Статистическая теория принятия решений

Сергей Николенко

Казанский Федеральный Университет, 2014

Outline

Метод ближайших соседей

- Линейная модель – очень сильные предположения, много точек не нужно.
- Совсем другой подход – давайте вообще никаких предположений не делать (это не совсем так, конечно :)), а будем отталкиваться от данных.
- Давайте не будем строить вообще никакой модели, а будем классифицировать новые примеры как

$$\hat{y}(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x})} y_i,$$

где $N_k(\mathbf{x})$ – множество k ближайших соседей точки $\mathbf{x}$ среди имеющихся данных $(\mathbf{x}_i, y_i)_{i=1}^N$.

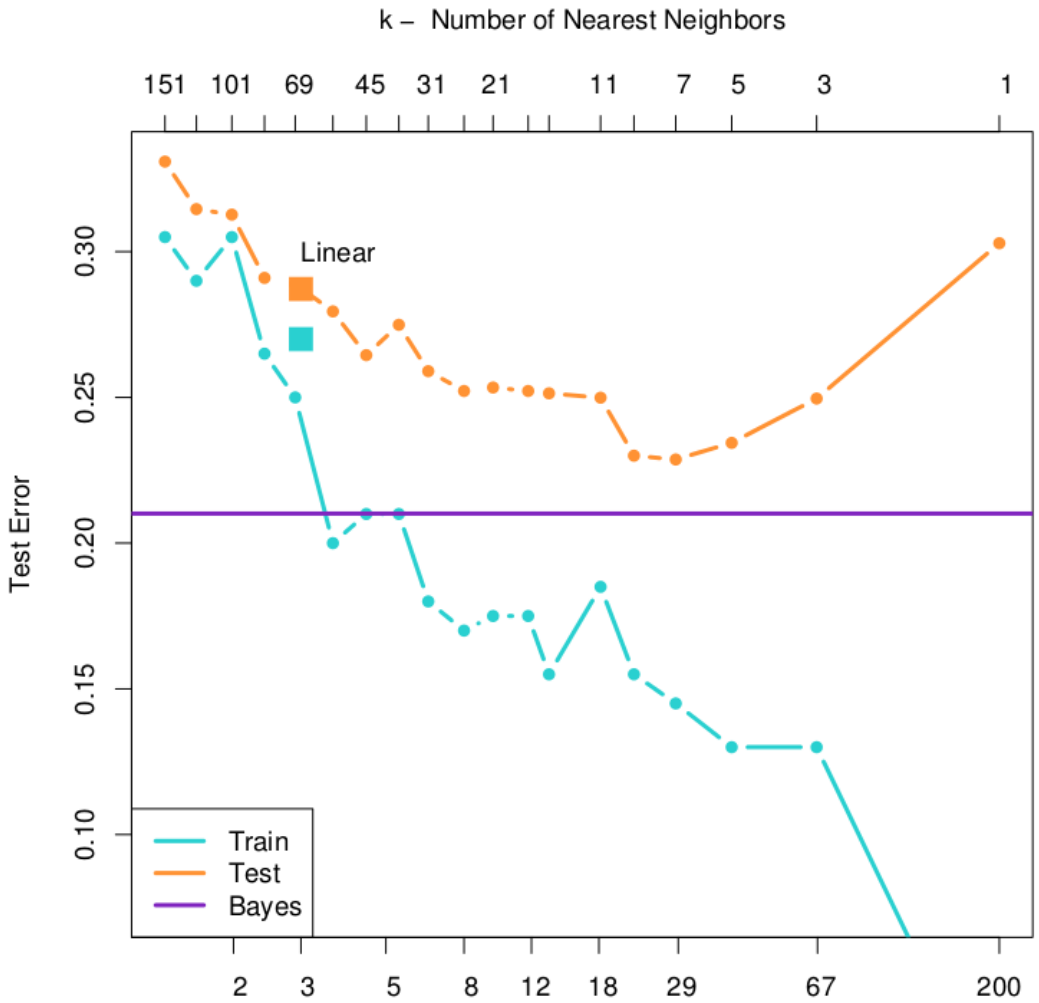
Метод ближайших соседей

- Снова смотрим на примеры – теперь появился параметр k, от которого многое зависит.
- Для разумно большого k у нас в нашем примере стало меньше ошибок.
- Но это не предел – для $k = 1$ на тестовых данных вообще никаких ошибок нету!
- Что это значит? В чём недостаток метода ближайших соседей при $k = 1$?
- Сколько параметров у метода k-NN?
- Как выбрать k? Можно ли просто подсчитать среднеквадратическую ошибку и минимизировать её?

