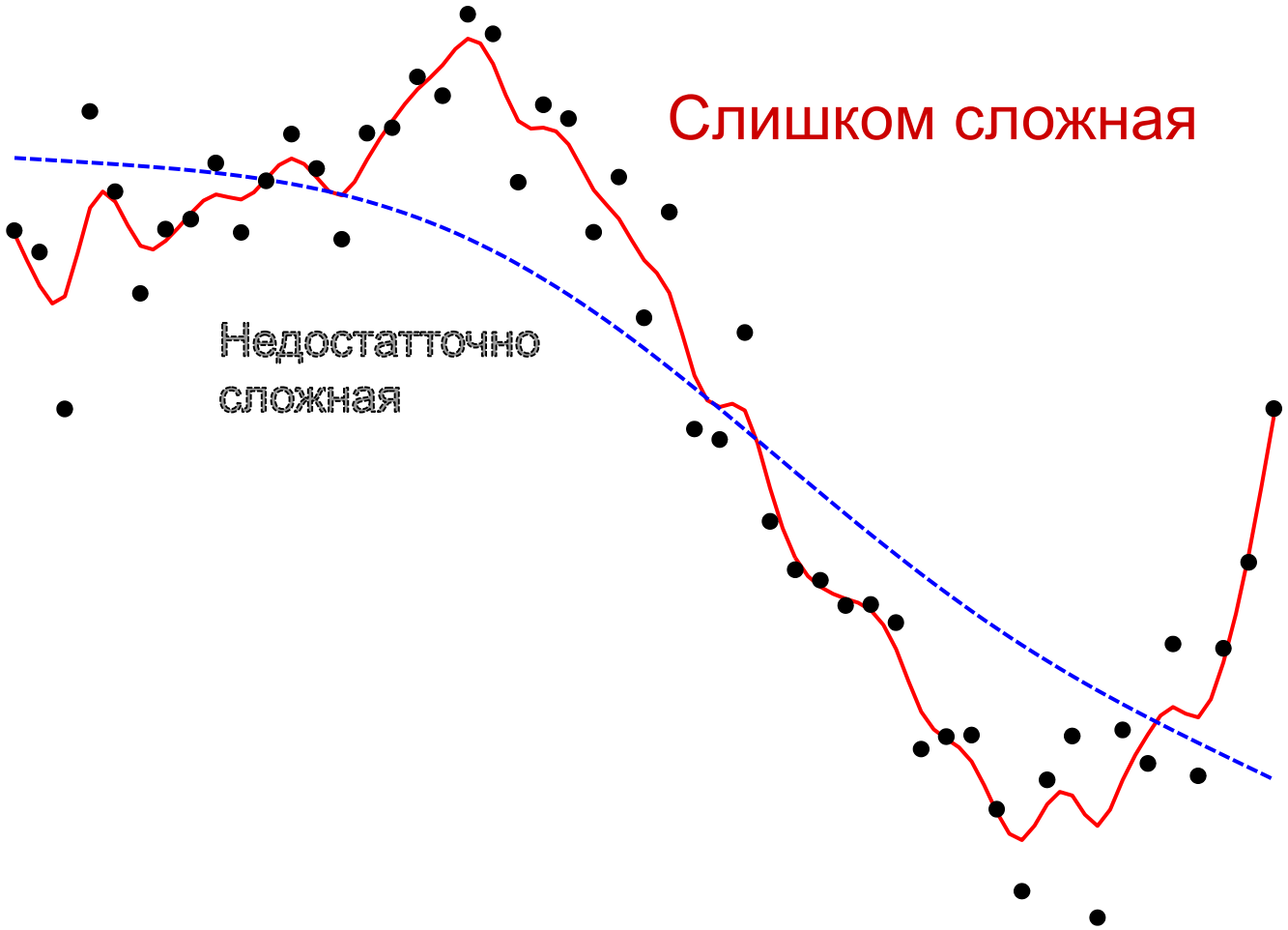
...

