Статистическая теория принятия решений

Сергей Николенко

Samsung AI Center — Москва

13 апреля 2020 г.

Random facts:

- 13 апреля — Всемирный день рок-н-ролла; за день до этого в 1954 г. Билл Хейли записал *Rock Around the Clock*; 13 апреля 1962 г. Beatles впервые выступили в «Star Club», а 13 апреля 1967 г. The Rolling Stones впервые проникли за железный занавес (в Варшаву)
- 13 апреля в Таиланде — Сонгкран, праздник нового года, который там отмечается по древнеиндийскому календарю, основанному на сидерическом зодиакальном цикле; Сонкгран — семейный праздник, когда принято угощать буддийских священнослужителей, омывать домашние статуи Будды и поливать водой друг друга
- 13 апреля 1919 г. было создано Временное правительство Республики Корея, правительство в изгнании на время японской оккупации; один из его лидеров, президент Ли Сын Ман, позже (в 1948) победил на первых всеобщих демократических выборах и стал первым президентом Республики Корея
- 13 апреля 1953 г. Аллен Даллес запустил проект MK-Ultra, программу далеко не всегда законных экспериментов над людьми, призванную разработать медикаменты и процедуры для управления сознанием: ослабляющие волю при допросах, стирающие память и т.д.; в 1953 г. бюджет программы составлял 6% общего бюджета ЦРУ

Статистическая теория принятия решений

Функция потери

- Сейчас мы попытаемся понять, что же на самом деле происходит в этих методах.
- Начнём с обычной регрессии – непрерывный вещественный вход $\mathbf{x} \in \mathbb{R}^p$, непрерывный вещественный выход $y \in \mathbb{R}$; 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):

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

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

Функция потери

- Это можно переписать как

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

  и, значит, можно теперь минимизировать EPE поточечно:

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

  а это можно решить и получить

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

- Это решение называется *функцией регрессии* и является наилучшим предсказанием y в любой точке x.

k-NN

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

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

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

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

- 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

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

- Перепишем как раньше:

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

- Опять достаточно оптимизировать поточечно:

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

Классификация

- Опять достаточно оптимизировать поточечно:

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

- Для 0-1 функции потери это упрощается до

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

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

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

Bias-variance decomposition

- 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

$$\hat{f}(\mathbf{x}) = \mathbf{E}\left[y \mid \mathbf{x}\right] = \int y p(y \mid \mathbf{x}) dx.$$

Bias-variance decomposition

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

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

потому что

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

Bias-variance decomposition

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

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

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

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

- A noise, $\int \left(\mathsf{E}\left[y \mid \mathbf{x}\right] - y\right)^2 p(\mathbf{x}, y) d\mathbf{x} dy$, – 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 $\hat{f}(\mathbf{x}) = \mathbf{E}\left[y \mid \mathbf{x}\right]$, 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:

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

и в ожидании получится...

Bias-variance decomposition

- ...и в ожидании получится

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

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

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

Bias-variance-noise

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

где

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

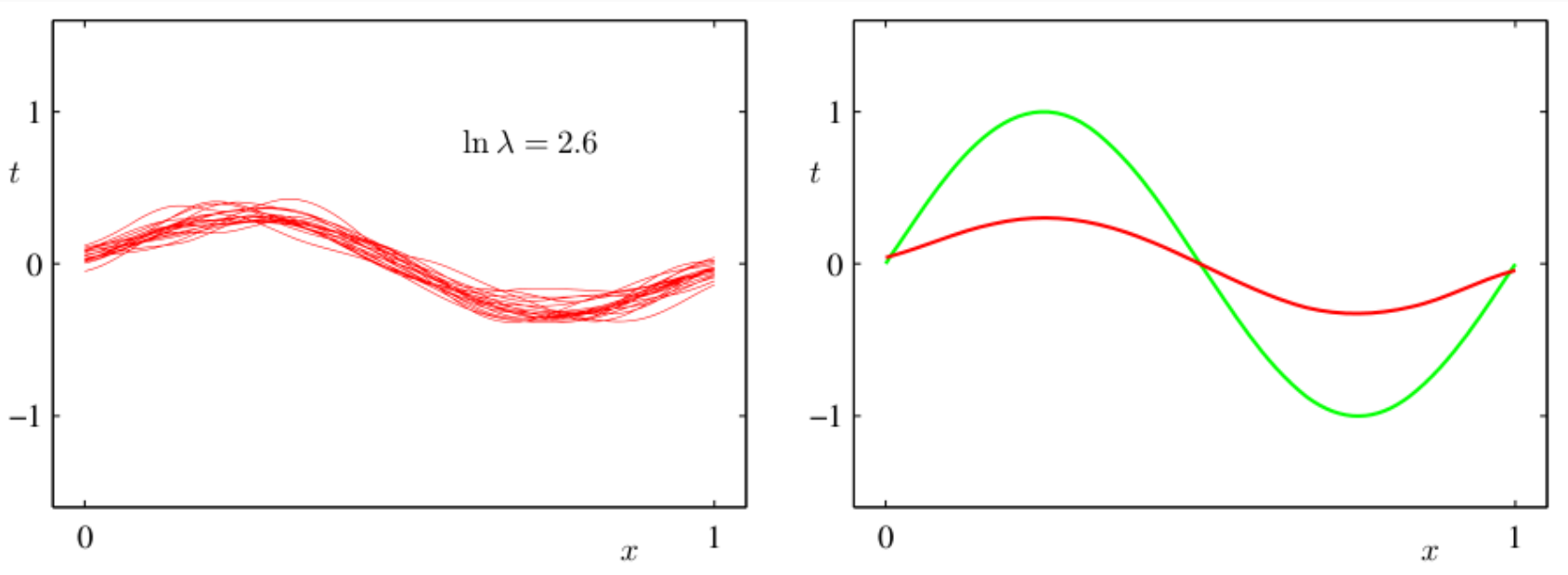
$$\text{variance} = \mathbf{E}_D\left[\left(f(\mathbf{x}; D) - \mathbf{E}_D\left[f(\mathbf{x}; D)\right]\right)^2\right],$$

