Машинное обучение и анализ данных

Сложность алгоритмов, переобучение, смещение и разброс

Дьяконов А.Г.

Московский государственный университет имени М.В. Ломоносова (Москва, Россия)

Проблема обобщения

напоминаем... $L(a, X_{\text{train}}) \vee L(a, X_{\text{test}})$

будет ли алгоритм также работать на новых данных?

Не путать с проблемой представительности выборки!

- **данные меняются со временем – предсказываем будущее**
- **другое распределение теста (ЭКГ)**

Считаем, что обучение и контроль одинаково распределены.

Сложность алгоритмов, переобучение, смещение и разброс

Переобучение (overfitting) – явление, когда ошибка на тестовой выборке заметно больше ошибки на обучающей

Это главная проблема машинного обучения!
Если бы её не было $\Rightarrow$ минимизация эмпирического риска

Недообучение (underfitting) – явление, когда ошибка на тестовой выборке достаточно большая (не удаётся «настроиться на выборку»)

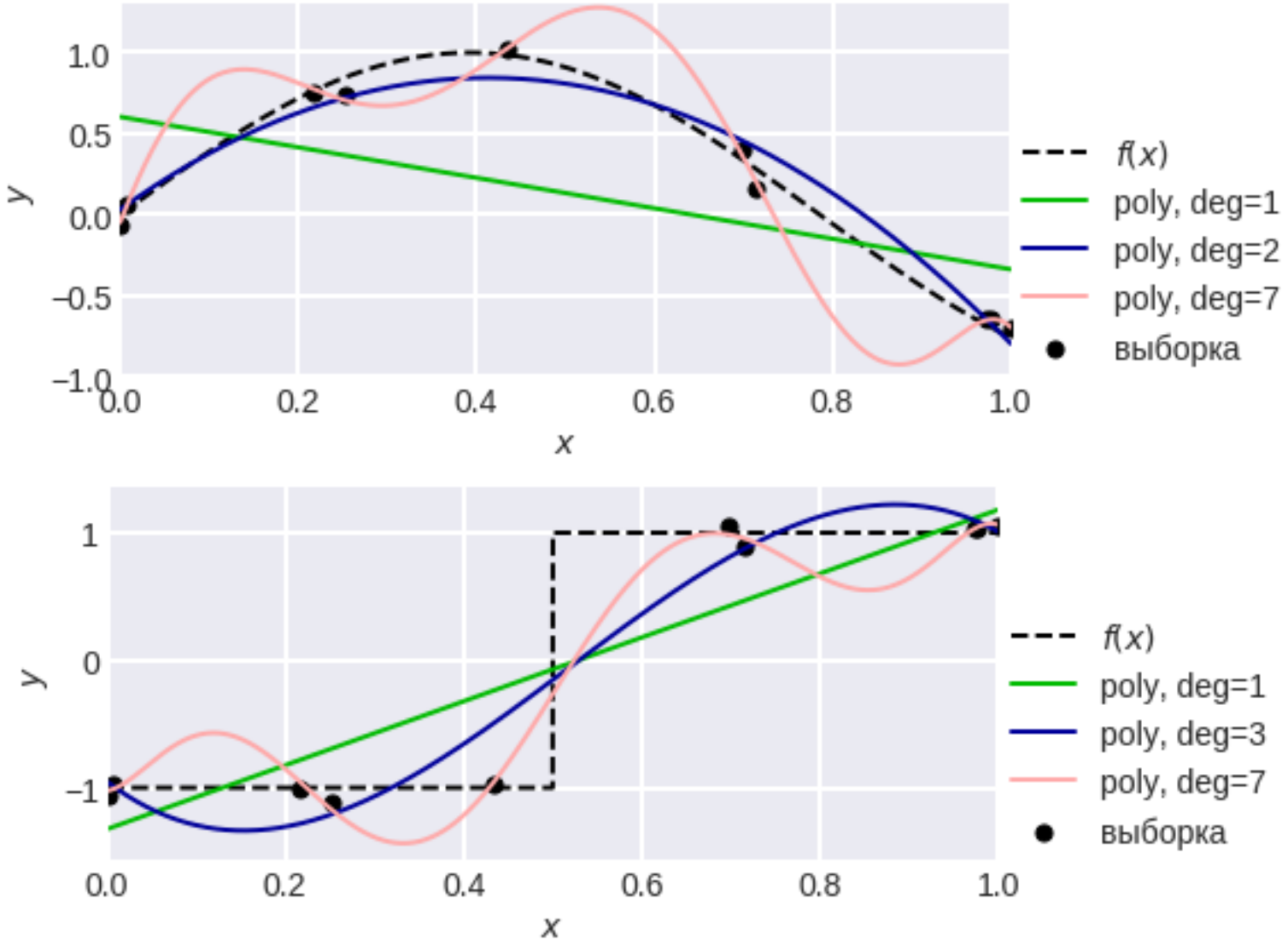
Сложность (complexity/capacity) модели алгоритмов (допускает множество формализаций) – оценивает, насколько разнообразно семейство алгоритмов в модели с точки зрения их функциональных свойств (например, способности настраиваться на выборки).

Повышение сложности решает проблему недообучения и вызывает переобучение.

Проблема

Есть целевая зависимость... известна с точностью до шума

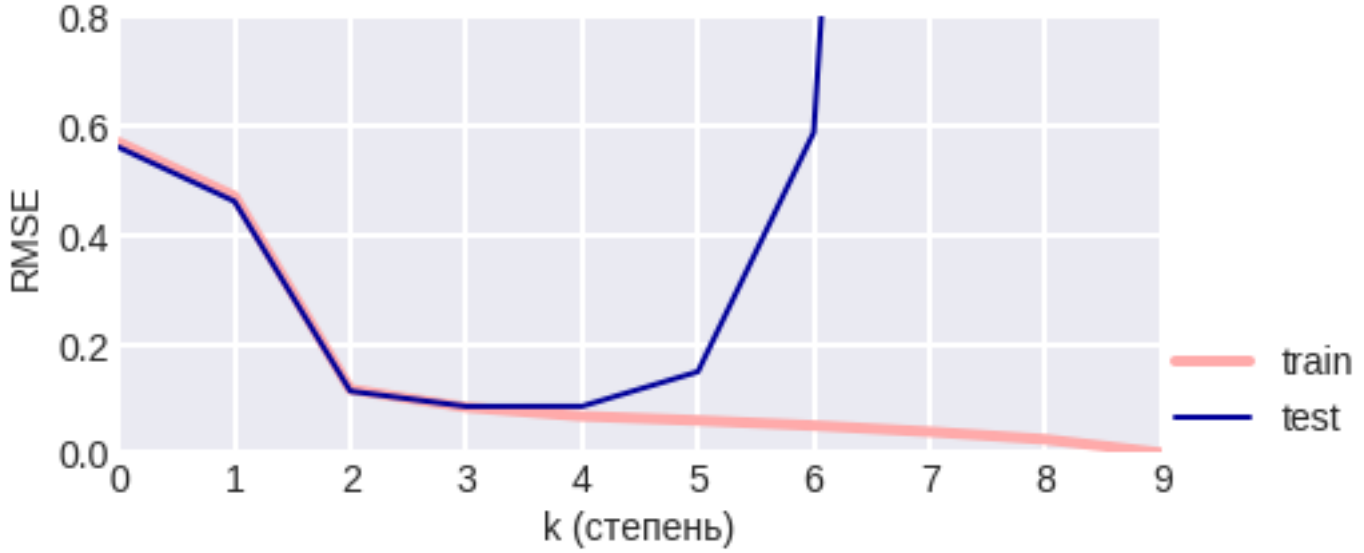
Ищем решение в классе полиномов



Проблема

Полиномы малой степени – недостаточно хорошо описывают данные

Полиномы большой степени – проходят через точки обучения, но явно не похожи на «естественные функции»



При увеличении степени

- **ошибка на обучении падает**
- **ошибка на контроле сначала падает, потом растёт**

Задача регрессии (есть обобщения для классификации)

Пусть

$$y \equiv y(x) = f(x) + \varepsilon, \varepsilon \sim \text{random}(0, \sigma^2)$$
$$a \equiv a(x)$$

тогда

$$\mathrm{E}(y-a)^2 = \mathrm{E}(y^2 + a^2 - 2ya) =$$
$$= \mathrm{E}\,y^2 - (\mathrm{E}\,y)^2 + (\mathrm{E}\,y)^2 + \mathrm{E}\,a^2 - (\mathrm{E}\,a)^2 + (\mathrm{E}\,a)^2 - 2f\,\mathrm{E}\,a =$$
$$= \mathrm{D}\,y + \mathrm{D}\,a + (\mathrm{E}\,y)^2 + (\mathrm{E}\,a)^2 - 2\mathrm{E}\,ya =$$
$$= \mathrm{D}\,y + \mathrm{D}\,a + f^2 + (\mathrm{E}\,a)^2 - 2f\,\mathrm{E}\,a =$$
$$= \mathrm{D}\,y + \mathrm{D}\,a + (\mathrm{E}(f-a))^2 =$$
$$= \sigma^2 + \text{variance}(a) + \text{bias}^2(f,a)$$

Разброс (Variance) – $\mathrm{D}\,a$

Смещение (Bias) – $\mathrm{E}(f-a)$

Задача регрессии

Важно: по чему берётся матожидание

$$\mathrm{E}(y-a)^2 \equiv \mathrm{E}_{(x_i, f(x_i)+\varepsilon_i)_{i=1}^m}(y-a)^2$$

по данным (обучающей выборке)!