Сложность модели



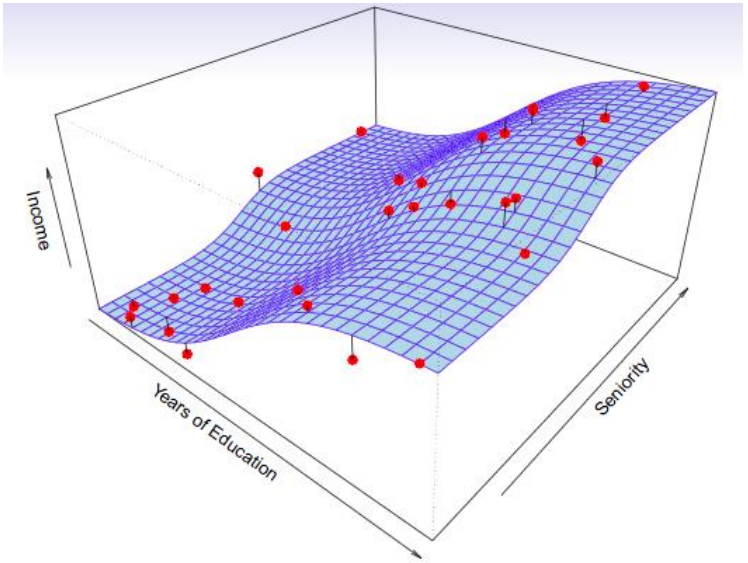
...

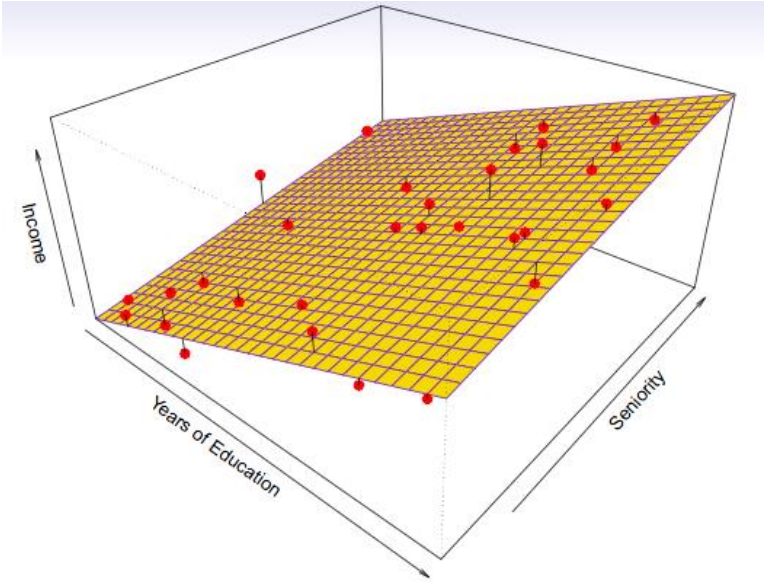
Сложность модели



...

Проблема недообучения и переобучения





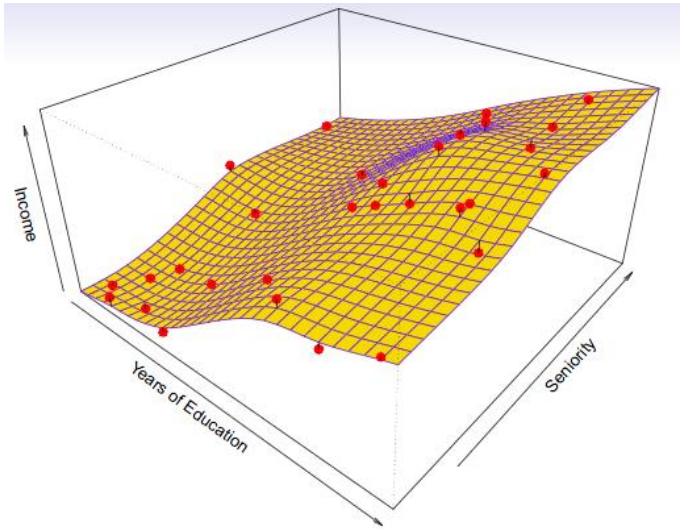
Модельный пример.

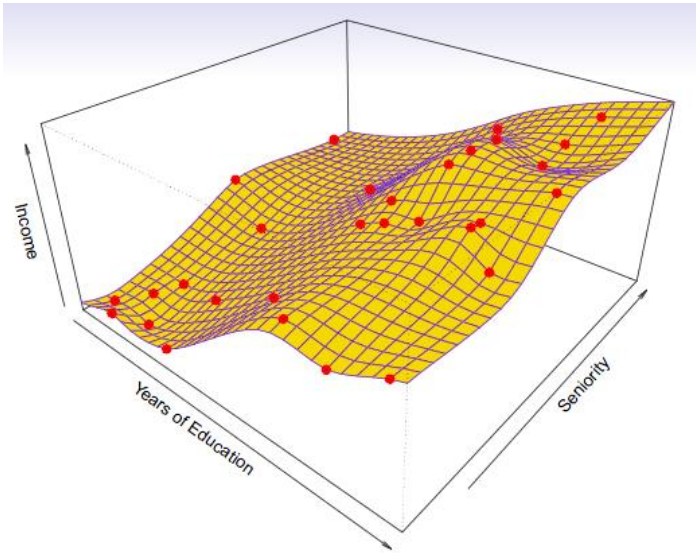
- Красные точки - наблюдения, синяя поверхность – истинная зависимость $\texttt{income} = f(\texttt{education}, \texttt{seniority}) + \epsilon$
- Желтая поверхность линейная модель