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

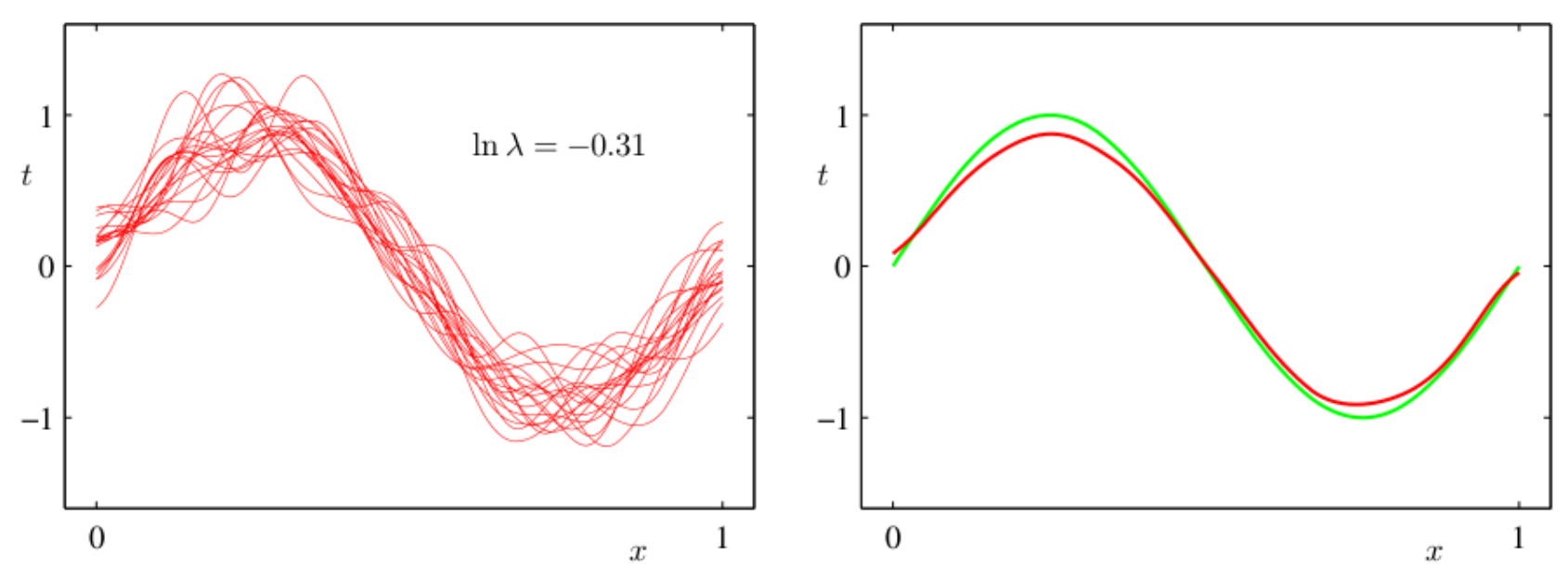
Пример

- Теперь давайте посмотрим на пример: опять та же синусоида, опять приближаем её линейной регрессией с полиномиальными признаками (максимальным их числом).
- И мы регуляризуем эту регрессию с параметром α.
- Будем набрасывать много датасетов и смотреть, что меняется при этом.