Метод ближайших соседей

- На самом деле данные были порождены так:
  - сначала по распределению $\mathcal{N}((1,0)^\top, \mathbf{I})$ породили 10 синих средних;
  - потом по распределению $\mathcal{N}((0,1)^\top, \mathbf{I})$ породили 10 красных средних;
  - потом для каждого из классов сгенерировали по 100 точек так: выбрать одно из 10 средних m_k равномерно (с вероятностью $\frac{1}{10}$), потом породили точку $\mathcal{N}(m_k, \frac{1}{5}\mathbf{I})$.
- Получилось, что мы разделяем две смеси гауссианов.

Качество метода K-NN



Функция потери

- Сейчас мы попытаемся понять, что же на самом деле происходит в этих методах.
- Начнём с настоящей регрессии – непрерывный вещественный вход $\mathbf{x} \in \boldsymbol{R}^p$, непрерывный вещественный выход $y \in \boldsymbol{R}$; у них есть некоторое совместное распределение $p(\mathbf{x}, y)$.
- Мы хотим найти функцию $f(\mathbf{x})$, которая лучше всего предсказывает y.

Функция потери

- Введём *функцию потери* (loss function) $L(y, f(\mathbf{x}))$, которая наказывает за ошибки; естественно взять квадратичную функцию потери

$$L\left(y, f(\mathbf{x})\right) = \left(y - f(\mathbf{x})\right)^2.$$

- Тогда каждому f можно сопоставить *ожидаемую ошибку предсказания* (expected prediction error):

$$\mathrm{EPE}(f) = \mathrm{E}(y - f(\mathbf{x}))^2 = \int\int (y - f(\mathbf{x}))^2 p(\mathbf{x}, y) d\mathbf{x} dy.$$

- И теперь самая хорошая функция предсказания $\hat{f}$ – это та, которая минимизирует $\mathrm{EPE}(f)$.

Функция потери

- Это можно переписать как

$$\mathrm{EPE}(f) = \mathrm{E}_{\mathbf{x}}\mathrm{E}_{y|\mathbf{x}}\left[(y - f(\mathbf{x}))^2 \mid \mathbf{x}\right],$$

и, значит, можно теперь минимизировать EPE поточечно:

$$\hat{f}(\mathbf{x}) = \arg\min_c \mathrm{E}_{y|\mathbf{x}'}\left[(y - c)^2 \mid \mathbf{x}' = \mathbf{x}\right],$$

а это можно решить и получить

$$\hat{f}(\mathbf{x}) = \mathrm{E}_{y|\mathbf{x}'}(y \mid \mathbf{x}' = \mathbf{x}).$$

- Это решение называется *функцией регрессии* и является наилучшим предсказанием y в любой точке $\mathbf{x}$.

k-NN

- Теперь мы можем понять, что такое k-NN.
- Давайте оценим это ожидание:

$$f(\mathbf{x}) = \mathbb{E}_{y|\mathbf{x}'}(y \mid \mathbf{x}' = \mathbf{x}).$$

- Оценка ожидания – это среднее всех y с данным $\mathbf{x}$. Конечно, у нас таких нету, поэтому мы приближаем это среднее как

$$\hat{f}(\mathbf{x}) = \text{Average}\left[y_i \mid \mathbf{x}_i \in N_k(\mathbf{x})\right].$$

- Это сразу два приближения: ожидание через среднее и среднее в точке через среднее в ближних точках.
- Иначе говоря, k-NN предполагает, что в окрестности $\mathbf{x}$ функция $y(\mathbf{x})$ не сильно меняется, а лучше всего – она кусочно-постоянна.

Линейная регрессия

- А линейная регрессия – это модельный подход, мы предполагаем, что функция регрессии линейна от своих аргументов:

$$f(\mathbf{x}) \approx \mathbf{x}^\top \mathbf{w}.$$

- Теперь мы не берём условие по $\mathbf{x}$, как в k-NN, а просто собираем много значений для разных $\mathbf{x}$ и обучаем модель.

Классификация

- То же самое можно и с задачей классификации сделать. Пусть у нас переменная g с K возможными значениями $g_1, \ldots, g_k$ предсказывается.
- Введём функцию потери, равную 1 за каждый неверный ответ. Получим

$$\mathrm{EPE} = \mathrm{E}\left[L(g, \hat{g}(\mathbf{x}))\right].$$

- Перепишем как раньше:

$$\mathrm{EPE} = \mathrm{E}_{\mathbf{x}} \sum_{k=1}^{K} L(g_k, \hat{g}(\mathbf{x})) p(g_k \mid \mathbf{x}).$$

- Опять достаточно оптимизировать поточечно:

$$\hat{g}(\mathbf{x}) = \arg\min_g \sum_{k=1}^{K} L(g_k, \hat{g}(\mathbf{x})) p(g_k \mid \mathbf{x}).$$