$$\hat{f}_L(\texttt{education}, \texttt{seniority}) = \hat{\beta}_0 + \hat{\beta}_1 \times \texttt{education} + \hat{\beta}_2 \times \texttt{seniority}$$

- Плохая точность приближения

Проблема недообучения и переобучения





Модельный пример.

- Более сложные модели (сплайны или полиномиальные регрессии или нейронные сети или еще что-то)
- Справа модель не допускает ошибок на обучающем наборе.
- Это хорошо? Нет!

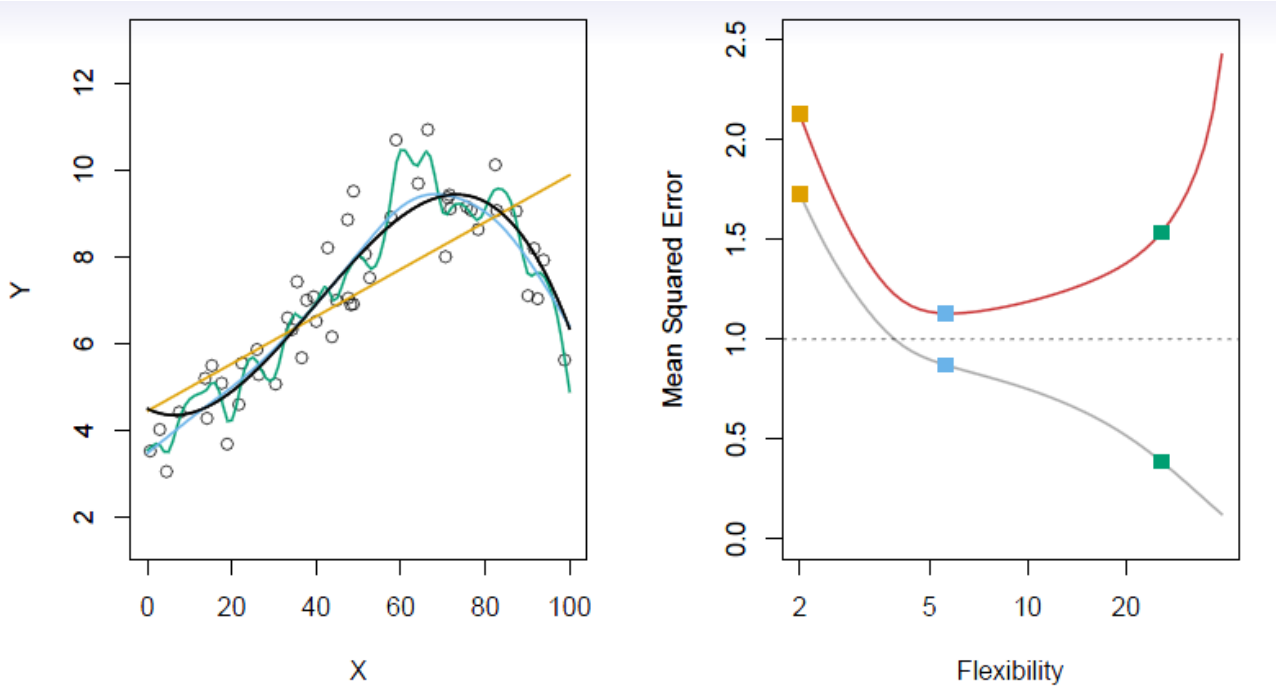
Недообучение vs переобучение

- Основная проблема машинного обучения!!!
  - Недообучение - низкое качество (большой эмпирический риск) на тренировочном наборе и на этапе скоринга
  - Переобучение - высокое качество (маленький эмпирический риск) на тренировочном наборе и плохое качество на этапе скоринга (большой эмпирический риск)
- Причины:
  - Сложность модели: например, для параметрических моделей много степеней свободы (параметров модели) или слишком сложное уравнение или большая норма вектора параметров
  - Плохое качество данных: шум и выбросы, малый объем или несбалансированность тренировочной выборки
  - Зависимости в пространстве признаков
- Обобщающая способность:
  - способность алгоритма «качественно» прогнозировать отклик для объектов одной природы (из одной генеральной совокупности), которых не было в тренировочном наборе
  - Как оценить?