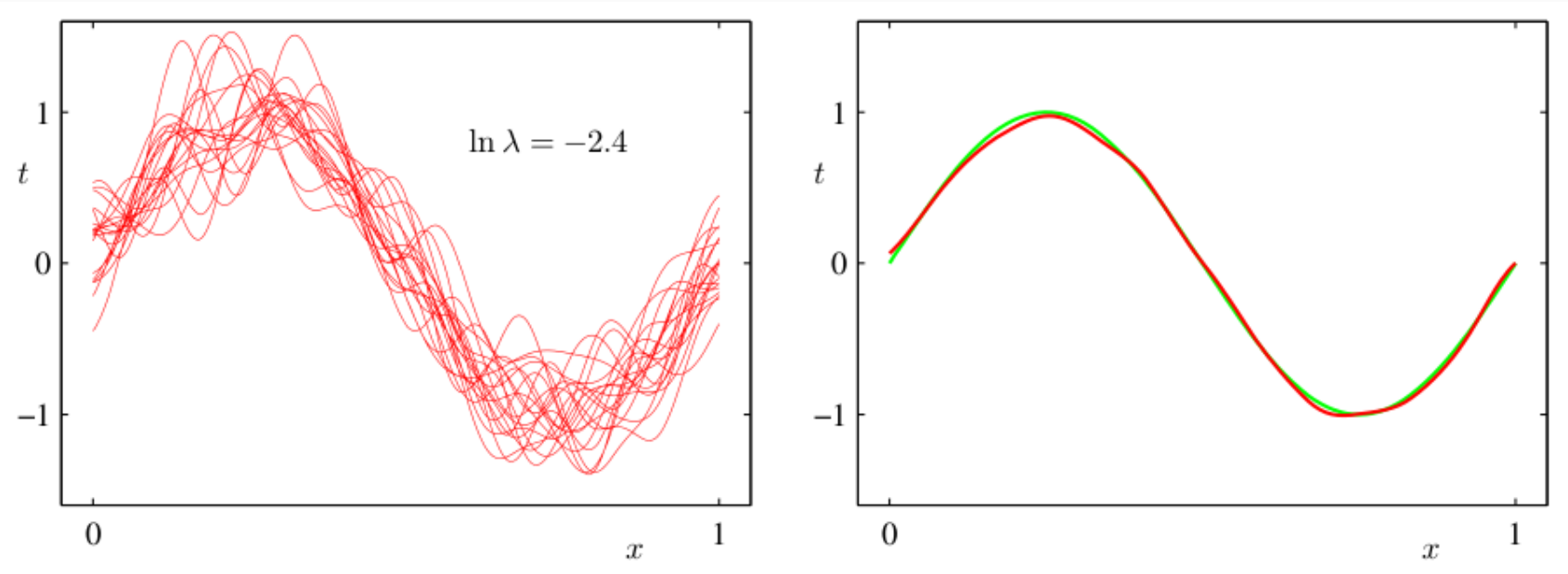
Регуляризатор и bias-variance



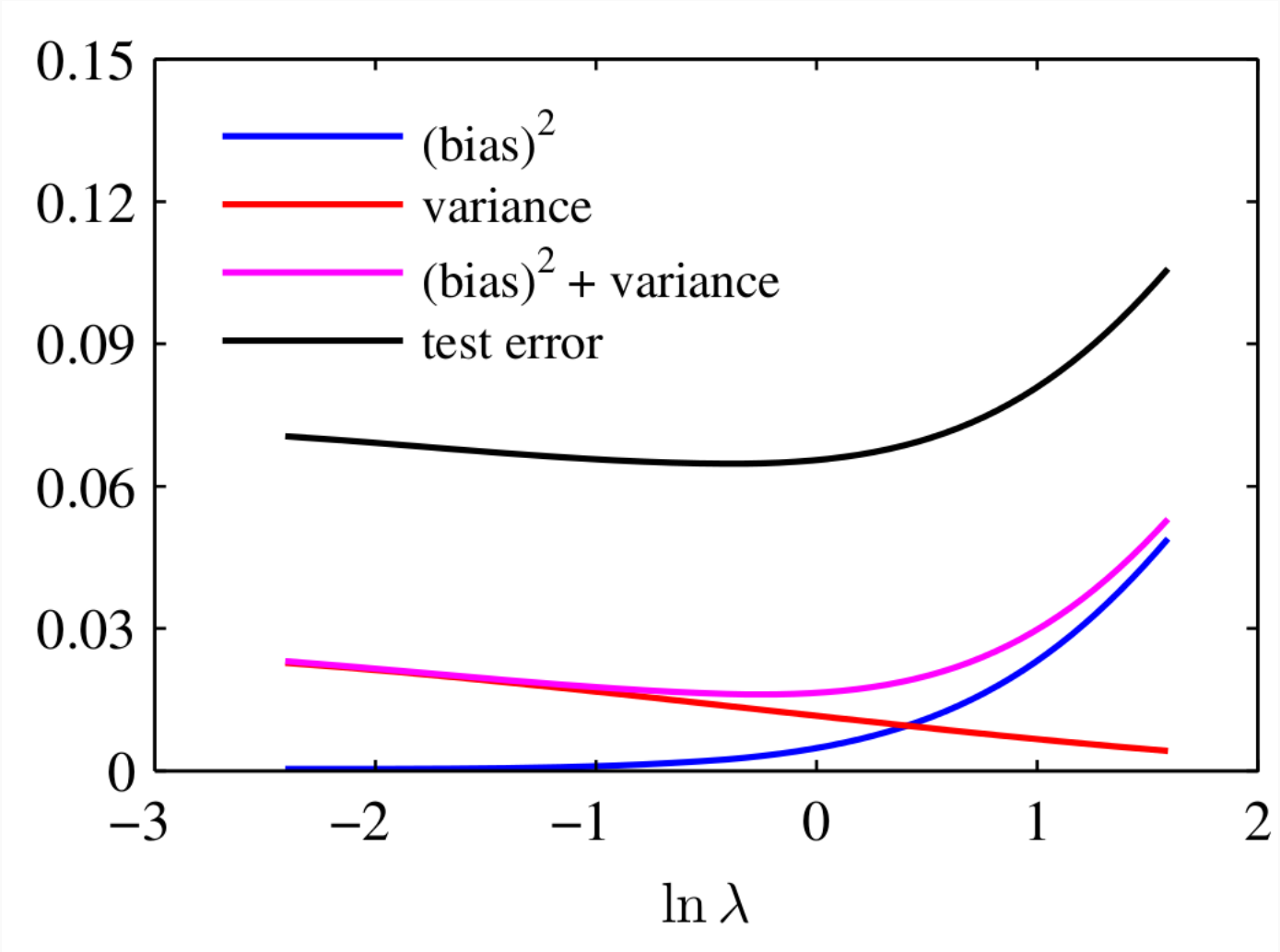
Регуляризатор и bias-variance



Регуляризатор и bias-variance



Регуляризатор и bias-variance



Эквивалентное ядро и сравнение моделей

Эквивалентное ядро

- Вспомним наши байесовские предсказания:

$$p(t \mid \mathbf{t}, \alpha, \beta) = \mathcal{N}(t \mid \boldsymbol{\mu}_N^\top \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2),$$
$$\text{где } \sigma_N^2 = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_N \boldsymbol{\phi}(\mathbf{x}).$$

- Давайте перепишем среднее апостериорного распределения в другой форме (вспомним, что $\boldsymbol{\mu}_N = \beta \boldsymbol{\Sigma}_N \boldsymbol{\Phi}^\top \mathbf{t}$):

$$y(\mathbf{x}, \boldsymbol{\mu}_N) = \boldsymbol{\mu}_N^\top \boldsymbol{\phi}(\mathbf{x}) = \beta \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_N \boldsymbol{\Phi}^\top \mathbf{t} =$$
$$= \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_N \boldsymbol{\phi}(\mathbf{x}_n) t_n.$$

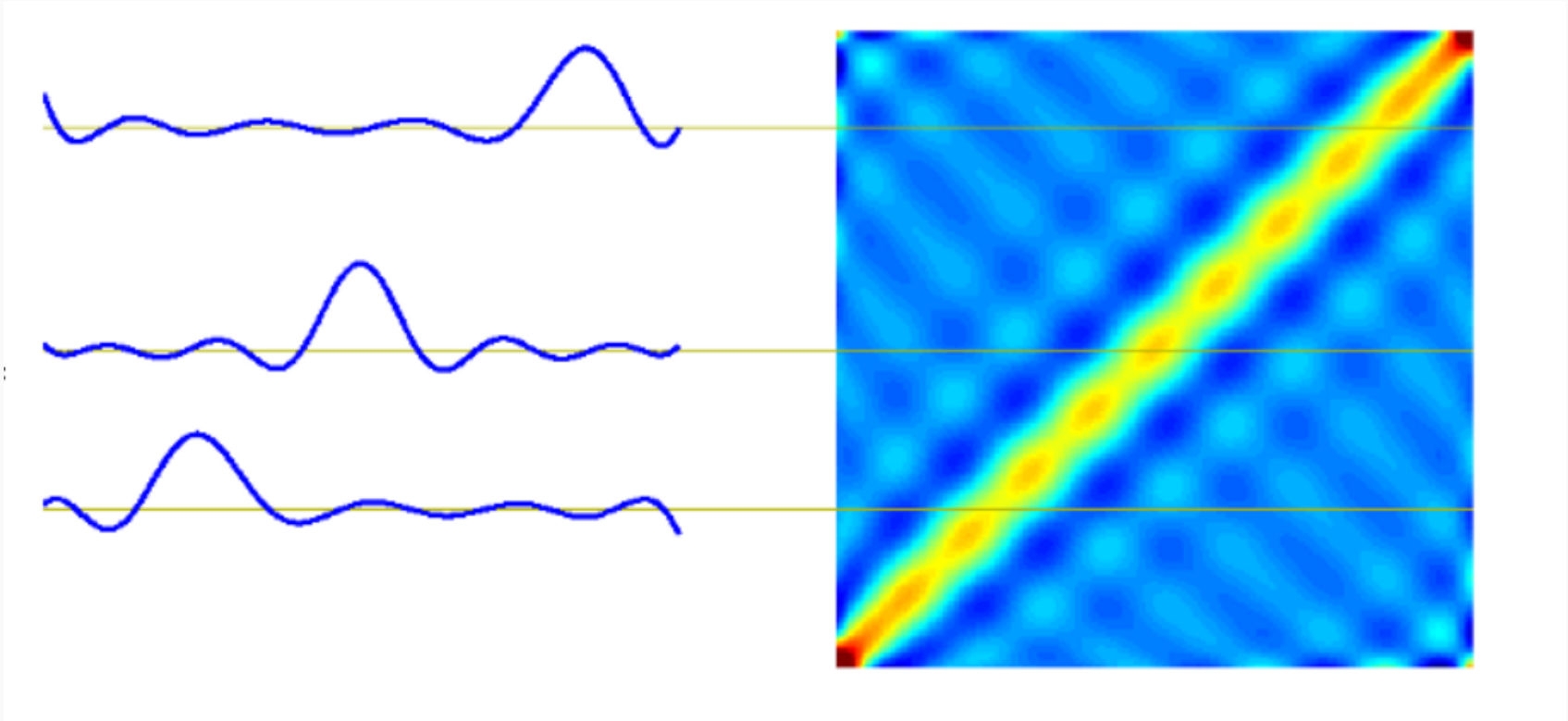
Эквивалентное ядро

- $y(\mathbf{x}, \boldsymbol{\mu}_N) = \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_N \boldsymbol{\phi}(\mathbf{x}_n) t_n$.
- Это значит, что предсказание можно переписать как