Классификация

- Опять достаточно оптимизировать поточечно:

$$\hat{g}(\mathbf{x}) = \arg\min_g \sum_{k=1}^{K} L(g_k, \hat{g}(\mathbf{x}))p(g_k \mid \mathbf{x}).$$

- Для 0-1 функции потери это упрощается до

$$\hat{g}(\mathbf{x}) = \arg\min_g \left[1 - p(g \mid \mathbf{x})\right], \text{ т.е.}$$

$$\hat{g}(\mathbf{x}) = g_k, \text{ если } p(g_k \mid \mathbf{x}) = \max_g p(g \mid \mathbf{x}).$$

- Это называется *оптимальным байесовским классификатором*; если модель известна, то его обычно можно построить.

Классификация

- Байесовский классификатор: $\hat{g}(\mathbf{x}) = g_k$ для $p(g_k \mid \mathbf{x}) = \max_g p(g \mid \mathbf{x})$.
- Опять k-NN строит приближение к этой формуле – выбирает большинством голосов в окрестности точки.
- Что делает линейный классификатор, мы уже обсуждали – кодируем g через 0-1 переменную y, приближаем y линейной функцией, предсказываем.
- Правда, странновато получается – наше приближение может быть отрицательным или большим 1, например.

Bias-variance decomposition

- На прошлой лекции мы уже изучали статистическую теорию принятия решений.
- Рассмотрим совместное распределение $p(y, \mathbf{x})$ и квадратичную функцию потерь $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$.
- Мы знаем, что тогда оптимальная оценка – это функция регрессии

$$\widehat{f}(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}] = \int y p(y \mid \mathbf{x}) dx.$$

Bias-variance decomposition

- Давайте подсчитаем ожидаемую ошибку и перепишем её в другой форме:

$$\mathbb{E}[L] = \mathbb{E}[(y - f(\mathbf{x}))^2] = \mathbb{E}[(y - \mathbb{E}[y \mid \mathbf{x}] + \mathbb{E}[y \mid \mathbf{x}] - f(\mathbf{x}))^2] =$$

$$= \int (f(\mathbf{x}) - \mathbb{E}[y \mid \mathbf{x}])^2 \, p(\mathbf{x}) d\mathbf{x} + \int (\mathbb{E}[y \mid \mathbf{x}] - y)^2 \, p(\mathbf{x}, y) d\mathbf{x} dy,$$

потому что
$\int (f(\mathbf{x}) - \mathbb{E}[y \mid \mathbf{x}]) (\mathbb{E}[y \mid \mathbf{x}] - y) \, p(\mathbf{x}, y) d\mathbf{x} dy = 0.$

Bias-variance decomposition

- Эта форма записи – разложение на bias-variance и noise:

$$\mathbb{E}[L] = \int (f(\mathbf{x}) - \mathbb{E}[y \mid \mathbf{x}])^2 p(\mathbf{x}) d\mathbf{x} + \int (\mathbb{E}[y \mid \mathbf{x}] - y)^2 p(\mathbf{x}, y) d\mathbf{x} dy,$$

- Отсюда, кстати, тоже сразу видно, что от $f(\mathbf{x})$ зависит только первый член, и он минимизируется, когда

$$f(\mathbf{x}) = \hat{f}(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}].$$

- A noise, $\int (\mathbb{E}[y \mid \mathbf{x}] - y)^2 p(\mathbf{x}, y) d\mathbf{x} dy$, – это просто свойство данных, дисперсия шума.

Bias-variance decomposition

- Если бы у нас был всемогущий компьютер и неограниченный датасет, мы бы, конечно, на этом и закончили, посчитали бы $\hat{f}(\mathbf{x}) = \mathbb{E}[y \mid \mathbf{x}]$, и всё.
- Однако жизнь – борьба, и у нас есть только ограниченный датасет из N точек. Предположим, что этот датасет берётся по распределению $p(\mathbf{x}, y)$ – т.е. фактически рассмотрим много-много экспериментов такого вида:
  - взяли датасет D из N точек по распределению $p(\mathbf{x}, y)$;
  - подсчитали нашу чудо-регрессию;
  - получили новую функцию предсказания $f(\mathbf{x}; D)$.
- Разные датасеты будут приводить к разным функциям предсказания...