Борьба с переобучением

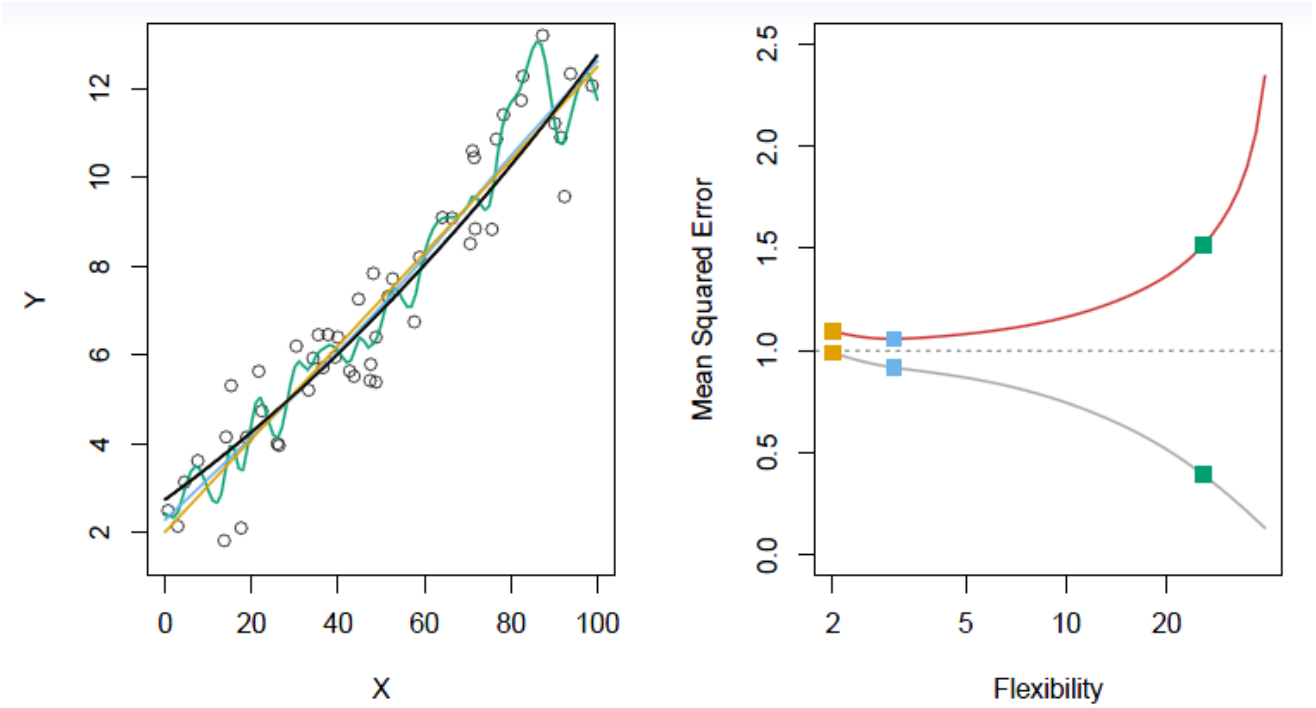
- Ограничить сложность модели (например, регуляризация)
- Преобразовать данные (удалять шум, уменьшать размерность и тд.)
- Использовать теоретические оценки обобщающей способности для некоторых методов обучения (обычно бесполезно, т.к. это оценки сверху)
- Эмпирически оценивать обобщающую способность с помощью тестовой выборки (или процедуры, имитируя проверку на тестовой выборке):
  - Строим модель на обучающем наборе данных и хотим, чтобы она была наилучшей.
  - Можем взять, например, квадратичную функцию потерь и оценить ее через среднеквадратичную ошибку MSE_{Tr}
  - Оценка может быть смещена в сторону более сложных моделей.
  - Поэтому мы вычисляем оценку MSE_{Te}, используя тестовый набор данных, который не участвовал в обучении модели

Оценка качества модели (сложная зависимость, много шума)