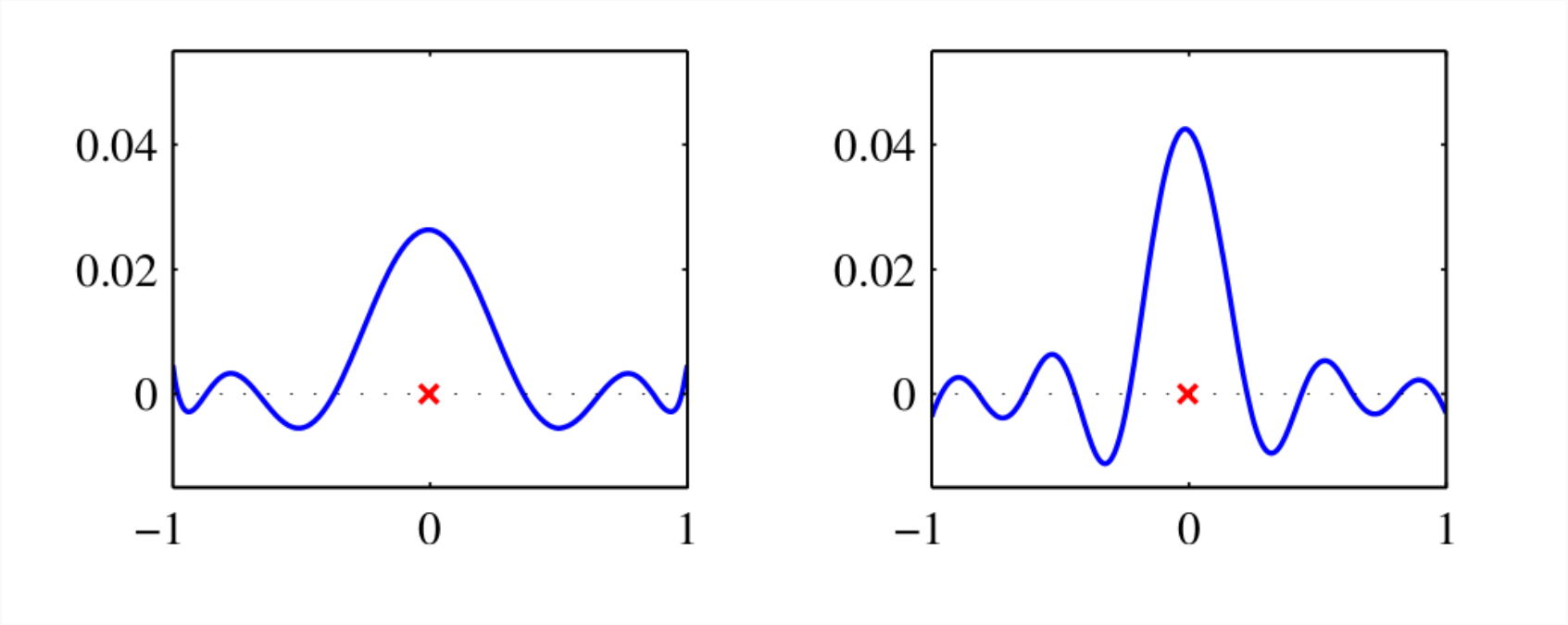
$$y(\mathbf{x}, \boldsymbol{\mu}_N) = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) t_n.$$

- Т.е. мы предсказываем следующую точку как линейную комбинацию значений в известных точках.
- Функция $k(\mathbf{x}, \mathbf{x}') = \beta \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_N \boldsymbol{\phi}(\mathbf{x}')$ называется *эквивалентным ядром* (equivalent kernel).

Эквивалентное ядро



Эквивалентное ядро



Выводы про эквивалентное ядро

- Эквивалентное ядро $k(\mathbf{x}, \mathbf{x}')$ локализовано вокруг $\mathbf{x}$ как функция $\mathbf{x}'$, т.е. каждая точка оказывает наибольшее влияние около себя и затухает потом.
- Можно было бы с самого начала просто определить ядро и предсказывать через него, безо всяких базисных функций ϕ – такой подход мы ещё будем рассматривать.

Упражнение. Докажите, что $\sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) = 1$.

Байесовское сравнение моделей

- Мы говорили о том, что при увеличении числа параметров модели возникает оверфиттинг.
- Как этого избежать? Как сравнить модели с разным числом параметров?
- Теория байесовского вывода предлагает такой выход: давайте будем не точечные оценки параметров модели рассматривать, а тоже интегрировать по параметрам модели.

Байесовское сравнение моделей

- Пусть мы хотим сравнить модели из множества $\{\mathcal{M}_i\}_{i=1}^{L}$.
- Модель – это распределение вероятностей над данными D.
- По тестовому набору D можно оценить апостериорное распределение

$$p(\mathcal{M}_i \mid D) \propto p(\mathcal{M}_i)p(D \mid \mathcal{M}_i).$$

Байесовское сравнение моделей

- Если знать апостериорное распределение, то можно сделать предсказание:

$$p(t \mid \mathbf{x}, D) = \sum_{i=1}^{L} p(t \mid \mathbf{x}, \mathcal{M}_i, \mathcal{D}) p(\mathcal{M}_i \mid D).$$

- *Model selection* (выбор модели) – это когда мы приближаем предсказание, выбирая просто самую (апостериорно) вероятную модель.