Bias-variance decomposition

- ...а потому давайте усредним теперь по датасетам.
- Наш первый член в ожидаемой ошибке выглядел как $\left(f(\mathbf{x}) - \hat{f}(\mathbf{x})\right)^2$, а теперь будет $\left(f(\mathbf{x}; D) - \hat{f}(\mathbf{x})\right)^2$, и его можно усреднить по D, применив такой же трюк:

$$\left(f(\mathbf{x}; D) - \hat{f}(\mathbf{x})\right)^2$$

$$= \left(f(\mathbf{x}; D) - \mathbb{E}_D\left[f(\mathbf{x}; D)\right] + \mathbb{E}_D\left[f(\mathbf{x}; D)\right] - \hat{f}(\mathbf{x})\right)^2$$

$$= \left(f(\mathbf{x}; D) - \mathbb{E}_D\left[f(\mathbf{x}; D)\right]\right)^2 + \left(\mathbb{E}_D\left[f(\mathbf{x}; D)\right] - \hat{f}(\mathbf{x})\right)^2 + 2(\ldots)(\ldots),$$

и в ожидании получится...

Bias-variance decomposition

- ...и в ожидании получится

$$\mathbb{E}_D\left[\left(f(\mathbf{x};D)-\hat{f}(\mathbf{x})\right)^2\right]=$$

$$=\mathbb{E}_D\left[\left(f(\mathbf{x};D)-\mathbb{E}_D\left[f(\mathbf{x};D)\right]\right)^2\right]+\left(\mathbb{E}_D\left[f(\mathbf{x};D)\right]-\hat{f}(\mathbf{x})\right)^2.$$

- Разложили на дисперсию $\mathbb{E}_D\left[\left(f(\mathbf{x};D)-\mathbb{E}_D\left[f(\mathbf{x};D)\right]\right)^2\right]$ и квадрат систематической ошибки $\left(\mathbb{E}_D\left[f(\mathbf{x};D)\right]-\hat{f}(\mathbf{x})\right)^2$; это и есть bias-variance decomposition.

Bias-variance-noise

$$\text{Expected loss} = (\text{bias})^2 + \text{variance} + \text{noise},$$

где

$$(\text{bias})^2 = \left(\mathbb{E}_D\left[f(\mathbf{x}; D)\right] - \hat{f}(\mathbf{x})\right)^2,$$

$$\text{variance} = \mathbb{E}_D\left[\left(f(\mathbf{x}; D) - \mathbb{E}_D\left[f(\mathbf{x}; D)\right]\right)^2\right],$$

$$\text{noise} = \int \left(\mathbb{E}\left[y \mid \mathbf{x}\right] - y\right)^2 p(\mathbf{x}, y) d\mathbf{x} dy.$$