Выборки (случайные!) выбираются согласно некоторому распределению (возможно, меняется шум – не зависит от выборки)

$\Rightarrow$ алгоритм a, полученный с помощью обучения на выборке, случаен

Формулу мы получили на конкретном объекте

$$\mathrm{E}(y-a)^2 \equiv \mathrm{E}(y(x)-a(x))^2$$

При желании можно проинтегрировать по всем объектам!

$$\mathrm{E}_D\mathrm{E}_X(y(x)-a_D(x))^2 = \mathrm{E}_X\mathrm{E}_D(y(x)-a_D(x))^2$$

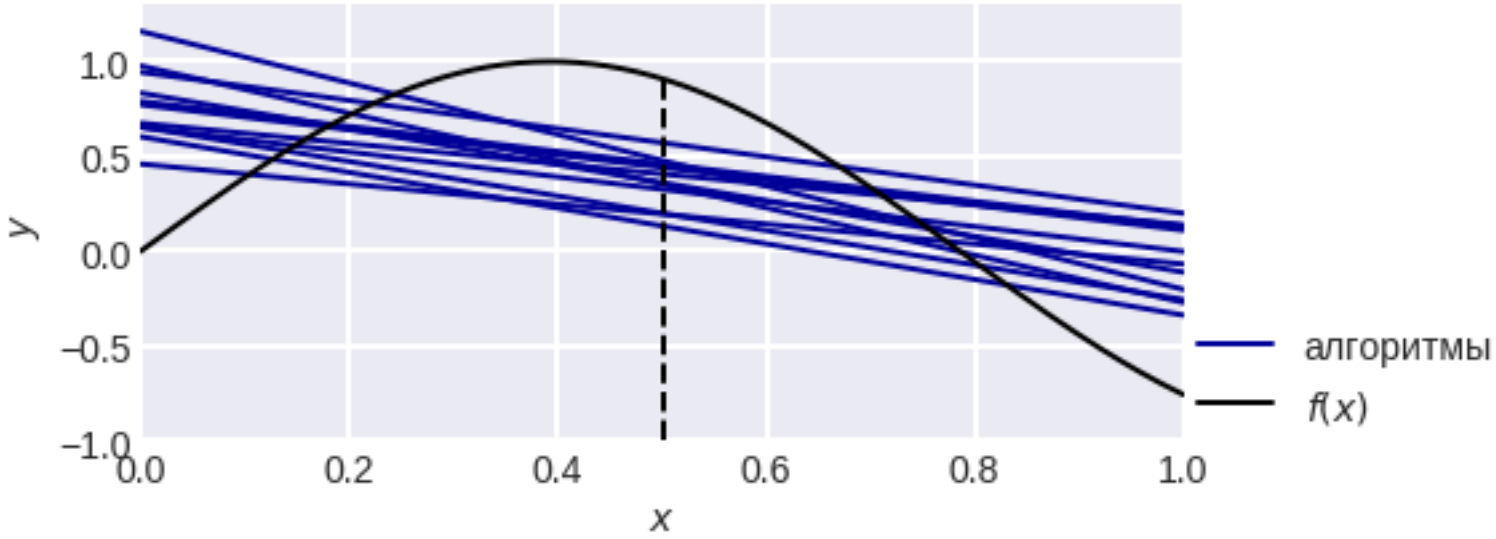
Разброс и смещение

Разброс (Variance) – $\mathrm{D}a$ – разнообразие алгоритмов

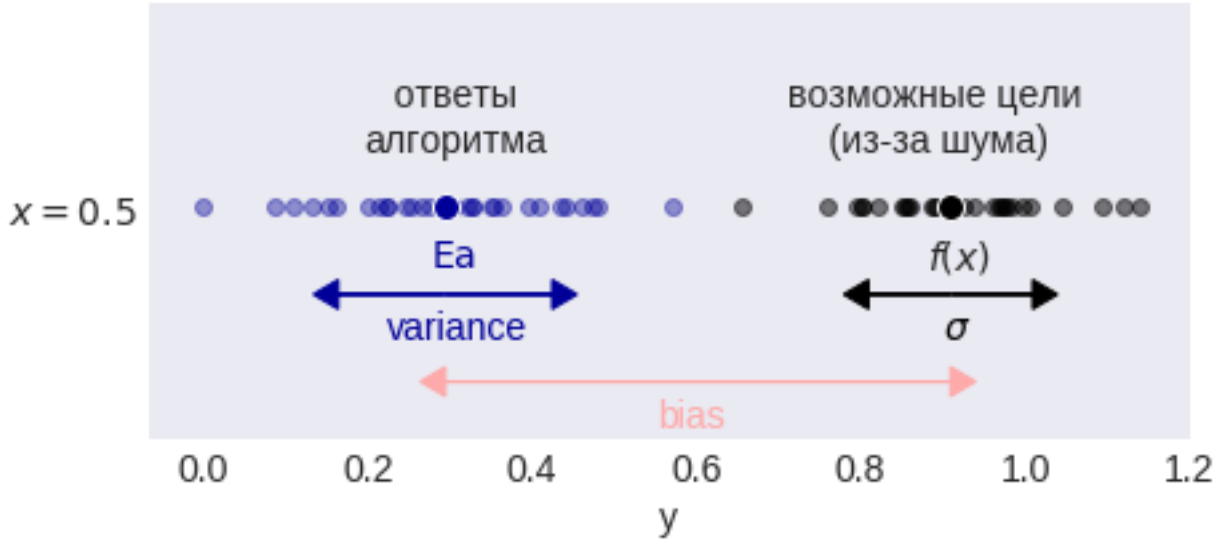
(из-за стохастической природы настройки и/или случайности обучающей выборки, в том числе, шума)

Смещение (Bias) – $\mathrm{E}(f - a)$ – способность модели алгоритмов настраиваться на целевую зависимость