Байесовское сравнение моделей

- Если модель определена параметрически, через $\mathbf{w}$, то

$$p(D \mid \mathcal{M}_i) = \int p(D \mid \mathbf{w}, \mathcal{M}_i) p(\mathbf{w} \mid \mathcal{M}_i) d\mathbf{w}.$$

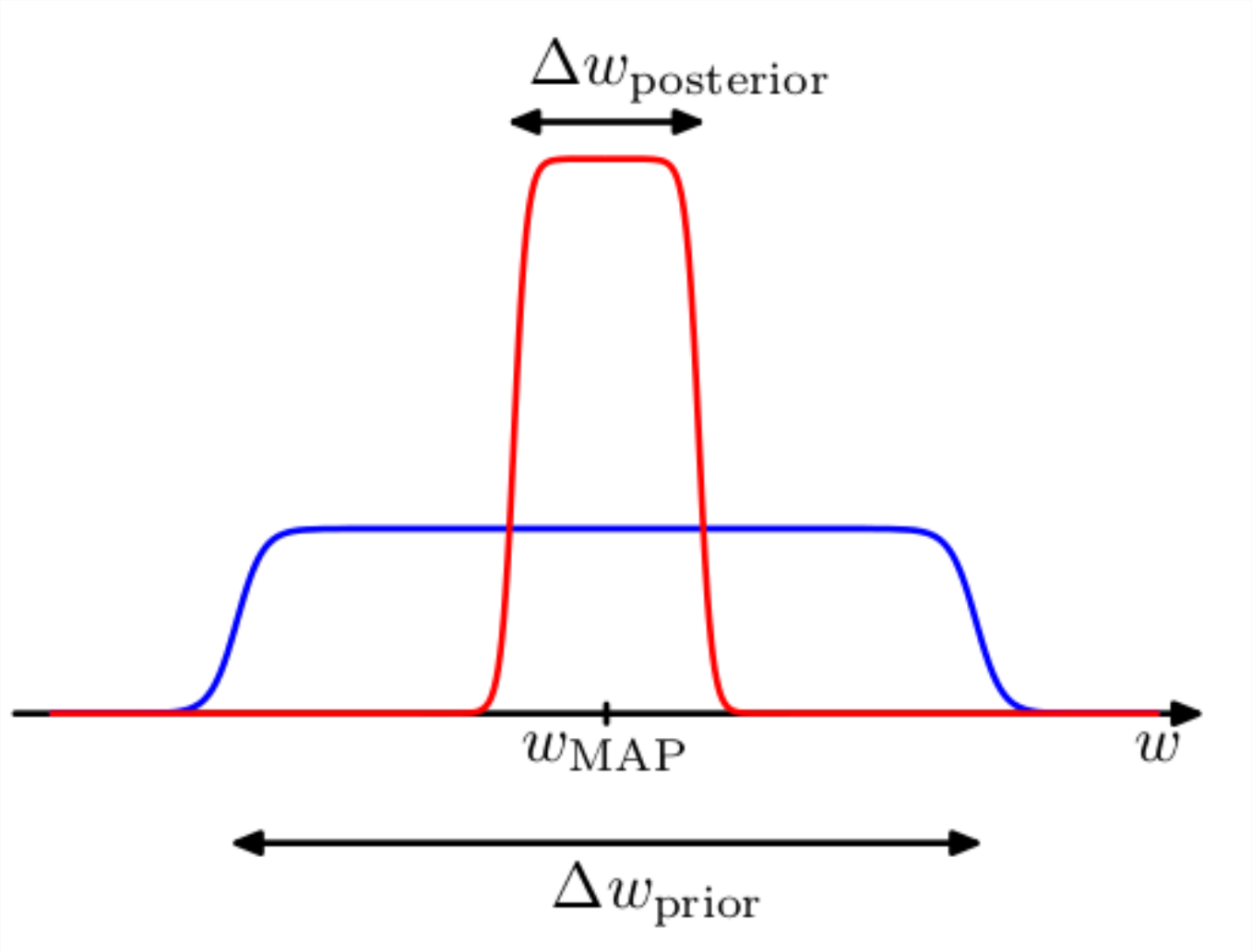
- Т.е. это вероятность сгенерировать D, если выбирать параметры модели по её априорному распределению, а потом накидывать данные.
- Это, кстати, в точности знаменатель из теоремы Байеса:

$$p(\mathbf{w} \mid \mathcal{M}_i, D) = \frac{p(D \mid \mathbf{w}, \mathcal{M}_i) p(\mathbf{w} \mid \mathcal{M}_i)}{p(D \mid \mathcal{M}_i)}.$$

Байесовское сравнение моделей

- Предположим, что у модели один параметр w, а апостериорное распределение – это острый пик вокруг w_{MAP} шириной $\Delta w_{\mathrm{posterior}}$.
- Тогда можно приблизить $p(D) = \int p(D \mid w)p(w)dw$ как значение в максимуме, умноженное на ширину.
- Предположим ещё, что априорное распределение тоже плоское, $p(w) = \frac{1}{\Delta w_{\mathrm{prior}}}$.

Приближение $p(D)$



Приближение $p(D)$

- Тогда получится

$$p(D) = \int p(D \mid w)p(w)dw \approx p\left(D \mid w_{\mathrm{MAP}}\right) \frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}},$$

$$\ln p(D) \approx \ln p\left(D \mid w_{\mathrm{MAP}}\right) + \ln\left(\frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}}\right).$$

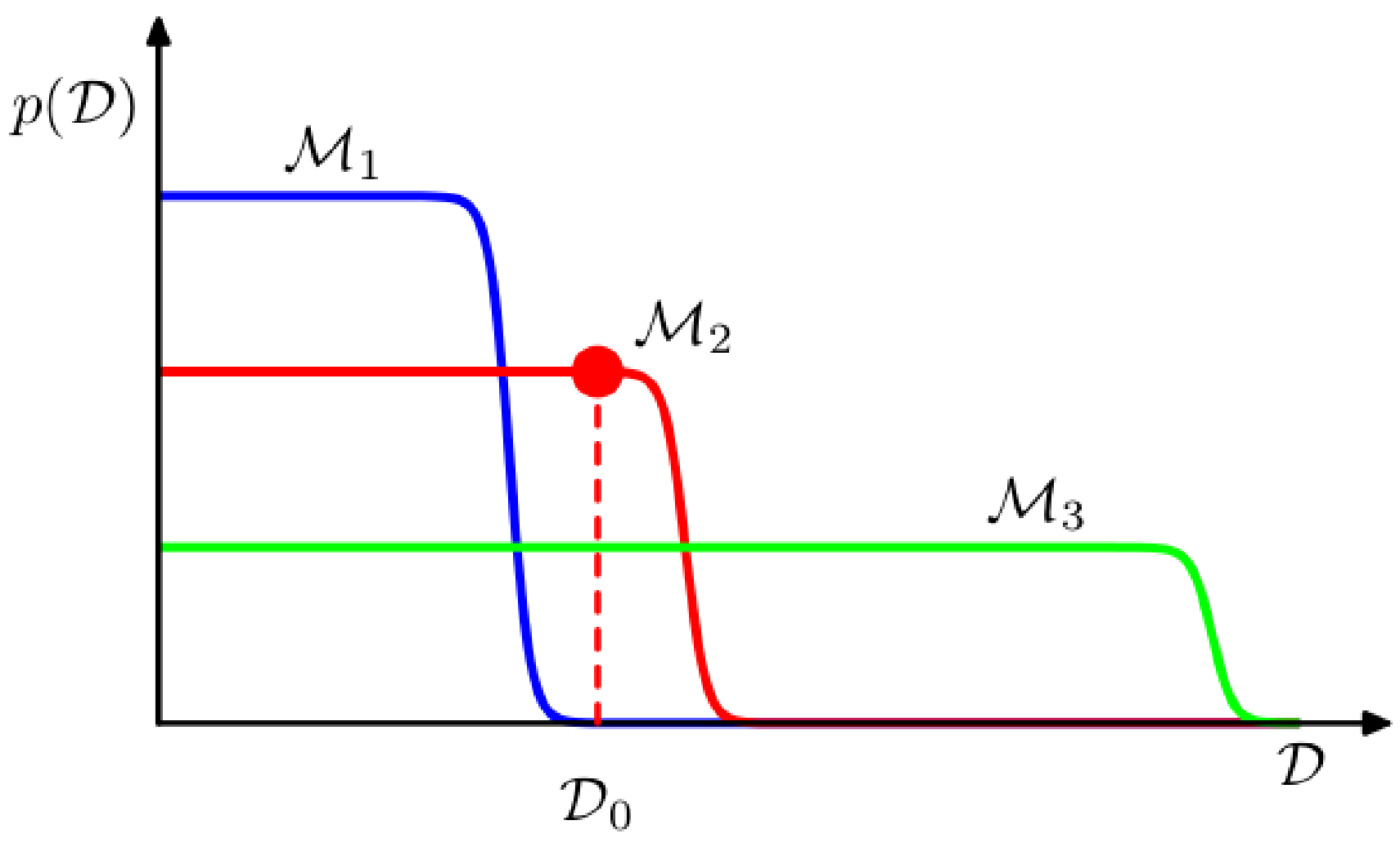
- Это значит, что мы добавляем штраф за «слишком узкое» апостериорное распределение – то есть в точности штраф за оверфиттинг!
- Для модели из M параметров, если предположить, что у них одинаковые $\Delta w_{\mathrm{posterior}}$, получим