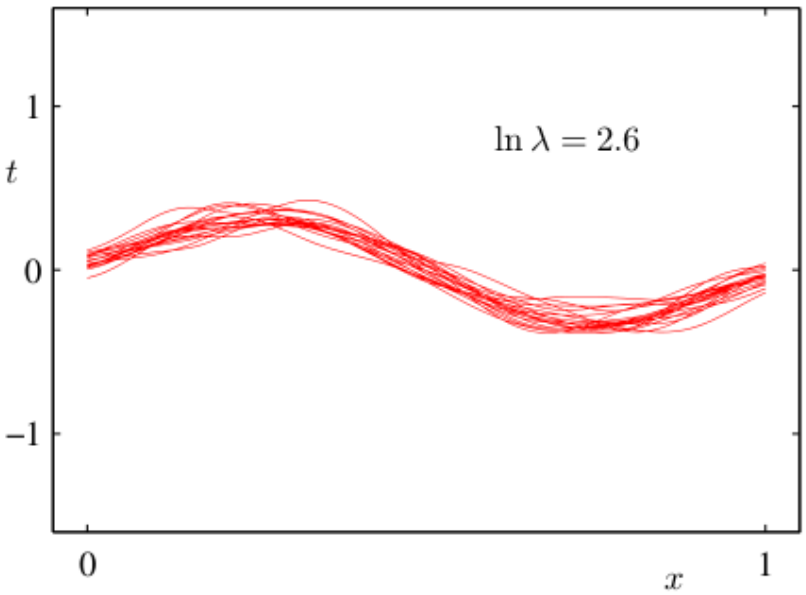
Пример

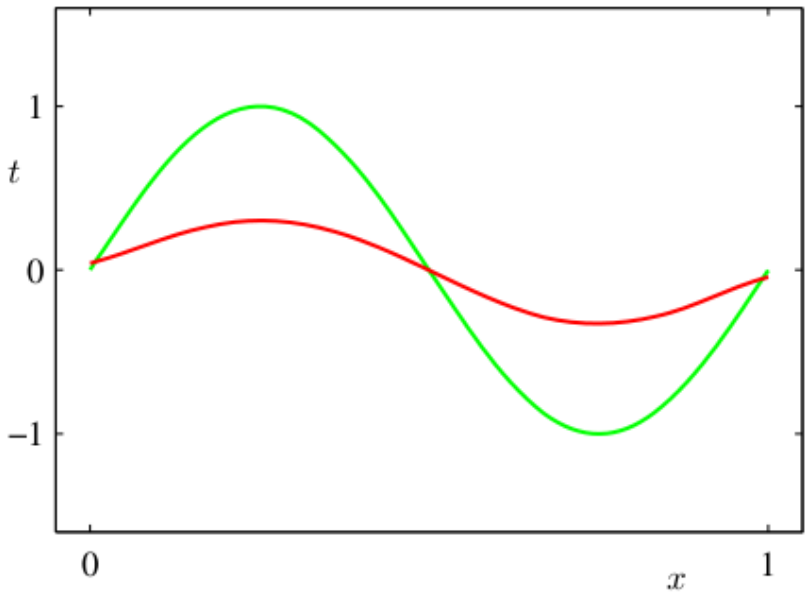
- Теперь давайте посмотрим на пример: опять та же синусоида, но теперь мы приближаем её линейной регрессией с гауссовскими базисными функциями:

$$\phi_j(x) = e^{-\frac{1}{2s^2}\left(x-\mu_j\right)^2}.$$

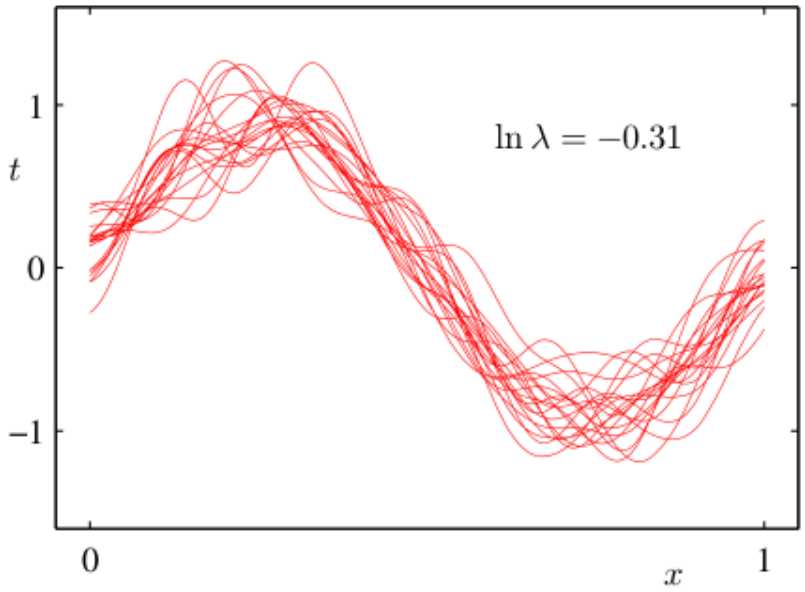
- И мы регуляризуем эту регрессию с параметром λ.
- Будем набрасывать много датасетов и смотреть, что меняется при этом.