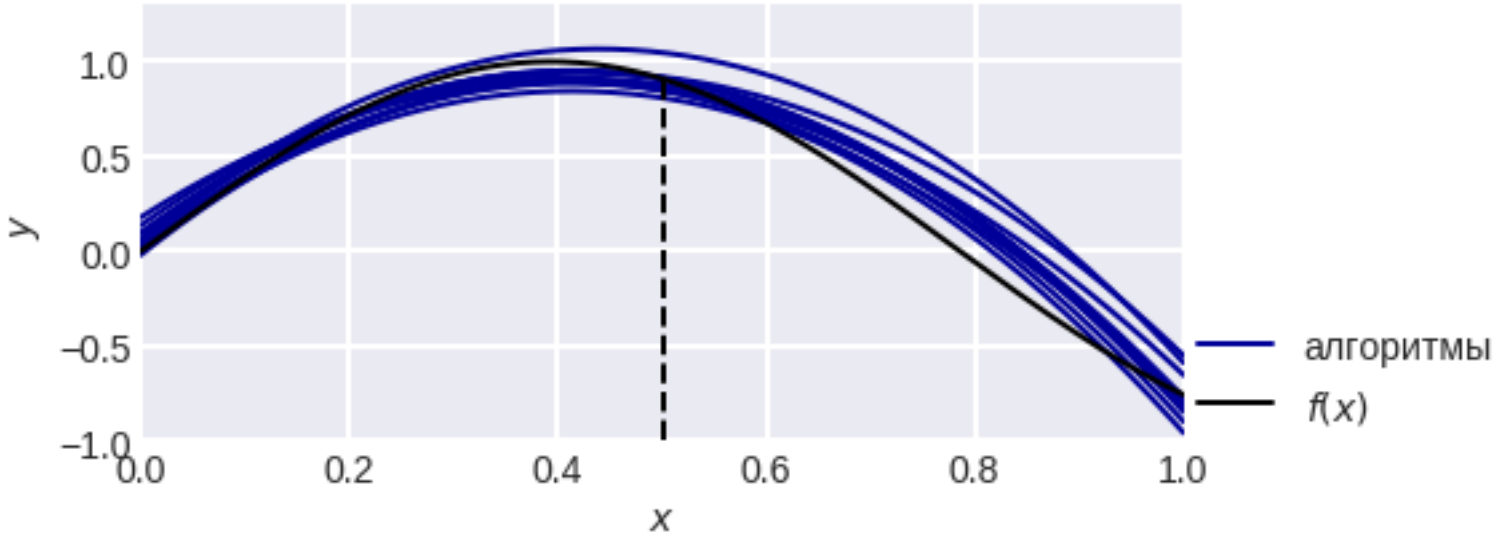
Разброс и смещение

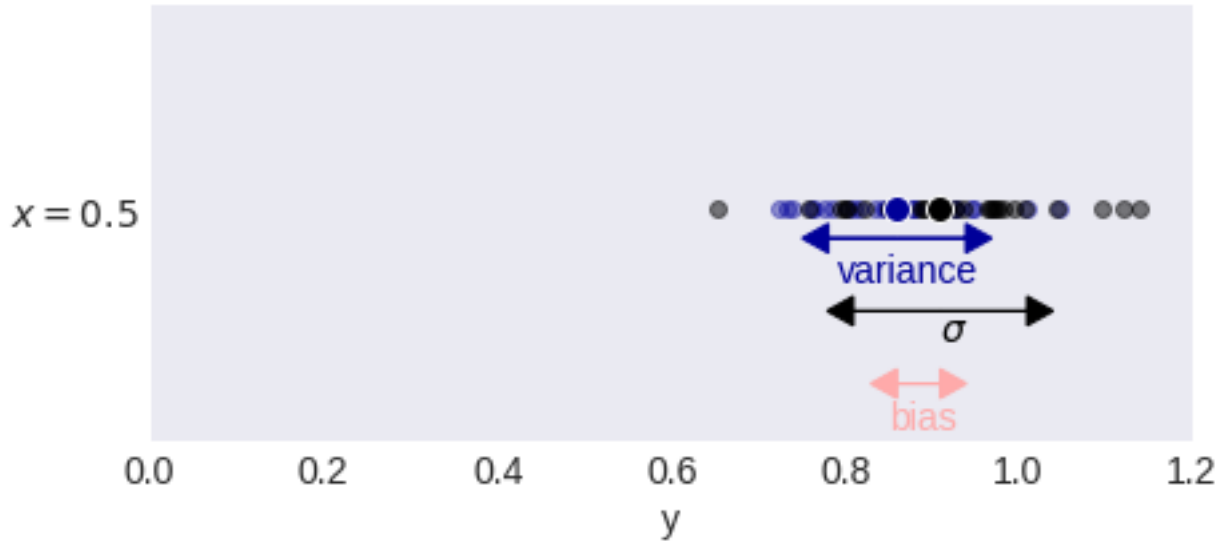


Эксперимент: генерируем разные обучающие выборки...



Разброс и смещение





Разброс и смещение

	Малое смещение Хорошо: настраиваемся на целевую зависимость	Большое смещение Плохо: модель не соответствует данным
Малый разброс Хорошо: Модель устойчива (не зависит от шума в данных)		
Большой разброс Хорошо: найдём алгоритм, который настроится на данных Плохо: слишком сложная модель (много алгоритмов в ней), настраиваемся на шум		

Разные причины ошибок:

«Бедная» модель – не может настроиться на целевую зависимость

«Сложная» модель – может, но не настраивается

(т.к. подвержена переобучению, настраивается на шум)

Большое смещение – часто – гипотеза не соответствует истине

(плохо описывает данные)

Большой разброс – часто – из-за сложности модели

(слишком много разных алгоритмов в ней)

Примеры

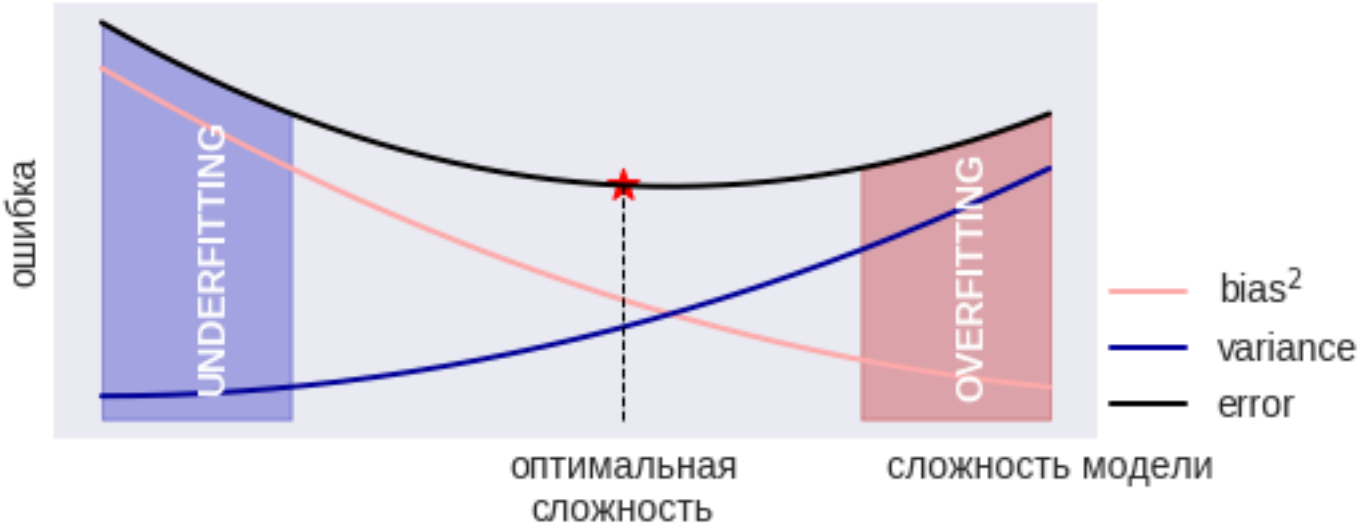
MLE обычно несмещённая оценка, но большой разброс

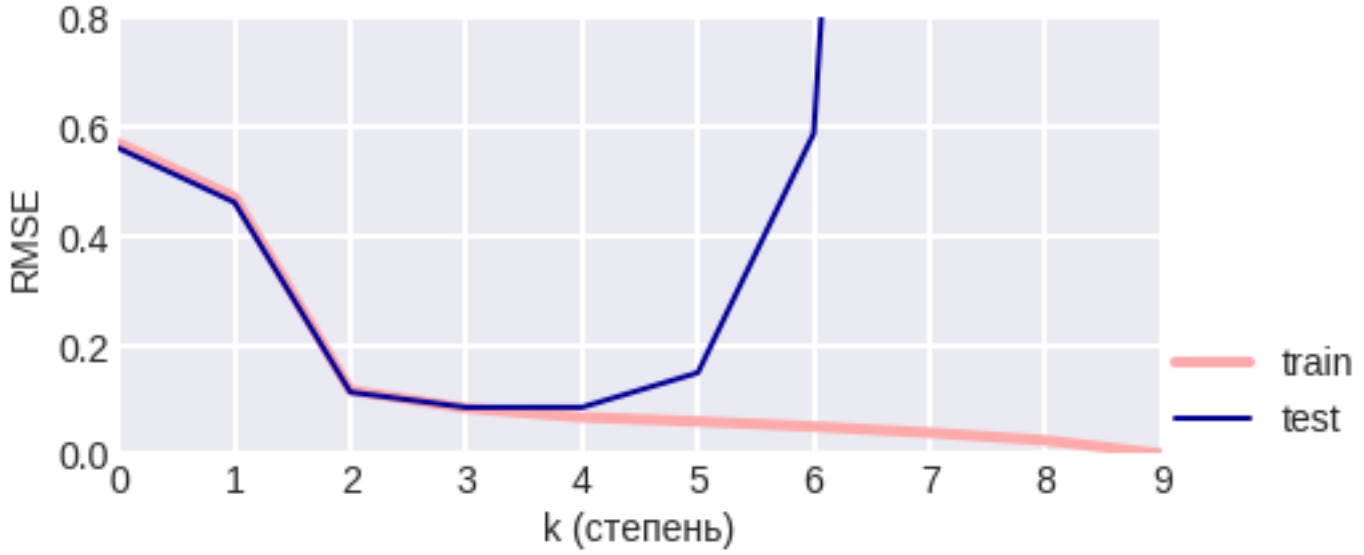
MAP – обычно смещённая, но малый разброс

Такое определение:

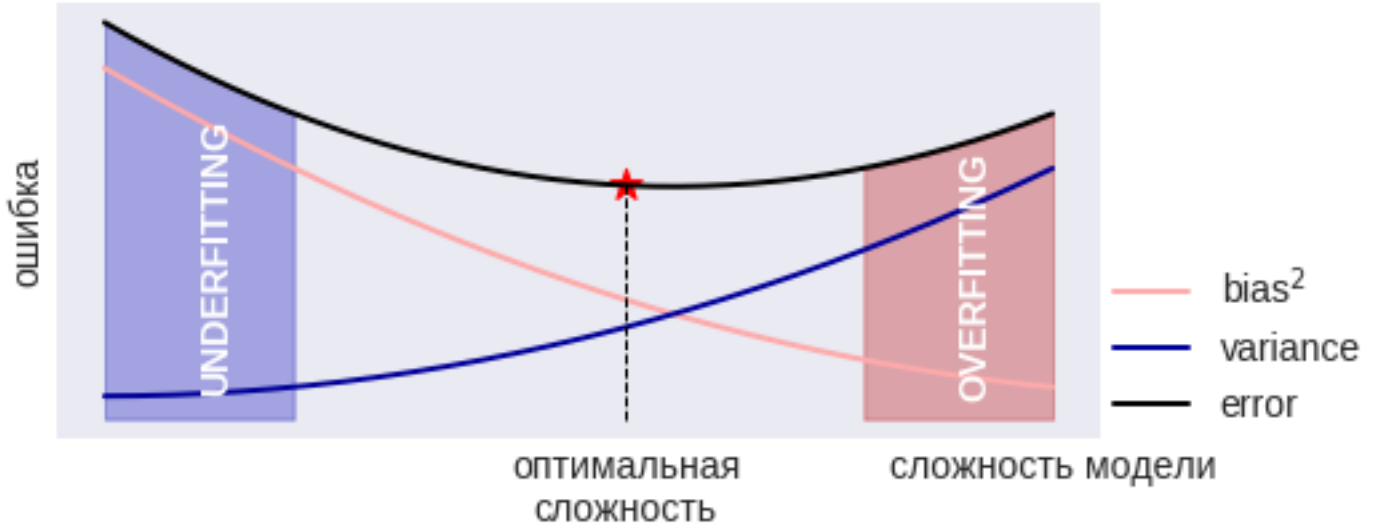
too much variance=overfitting

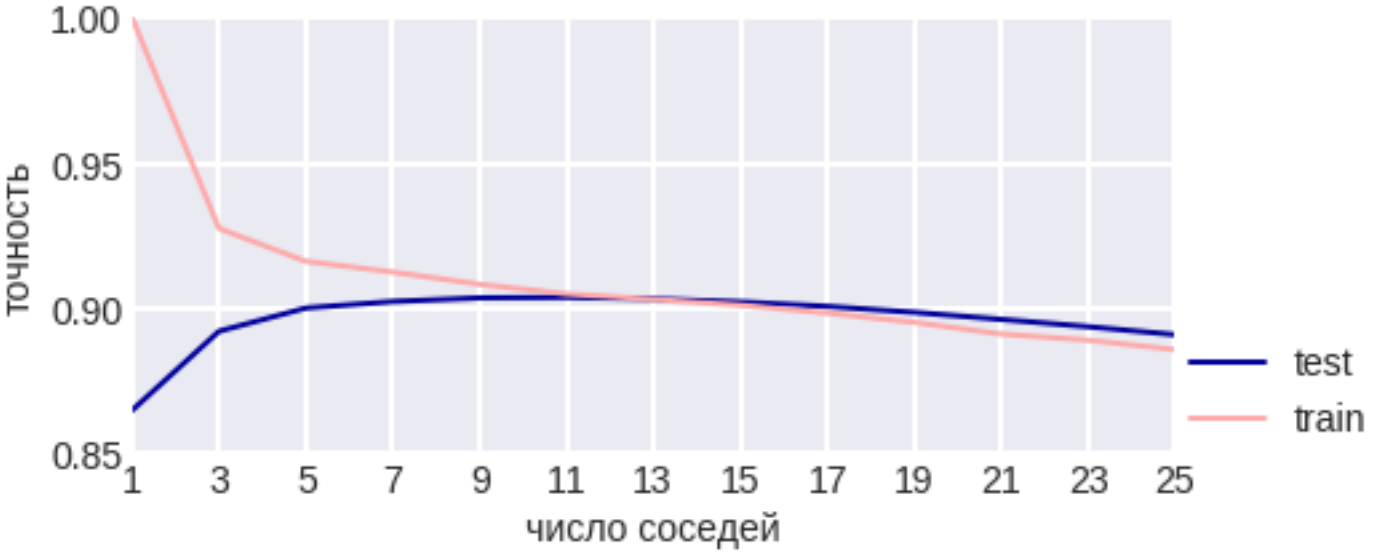
Частая картинка





Частая картинка





тут точность (не ошибка) и сложность ~ 1/(число соседей)

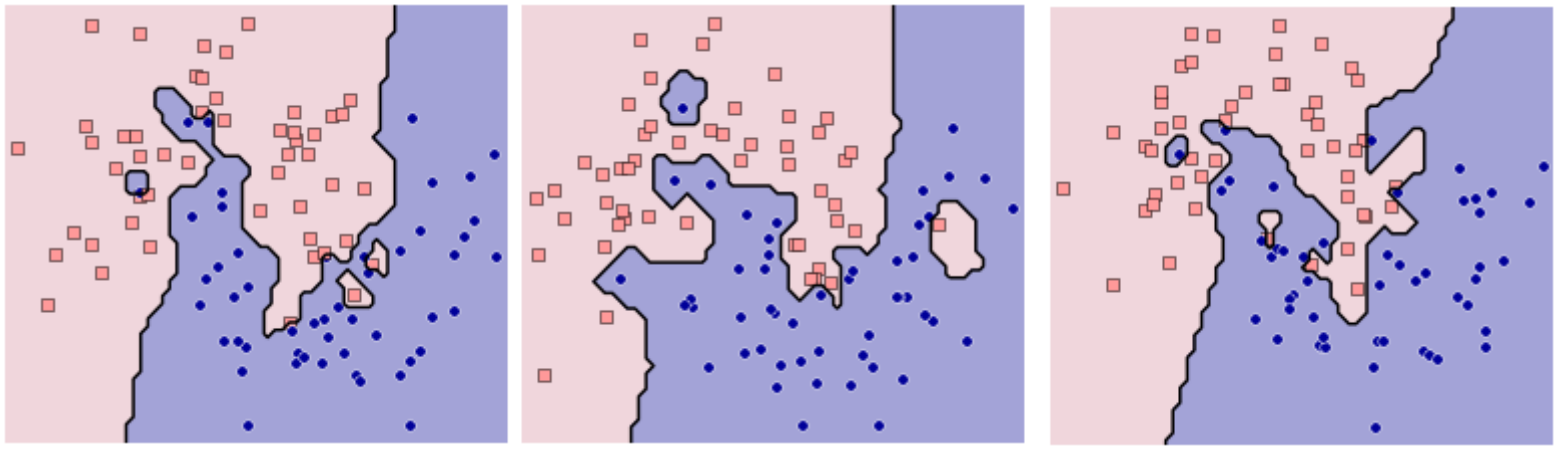
Что же такое сложность?

Подходов к определению много...

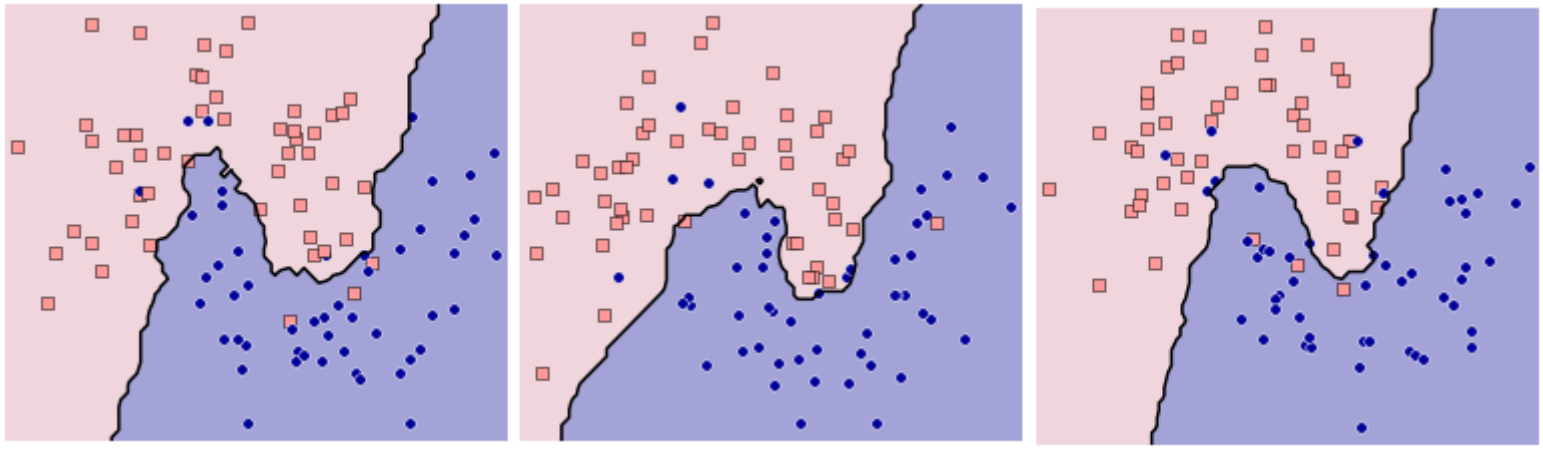
но можно просто ~ (1/variance)

Часто: «ёмкость» (capacity), «способность к обобщению» (representation power)

Почему 1NN сложнее 9NN



Разделяющие поверхности 1NN для разных выборок (одинаково распределённых)



Разделяющие поверхности 9NN для тех же выборок

Результат стабилен!