$$\ln p(D) \approx \ln p\left(D \mid w_{\mathrm{MAP}}\right) + M \ln\left(\frac{\Delta w_{\mathrm{posterior}}}{\Delta w_{\mathrm{prior}}}\right).$$

Другой взгляд

- Другими словами: давайте посмотрим, какие датасеты может генерировать та или иная модель.
- Простая модель (e.g., линейная) генерирует похожие датасеты, «мало» разных датасетов, у неё высокая $p(D \mid \mathcal{M})$.
- Сложная модель (e.g., многочлен девятой степени) генерирует «много» разных датасетов, у неё низкая $p(D \mid \mathcal{M})$.
- Но сложная может хорошо выразить датасеты, которые не может выразить простая; поэтому в сумме надо выбирать «среднюю».

Приближение $p(D)$



Правильный ответ лучше

- Sanity check: тут какие-то штрафы мы навводили; будет ли истинный правильный ответ $p(D \mid \mathcal{M}_{\text{true}})$ всегда оптимальным в этом смысле?
- Конечно, для конкретного датасета может так повезти, что не будет.
- Но если усреднить по всем датасетам, выбранным по $p(D \mid \mathcal{M}_{\text{true}})$...

Правильный ответ лучше

- ...то получится

$$\mathsf{E}\left[\ln \frac{p(D \mid \mathcal{M}_{\text{true}})}{p(D \mid \mathcal{M})}\right] = \int p(D \mid \mathcal{M}_{\text{true}}) \ln \frac{p(D \mid \mathcal{M}_{\text{true}})}{p(D \mid \mathcal{M})} dD.$$

- Это называется *расстоянием Кульбака-Лейблера* (Kullback-Leibler divergence) между распределениями $p(D \mid \mathcal{M}_{\text{true}})$ и $p(D \mid \mathcal{M})$.

Введение в классификацию

Задача классификации

- Теперь классификация: определить вектор $\mathbf{x}$ в один из K классов $\mathcal{C}_k$.
- В итоге у нас так или иначе всё пространство разобьётся на эти классы.
- Т.е. на самом деле мы ищем *разделяющую поверхность* (decision surface, decision boundary).

Задача классификации

- Как кодировать? Бинарная задача – очень естественно, переменная t, $t = 0$ соответствует $\mathcal{C}_1$, $t = 1$ соответствует $\mathcal{C}_2$.
- Оценку t можно интерпретировать как вероятность (по крайней мере, мы постараемся, чтобы было можно).
- Если несколько классов – удобно 1-of-K:

$$\mathbf{t} = (0, \ldots, 0, 1, 0, \ldots)^\top.$$

- Тоже можно интерпретировать как вероятности – или пропорционально им.