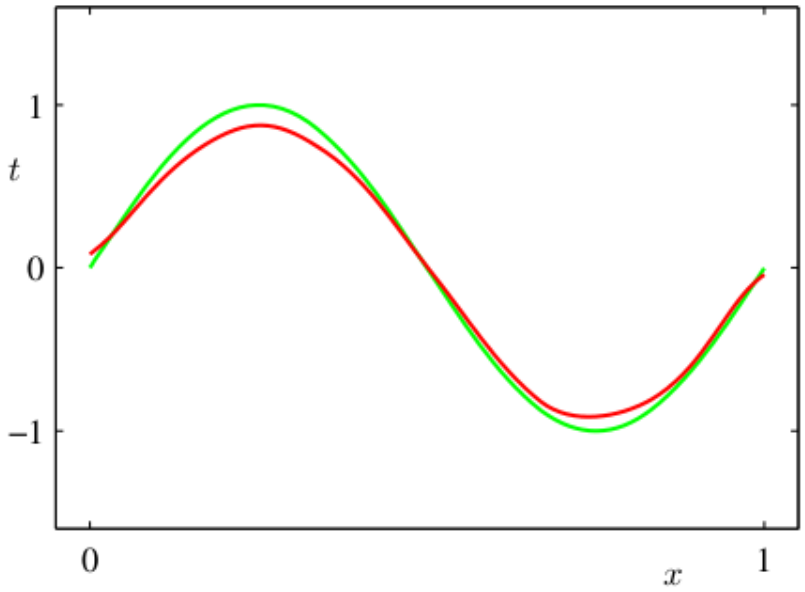
Регуляризатор и bias-variance



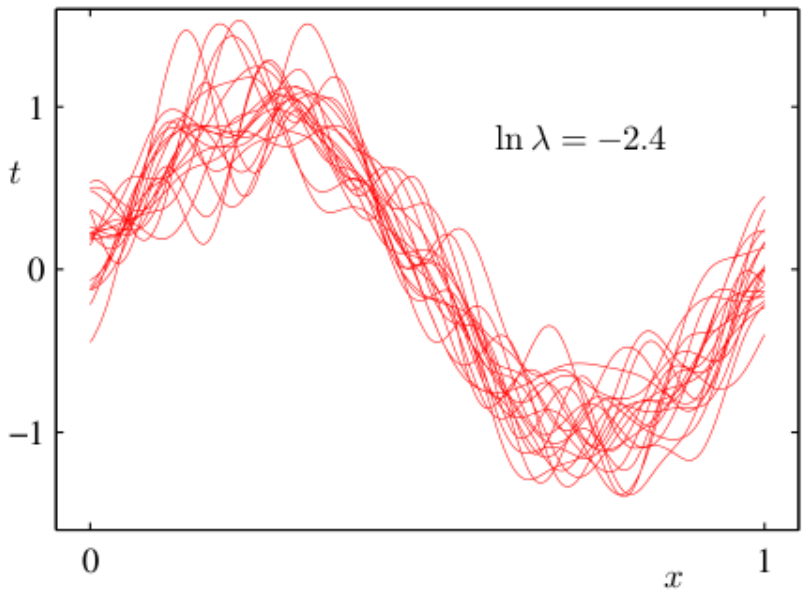


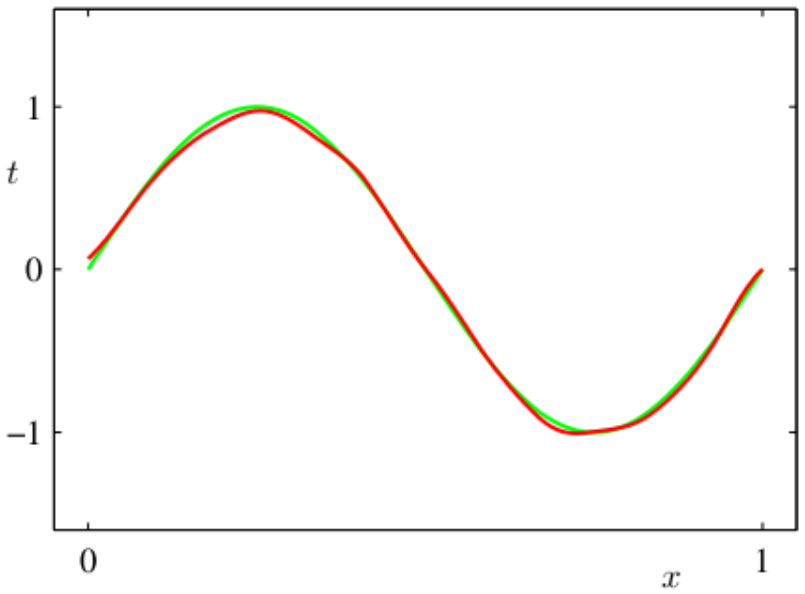
Регуляризатор и bias-variance



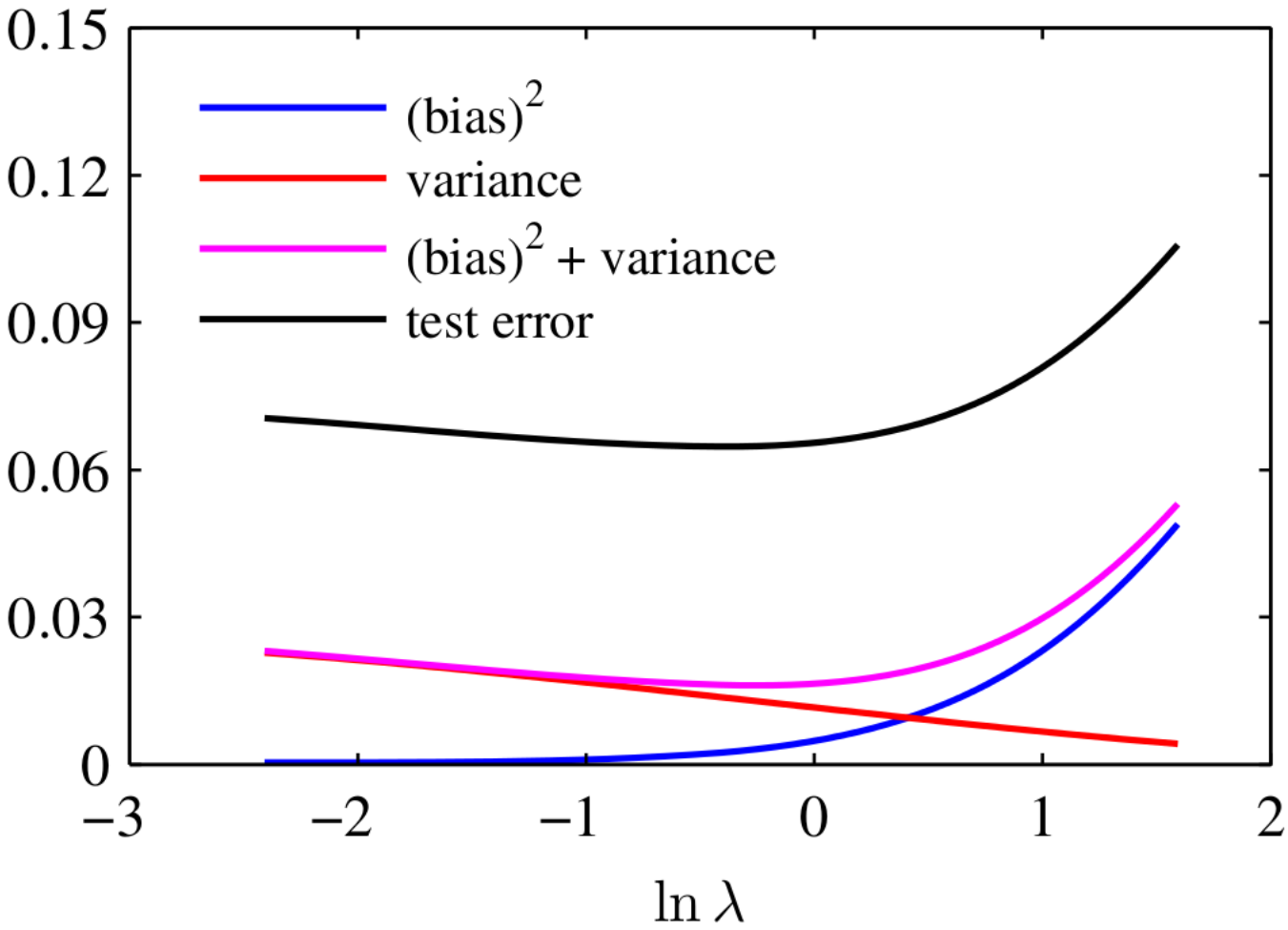


Регуляризатор и bias-variance





Регуляризатор и bias-variance



Outline

В предыдущих сериях...

- Теорема Байеса:

$$p(\theta|D) = \frac{p(\theta)p(D|\theta)}{p(D)}.$$

- Две основные задачи байесовского вывода:
  1. найти апостериорное распределение на гипотезах/параметрах:

$$p(\theta \mid D) \propto p(D|\theta)p(\theta)$$

     (и/или найти максимальную апостериорную гипотезу $\arg\max_\theta p(\theta \mid D)$);
  2. найти апостериорное распределение исходов дальнейших экспериментов:

$$p(x \mid D) \propto \int_{\theta\in\Theta} p(x \mid \theta)p(D|\theta)p(\theta)\mathrm{d}\theta.$$

В предыдущих сериях...

- Мы изучили метод наименьших квадратов для линейной регрессии и метод ближайших соседей...
- ...построили функцию регрессии

$$\hat{f}(\mathbf{x}) = \mathrm{E}_{y|\mathbf{x}'}(y \mid \mathbf{x}' = \mathbf{x})$$

и оптимальный классификатор