Почему 1NN сложнее 9NN

Эти алгоритмы имеют

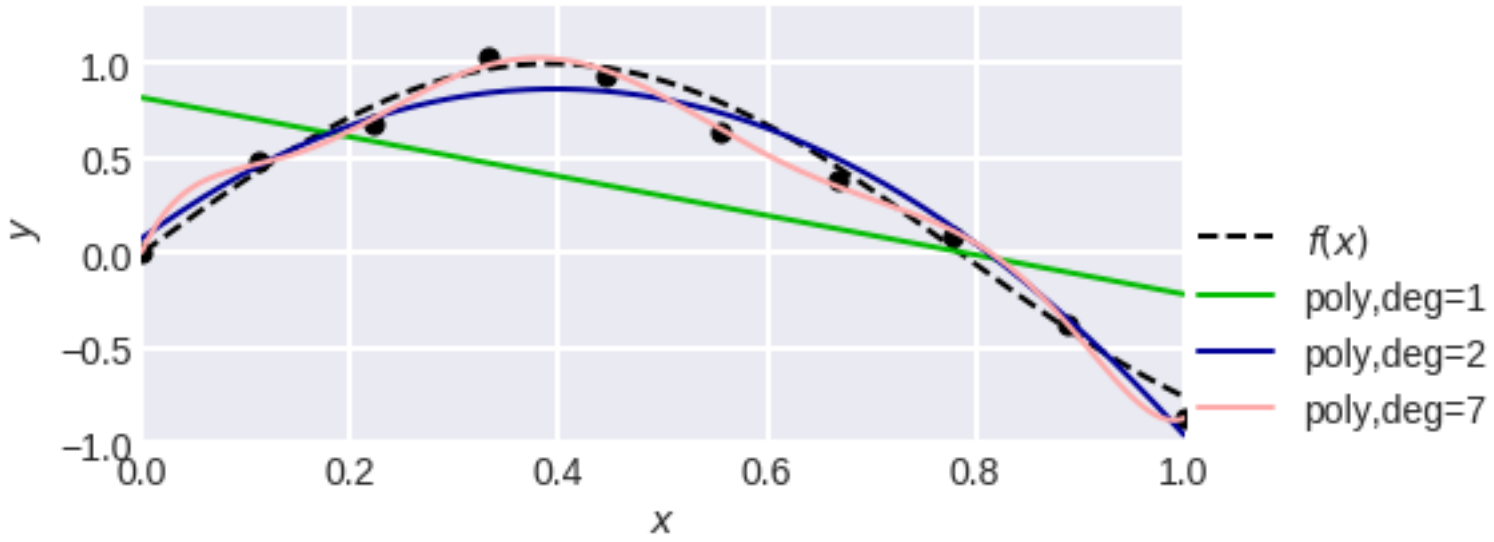
- **одинаковые параметры (что бы не понималось под этим...)**
- **требуют хранения всей обучающей выборки (lazy algorithms)**
- **9NN даже «чуть сложнее в реализации»**

Но разброс у 9NN меньше...

смещения не отличаются???

Способы борьбы с переобучением

1. Выборка специальной структуры

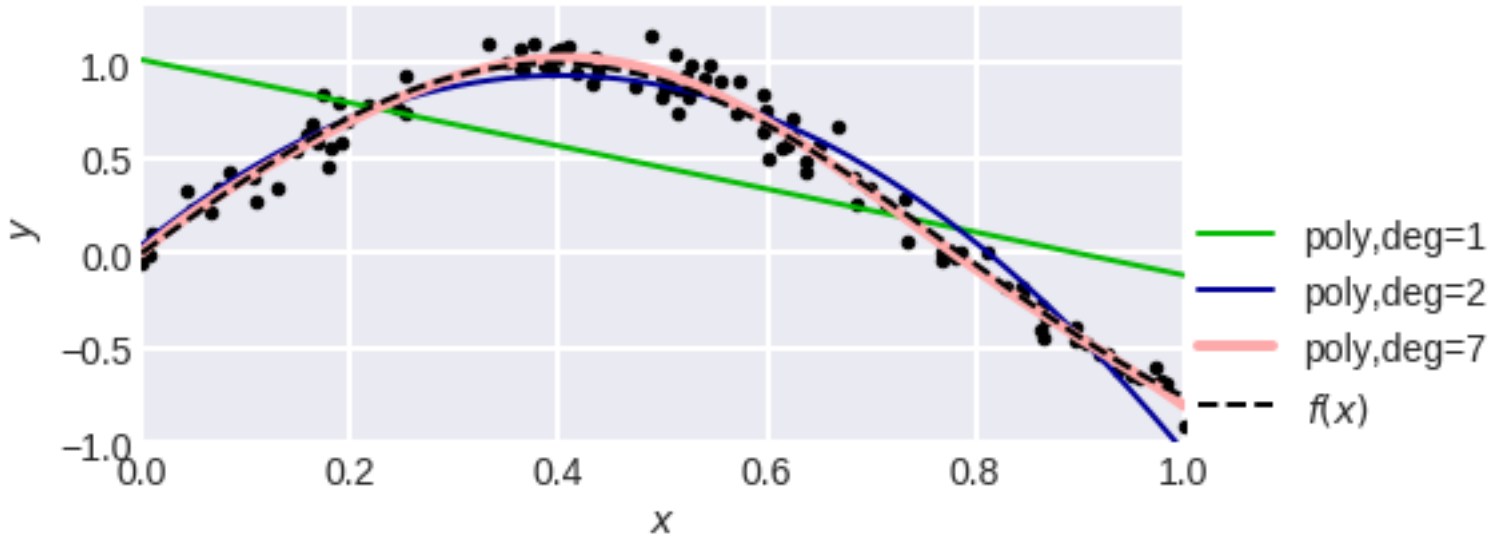


Даже при наличии шума, если есть возможность «формировать выборку», это можно сделать так, чтобы уменьшить переобучение

Выбор специальных данных (ex: которые обманывают алгоритм)

Способы борьбы с переобучением

2. Увеличение объёма данных

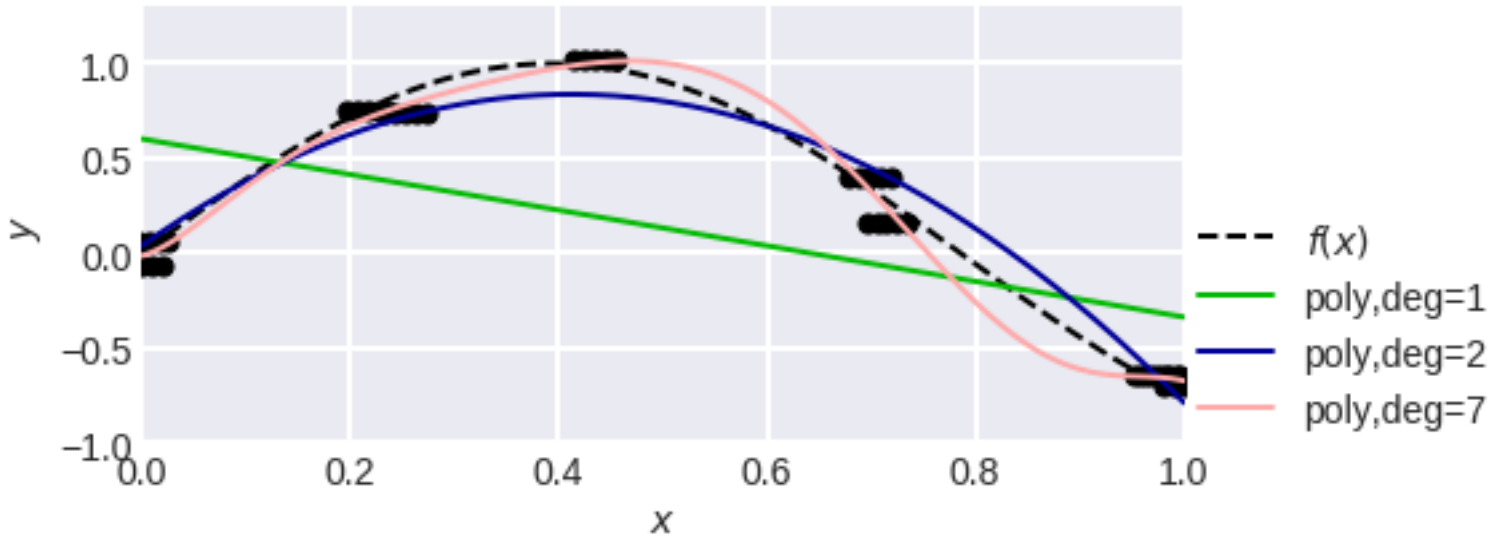


Данные первичны, алгоритмы вторичны!

Но чтобы сложные алгоритмы не переобучались нужны действительно большие объёмы.

Способы борьбы с переобучением

1+2=3. «Аугментация»

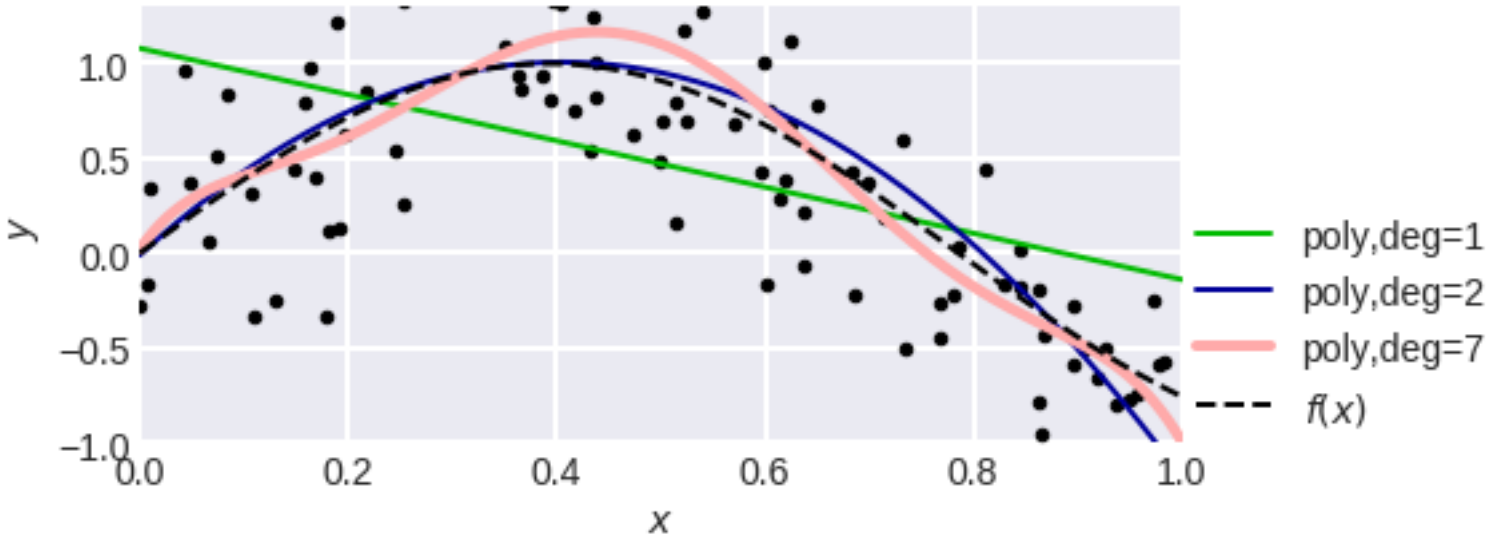


Искусственное увеличение выборки так,
чтобы алгоритм удовлетворял требуемым свойствам

Частый приём: внесение шума в данные
В нейросетях м.б. добавления шума в промежуточные слои!
Иногда: в целевой признак.