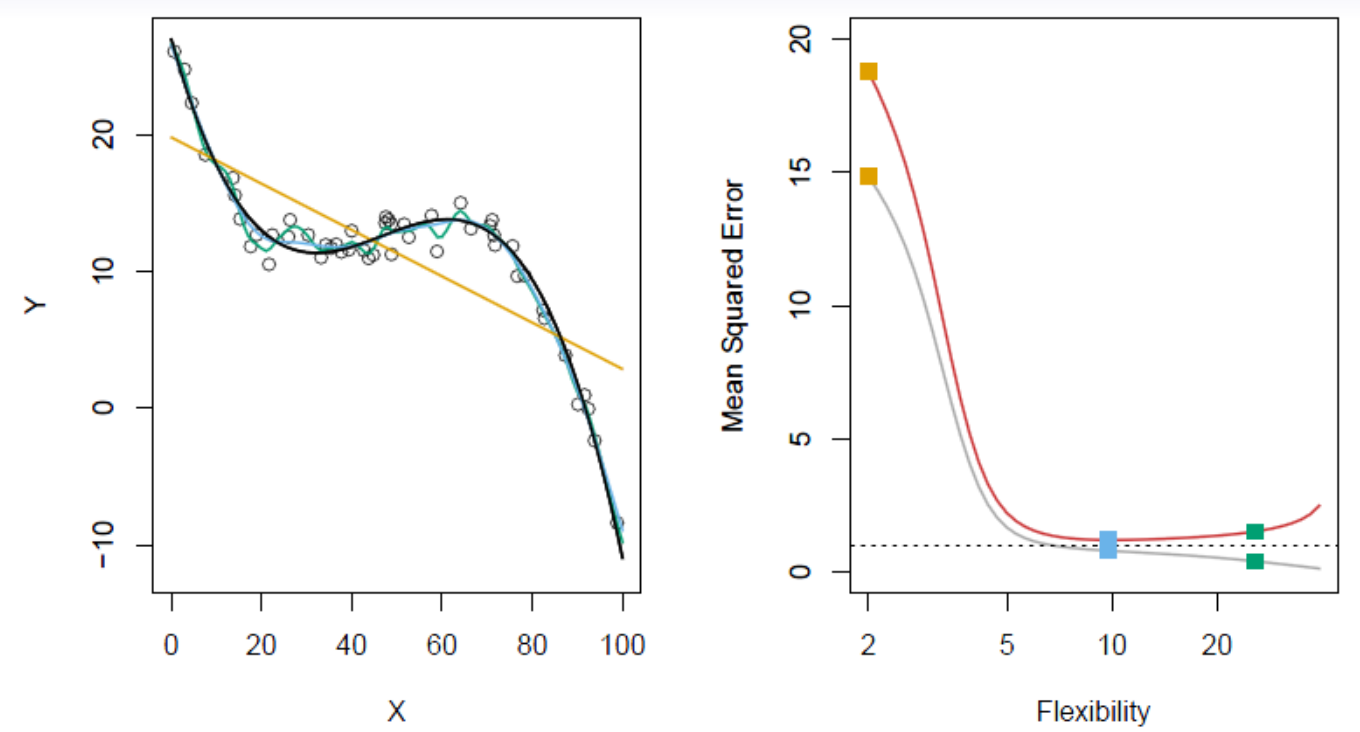
- Кривая, обозначенная черным цветом, - истинные значения.
- Красная кривая на правом рисунке – MSE_{Te} , серая кривая – MSE_{Tr}.
- Оранжевая, голубая и зеленая кривые соответствуют подгонке моделей различной сложности.
- Простые модели недообучены, сложные модели переобучены

Оценка качества модели (простая зависимость, много шума)



- Простые модели дают высокую обощающую способность
- Сложные модели переобучены

Оценка качества модели (сложная зависимость, мало шума)



- Простые модели недообучены
- Сложные обладают хорошей обобщающей способностью

MSE декомпозиция

$$MSE = E[(a(x) - y(x)]^2 = E[a(x)^2] + E[y(x)^2] - E[2a(x)y(x)] =$$
$$= Var(a(x)) + Var(y(x)) + (E[a(x)] - E[y(x)])^2$$

Дисперсия прогноза

Дисперсия шума

(не зависит от модели)

Квадрат смещения

Компромисс: Дисперсией vs Смещение!!!!