$$\hat{g}(\mathbf{x}) = \arg\min_g \sum_{k=1}^{K} L(g_k, \hat{g}(\mathbf{x}))p(g_k \mid \mathbf{x})...$$

- ...и выяснили, что метод наименьших квадратов – это метод максимального правдоподобия для нормально распределённого шума.

Проклятие размерности

- В прошлый раз k-NN давали гораздо более разумные результаты, чем линейная модель, особенно если хорошо выбрать k.
- Может быть, нам в этой жизни больше ничего и не нужно?
- Давайте посмотрим, как k-NN будет вести себя в более высокой размерности (что очень реалистично).

Проклятие размерности

- Давайте поищем ближайших соседей у точки в единичном гиперкубе. Предположим, что наше исходное распределение равномерное.
- Чтобы покрыть долю α тестовых примеров, нужно (ожидаемо) покрыть долю α объёма, и ожидаемая длина ребра гиперкуба-окрестности в размерности p будет $e_p(\alpha) = \alpha^{1/p}$.
- Например, в размерности 10 $e_{10}(0.1) = 0.8$, $e_{10}(0.01) = 0.63$, т.е. чтобы покрыть 1% объёма, нужно взять окрестность длиной больше половины носителя по каждой координате!
- Это скажется и на k-NN: трудно отвергнуть по малому числу координат, быстрые алгоритмы хуже работают.