Способы борьбы с переобучением

4. Улучшение качества данных



шум

выбросы

аномальные дубликаты и пропуски

Это больше влияет на ошибку ε **в** $y(x) = f(x) + \varepsilon$**!**

Способы борьбы с переобучением

5. Использование других данных / задач / готовых моделей

Как правило в DL, где модели сложные

1) нейросеть можно обучить на аналогичной задаче

ex.: другая задача классификации

ex.: такая же задача, но данные на другом оборудовании

ex.: синтетические данные (в сегментации)

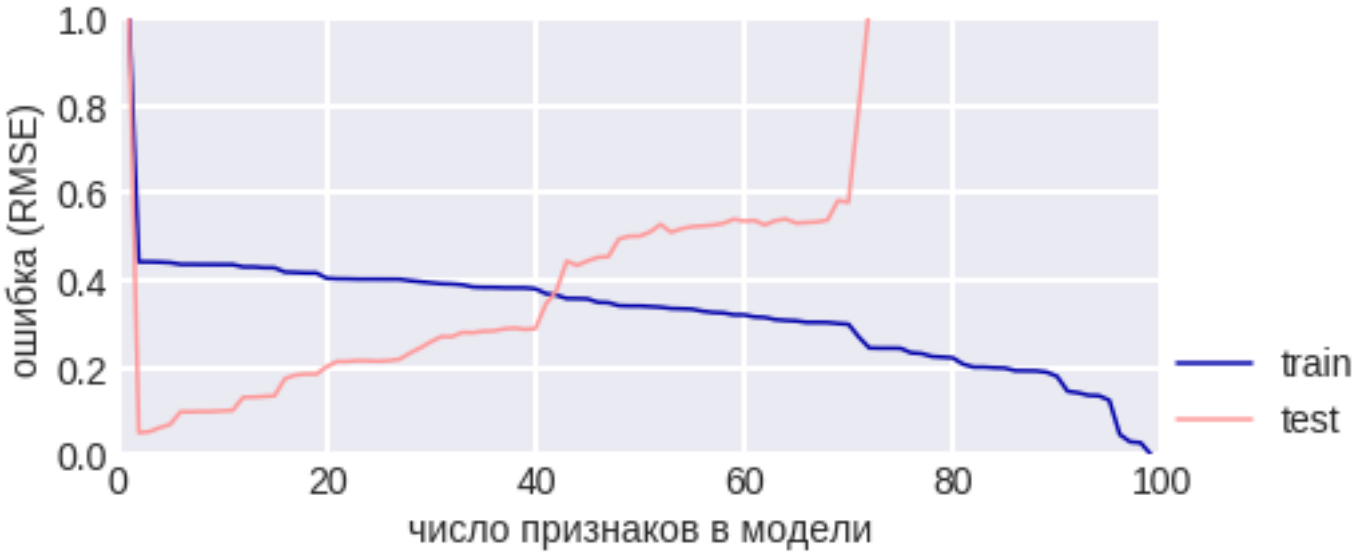
2) можно взять уже обученную (на другой задаче) нейросеть и дообучить её

6. Сокращение размерности, отбор признаков

(тоже формально про данные)

Почему много признаков – плохо

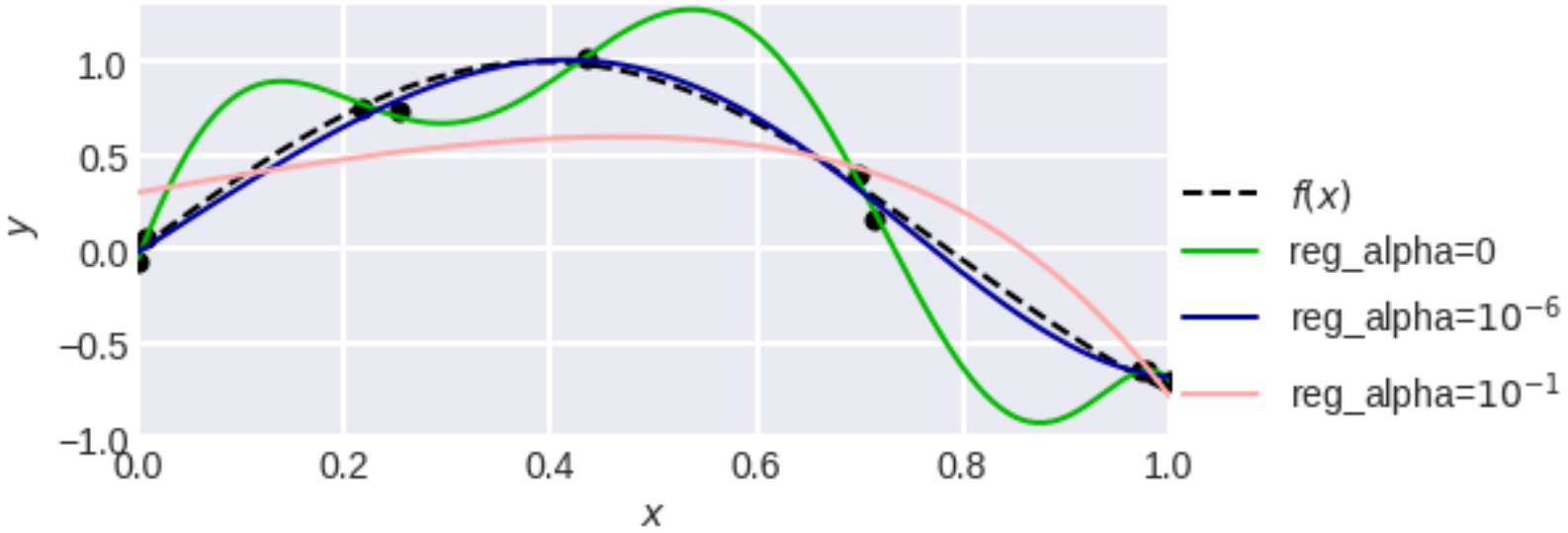
$$m = n = 100,\ y = X_1 - X_2 + \text{norm}(0, 0.5),\ X_i = \text{norm}(0, 1)$$



Способы борьбы с переобучением

7. Регуляризация

До этого говорили про данные, теперь про алгоритмы...



Уменьшение сложности модели!