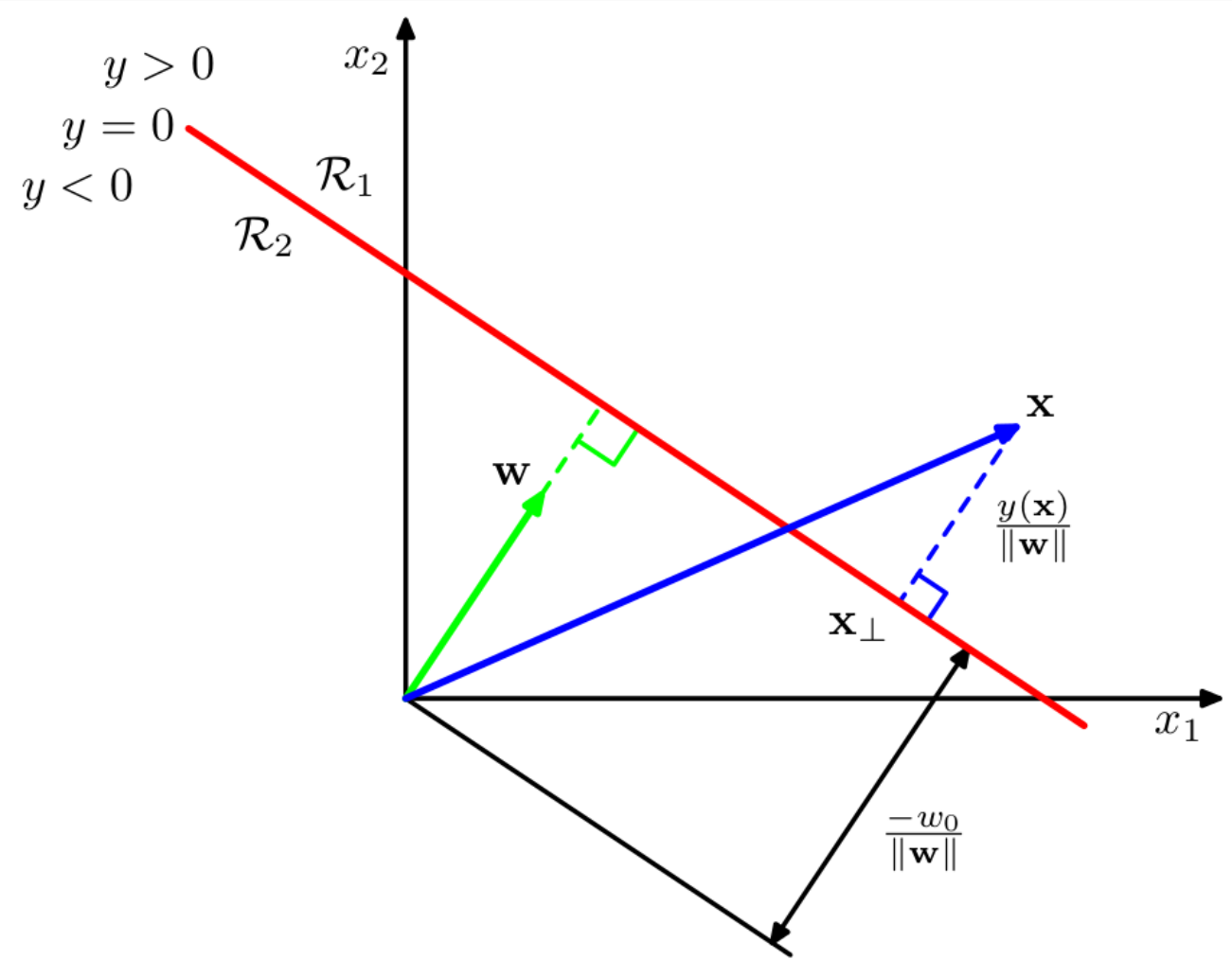
Разделяющая гиперплоскость

- Начнём с геометрии: рассмотрим линейную дискриминантную функцию

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0.$$

- Это гиперплоскость, и $\mathbf{w}$ – нормаль к ней.
- Расстояние от начала координат до гиперплоскости равно $\frac{-w_0}{\|\mathbf{w}\|}$.
- $y(\mathbf{x})$ связано с расстоянием до гиперплоскости: $d = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}$.