Проклятие размерности

- Второе проявление the curse of dimensionality: пусть N точек равномерно распределены в единичном шаре размерности p. Тогда среднее расстояние от нуля до точки равно

$$d(p, N) = \left(1 - \frac{1}{2}^{1/N}\right)^{1/p},$$

  т.е., например, в размерности 10 для $N = 500$ $d \approx 0.52$, т.е. больше половины.
- Большинство точек в результаты ближе к границе носителя, чем к другим точкам, а это для ближайших соседей проблема – придётся не интерполировать внутри существующих точек, а экстраполировать наружу.

Проклятие размерности

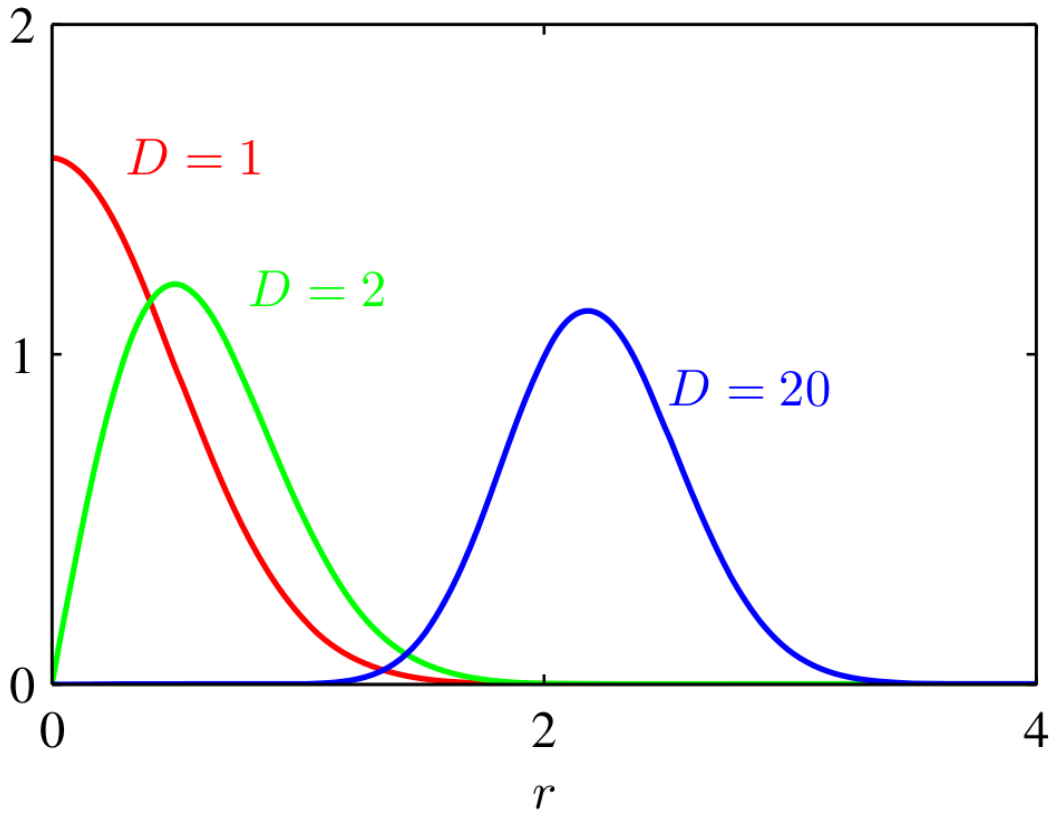
- Третье проявление: проблемы в оптимизации, которые и имел в виду Беллман.
- Если нужно примерно оптимизировать функцию от d переменных, на решётке с шагом ϵ понадобится примерно $\left(\frac{1}{\epsilon}\right)^d$ вычислений функции.
- В численном интегрировании – чтобы интегрировать функцию с точностью ϵ, нужно тоже примерно $\left(\frac{1}{\epsilon}\right)^d$ вычислений.

Проклятие размерности

- Плотные множества становятся очень разреженными. Например, чтобы получить плотность, создаваемую в размерности 1 при помощи $N = 100$ точек, в размерности 10 нужно будет 100^{10} точек.
- Поведение функций тоже усложняется с ростом размерности – чтобы строить регрессии в высокой размерности с той же точностью, может потребоваться экспоненциально больше точек, чем в низкой размерности.
- А у линейной модели ничего такого не наблюдается, она не подвержена проклятию размерности.

Проклятие размерности

- Ещё пример: нормально распределённая величина будет сосредоточена в тонкой оболочке.



Упражнение. Переведите плотность нормального распределения в полярные координаты и проверьте это утверждение.

Thank you!

Спасибо за внимание!