Изменение настройки модели

здесь: добавление штрафующего слагаемого в опт. функционал

Способы борьбы с переобучением

7. Регуляризация

- **Добавление штрафующего слагаемого к минимизируемому функционалу**

$$(y(x) - f(x \mid w))^2 + \lambda \| w \|^p \to \min$$

обоснование в MLE

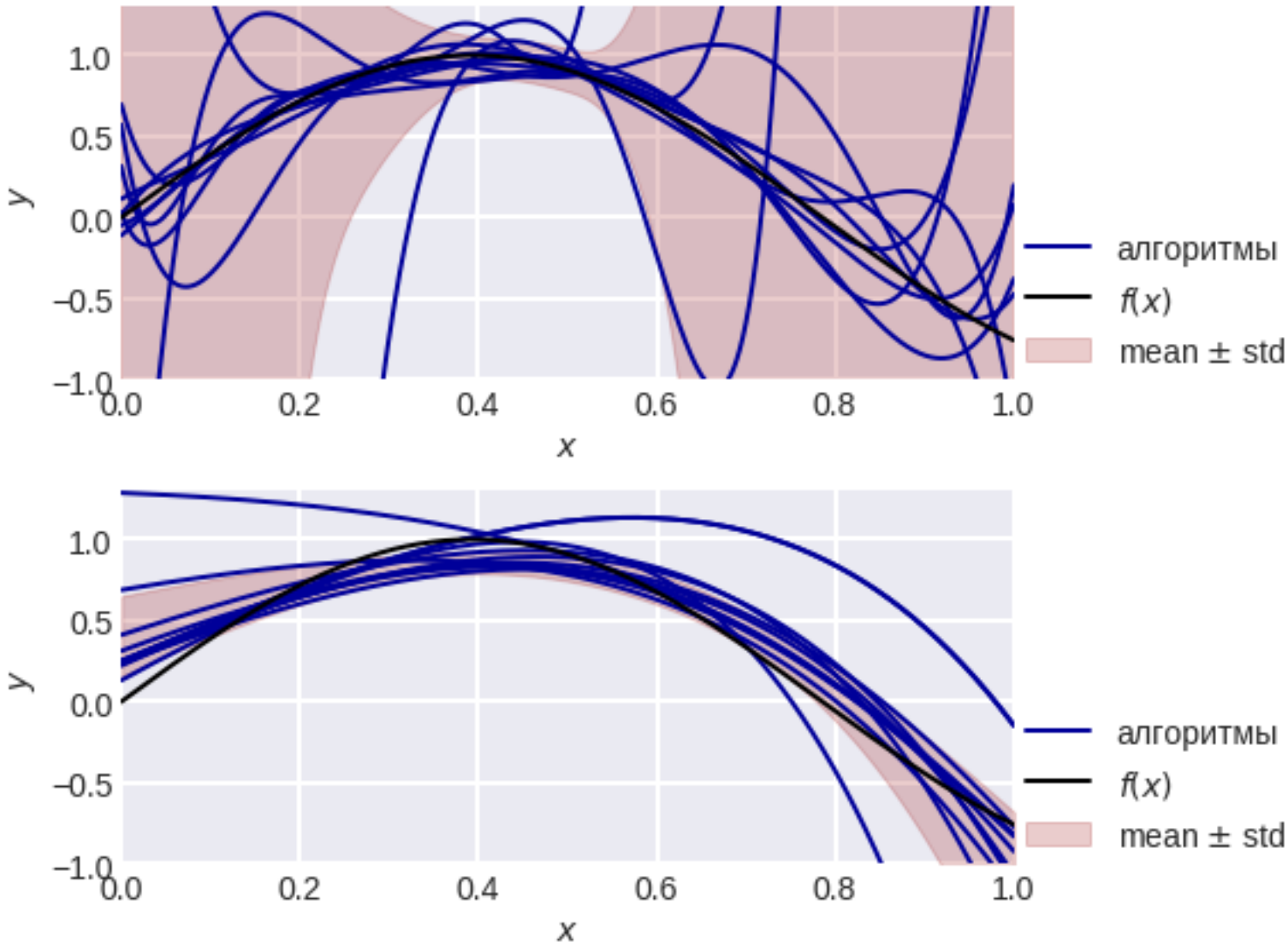
- **Разреженные представления**

(зануления весов, выходов нейронов)

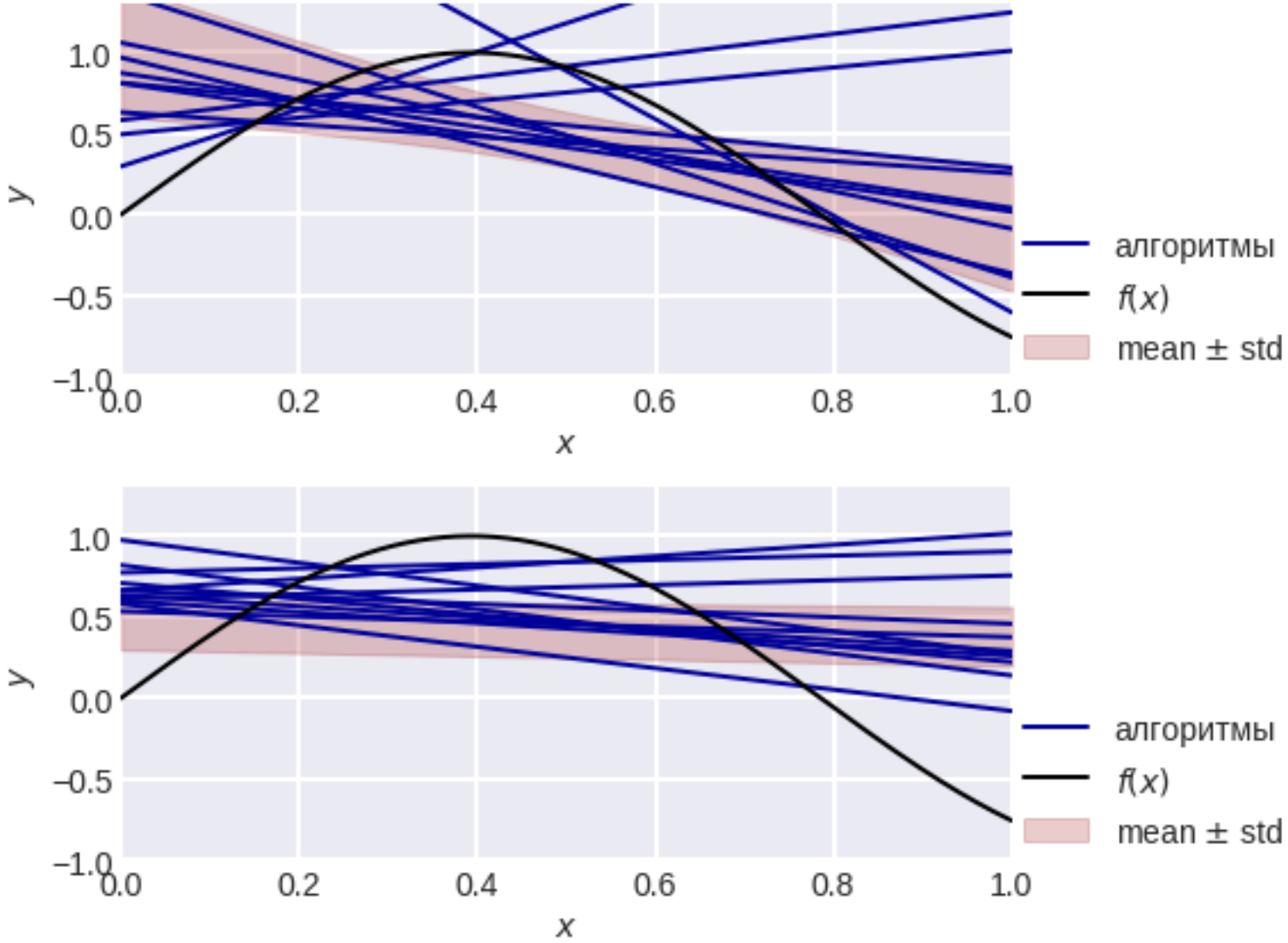
- **Прореживание (Drop Out)**
- **Подрезка деревьев (Pruning)**
- **Разделение параметров (Parameter Sharing)**

Тренируем НС требуя, чтобы значения её параметров были также близки к параметрам другой НС, обученной без учителя

Пример регуляризации: до и после



Пример регуляризации: до и после



Способы борьбы с переобучением

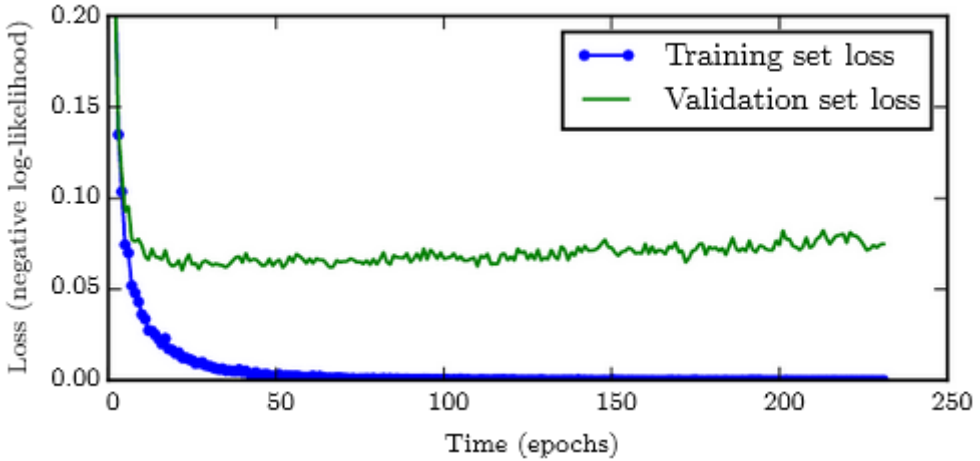
8. Организация контроля – самое важное!

hold out, CV, и т.п.

- **ранняя остановка (early stopping)**

обучение НС, бустинг – где есть итерации

используем отложенный контроль



[DLbook]

В модельной ситуации ES эквивалентна L2-регуляризации

Способы борьбы с переобучением

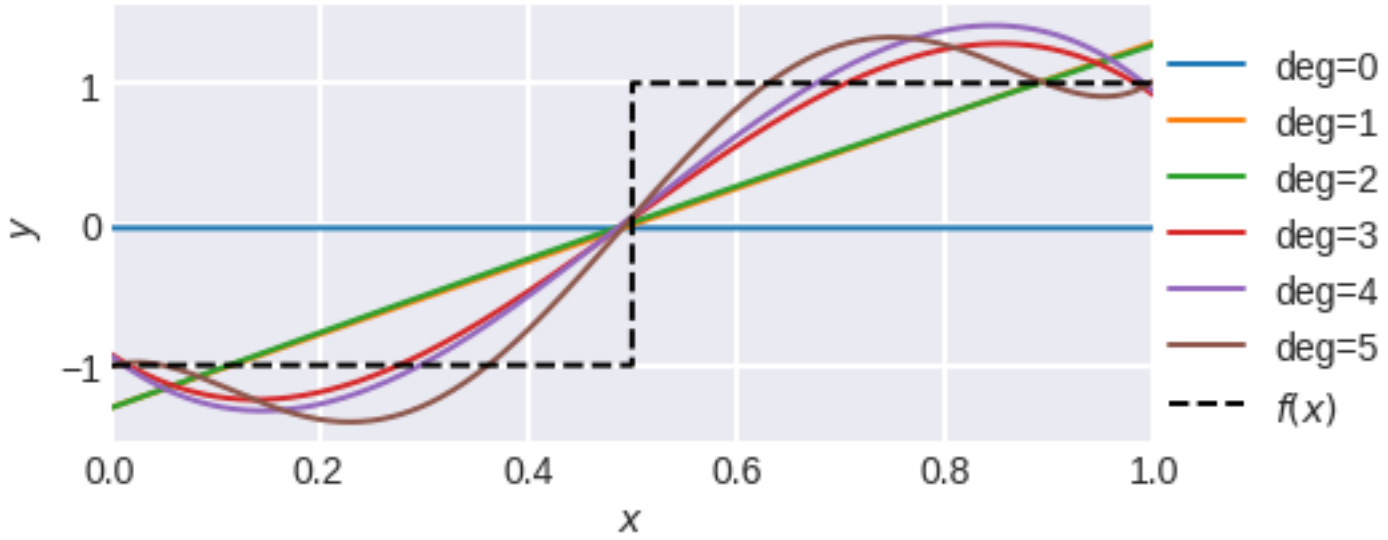
9. Выбор архитектуры алгоритма

Пример: свёртки, где есть инвариантность
(+ сокращает число параметров)
Пример: усреднение, бэгинг