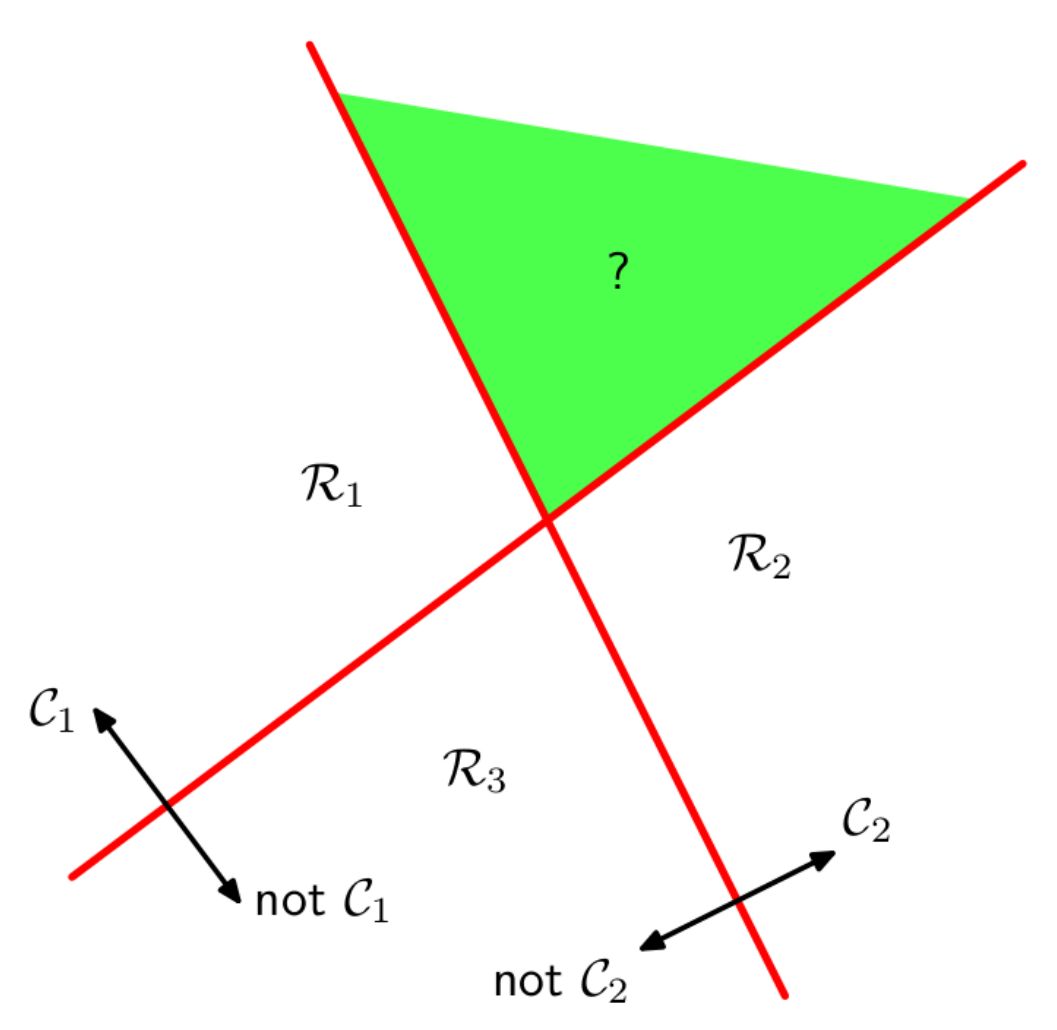
Разделяющая гиперплоскость



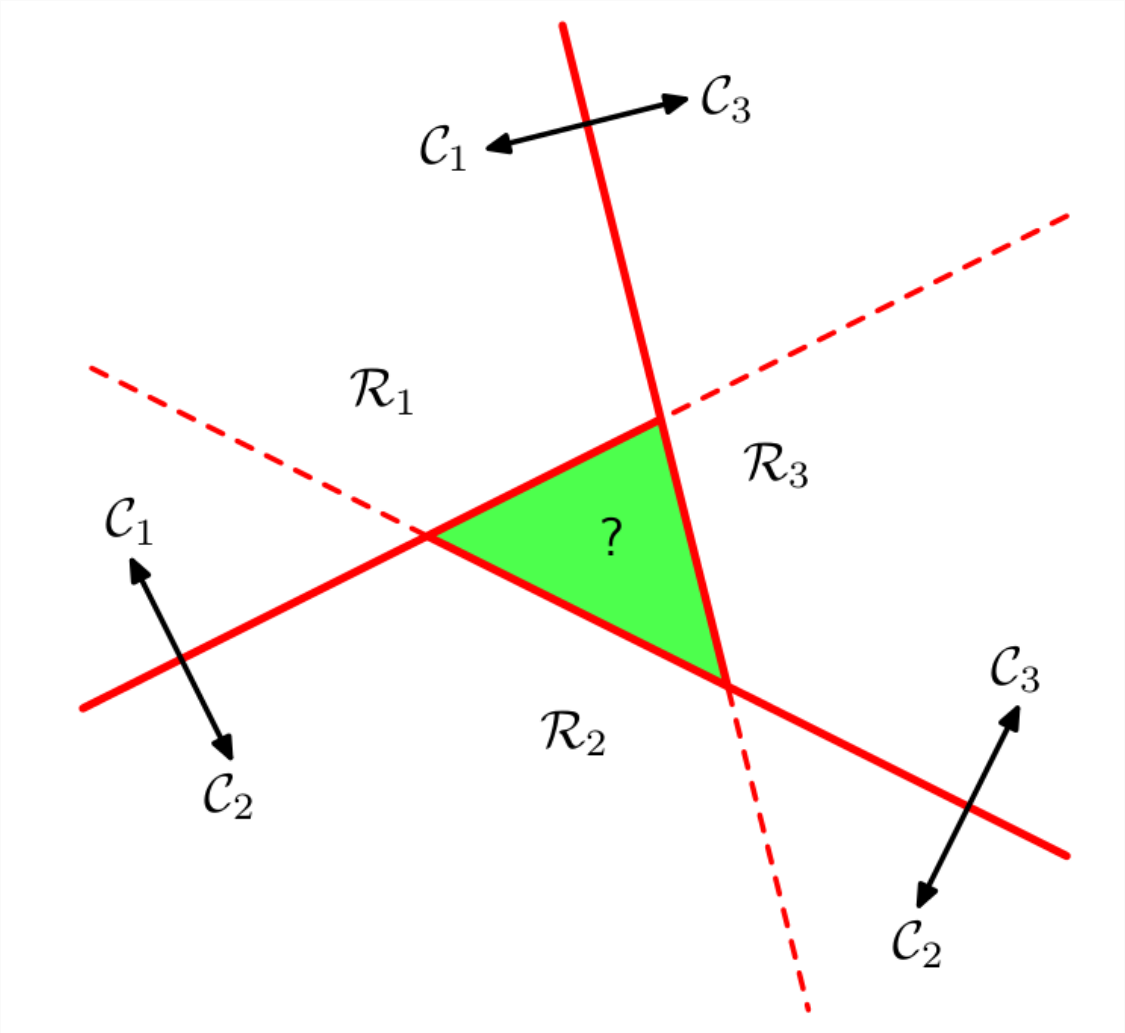
Несколько классов

- С несколькими классами выходит незадача.
- Можно рассмотреть K поверхностей вида «один против всех».
- Можно – $\binom{K}{2}$ поверхностей вида «каждый против каждого».
- Но всё это как-то нехорошо.

Несколько классов



Несколько классов



Несколько классов

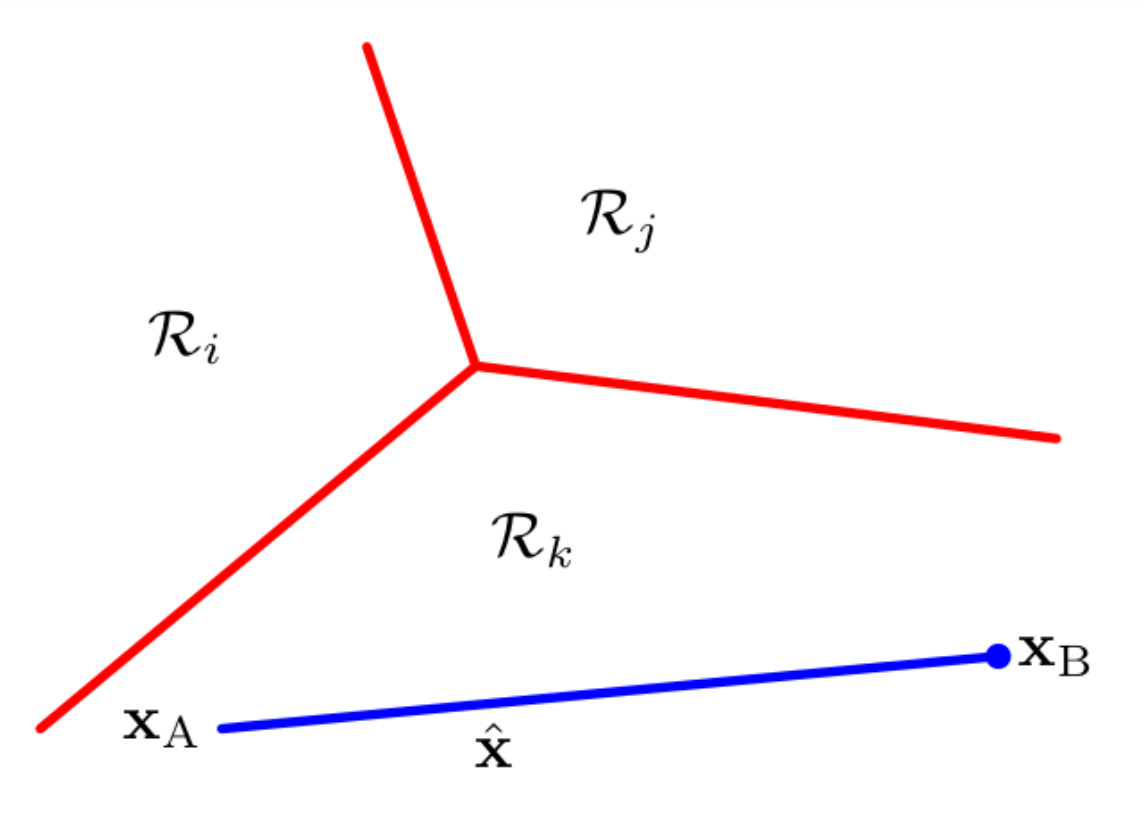
- Лучше рассмотреть единый дискриминант из K линейных функций:

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}.$$

- Классифицировать в $\mathcal{C}_k$, если $y_k(\mathbf{x})$ – максимален.
- Тогда разделяющая поверхность между $\mathcal{C}_k$ и $\mathcal{C}_j$ будет гиперплоскостью вида $y_k(\mathbf{x}) = y_j(\mathbf{x})$:

$$\left(\mathbf{w}_k - \mathbf{w}_j\right)^\top \mathbf{x} + \left(w_{k0} - w_{j0}\right).$$

Несколько классов



Упражнение. Докажите, что области, соответствующие классам, при таком подходе всегда односвязные и выпуклые.

Спасибо!

Спасибо за внимание!