Сложнее модель => точнее приближение => меньше смещение +++

Сложнее модель => больше параметров => больше дисперсия ---

… и наоборот …

Поиск баланса между точностью и сложностью = поиск компромисса между смещением и дисперсией

MSE декомпозиция (примеры)

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

y – наблюдения отклика, *y(.)* – истинная зависимость, ε – шум $N(0, \sigma)$

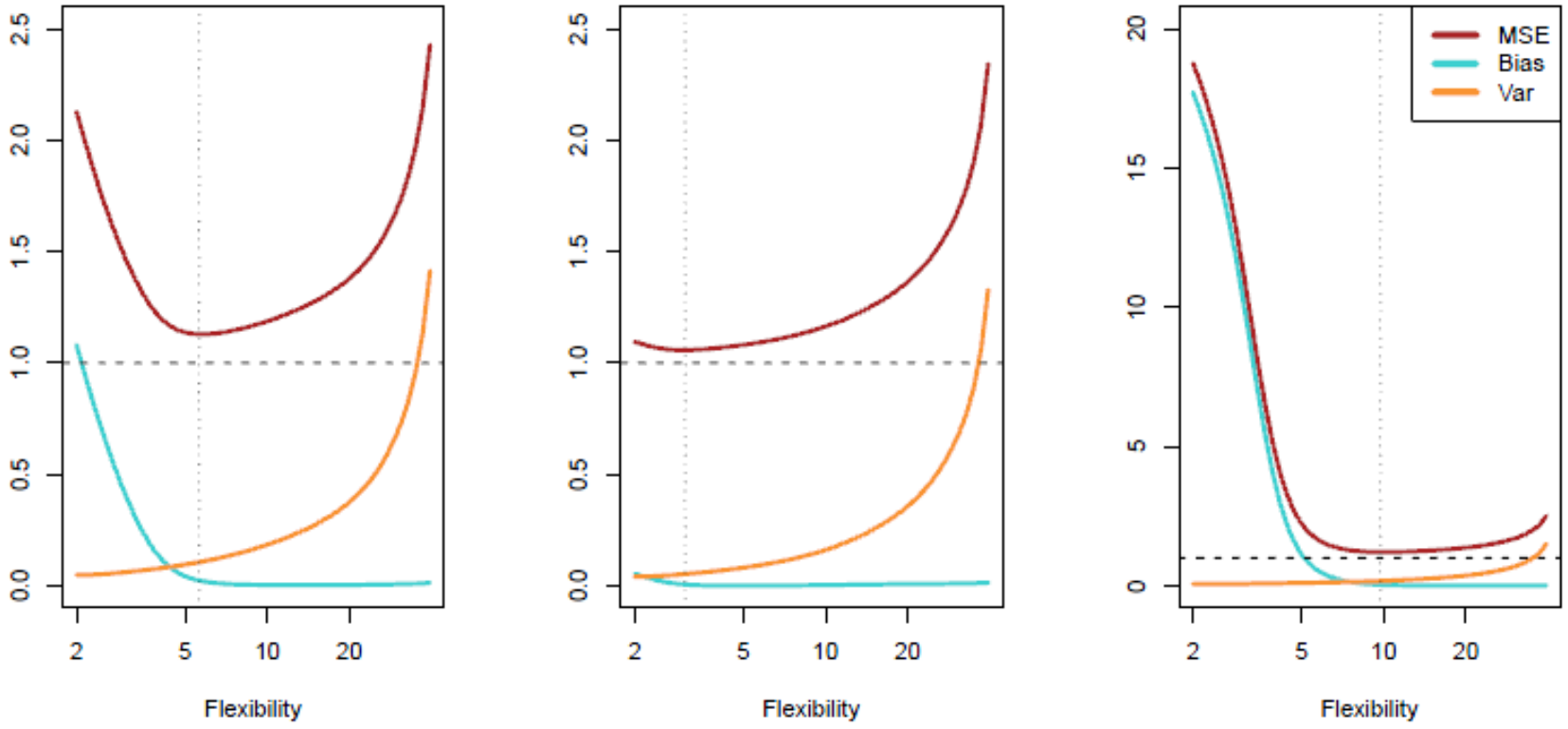
- KNN (k-число соседей):

$$MSE = \sigma^2 + \frac{\sigma^2}{k} + \left[\frac{1}{k}\sum_{i \in N_k(x)} E[a\,(x)] - y(x)\right]^2$$

- Линейная регрессия (p-размерность пространства признаков, l – размер выборки):

$$MSE = \sigma^2 + \frac{p}{l}\sigma^2 + \frac{1}{l}\sum_{x}[E[a\,(x)] - y(x)]^2$$

Компромис отклонения и смещения для трех примеров



Качество на обучающем и тестовом наборе