в отличие от уменьшения сложности (см. раньше) тут сразу выбираем простое/специально устроенное и т.п.

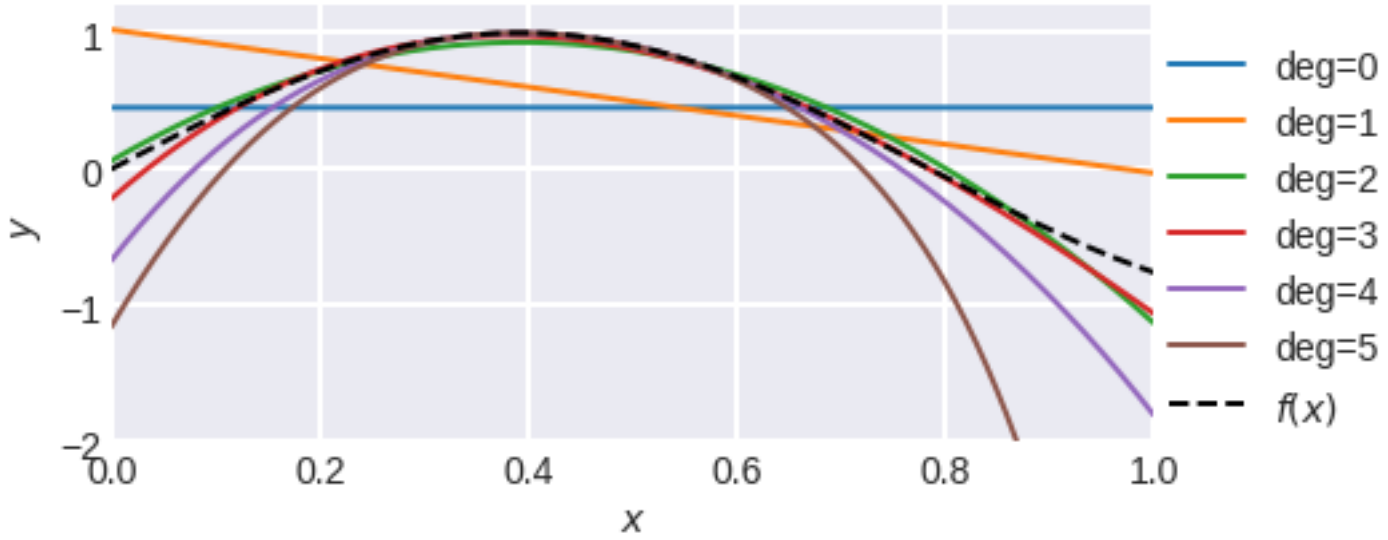
- **batch normalization**

В чём нас обманывают...

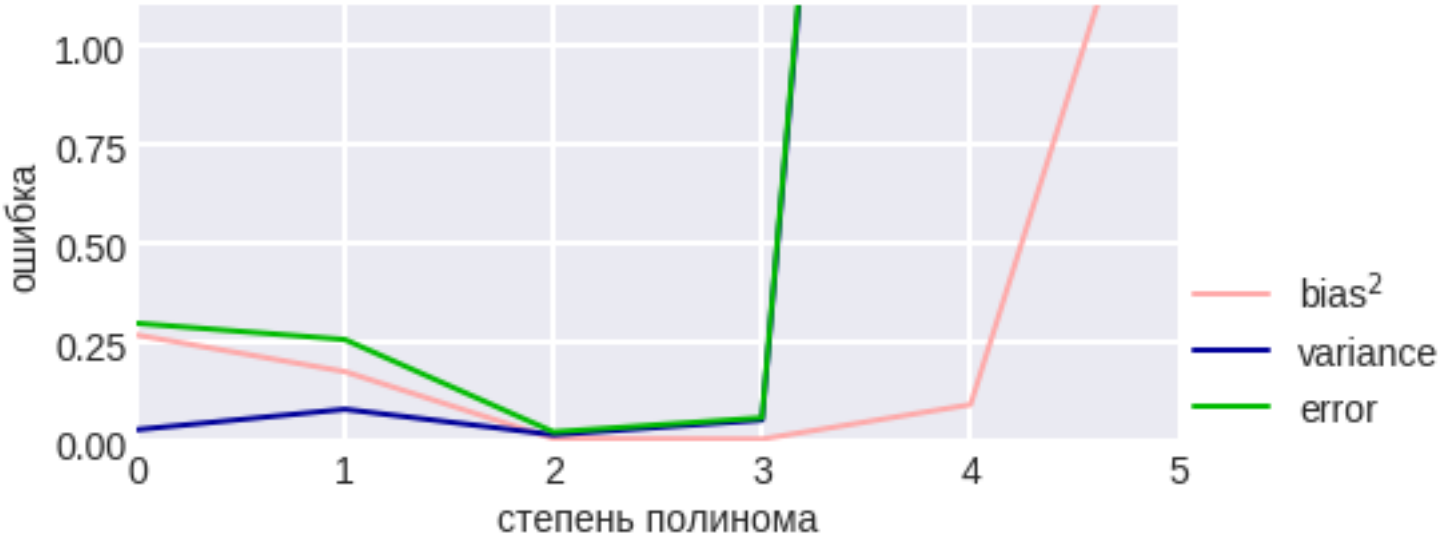


Это матожидания наших полиномиальных моделей!

В чём нас обманывают...



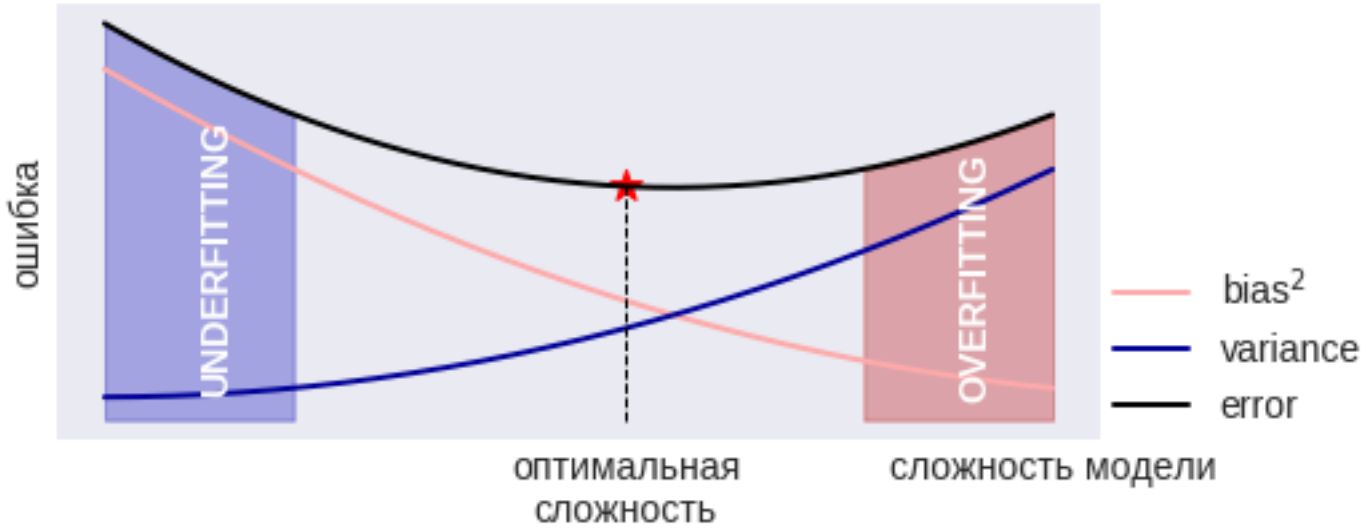
Полиномы 2й степени «самые лёгкие...»



В чём нас обманывают...

Степень полинома – «естественная» мера его сложности

Но тогда классической картинки мы не видим!



- **общая ошибка может быть слегка неунимодальной**
- **смещение и разброс могут не быть строго монотонными!**
- **смещение может возрастать при увеличении сложности!**

Может быть (и это нормально)

сложность модели относительно данных!

Вспомним... сложность реализации схемы в конкретном базисе