Topics:

- Taylor Series
- What is supervised Machine Learning
- Engineering problems around Machine Learning
- Geometric interpretation of bias-variance trade-off
- Data-matrix
- Notion of regularization

# Taylor series

1. Named for the English mathematician *Taylor* (1685-1731)
2. We can control **p** and **n**. But $\theta \ from \ [0,1]$ is not our control.
3. $\boldsymbol{f^k}$ is take derivative to function k times
4. $\boldsymbol{f}: \mathbb{R} \to \mathbb{R}$
5. This form is called as a form of "Шлёмильха Роша".

$$f(x) = \sum_{k=0}^{n} \frac{f^k(x_0)}{k!}(x - x_0)^k + R_n(x)$$

$$R_n(x) = \frac{(x - x_0)^{n+1}(1 - \theta)^{n-p+1}}{pn!} f^{n+1}(x_0 + \theta(x - x_0))$$

# What is Supervised Machine Learning

***1. Model (or pattern) structure***

$\hat{F}(x) = \hat{F}(x; a) \in \mathcal{F}(a)$ *(Parametric way to define class of functions. Alternatives exist)*

***2. Score criteria***

$L(y, \hat{y})$ *is a scalar-value functions of two scalar arguments called* ***loss****. Usually:*

- *It states how we unhappy when real value* ***y*** *and we predict* $\hat{\boldsymbol{y}}$
- *L has minimum when* $\hat{y}$*=y. Also p*eople usually use some well-known loss
- *Via function L we create* ***prediction risk n data***

| Name | *Math form* |
|---|---|
| ***score on data.*** | $\widetilde{S(\boldsymbol{a})} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, F(x_i; \boldsymbol{a})) + \lambda P(a)$ (P(a) allow incorporate knowledge about model parameters) |
| ***optimal or target function*** $\boldsymbol{F^*}$ | $F^* = argmin_F S(F)$ |
| | |

***3. Search strategy. The most important thing.***

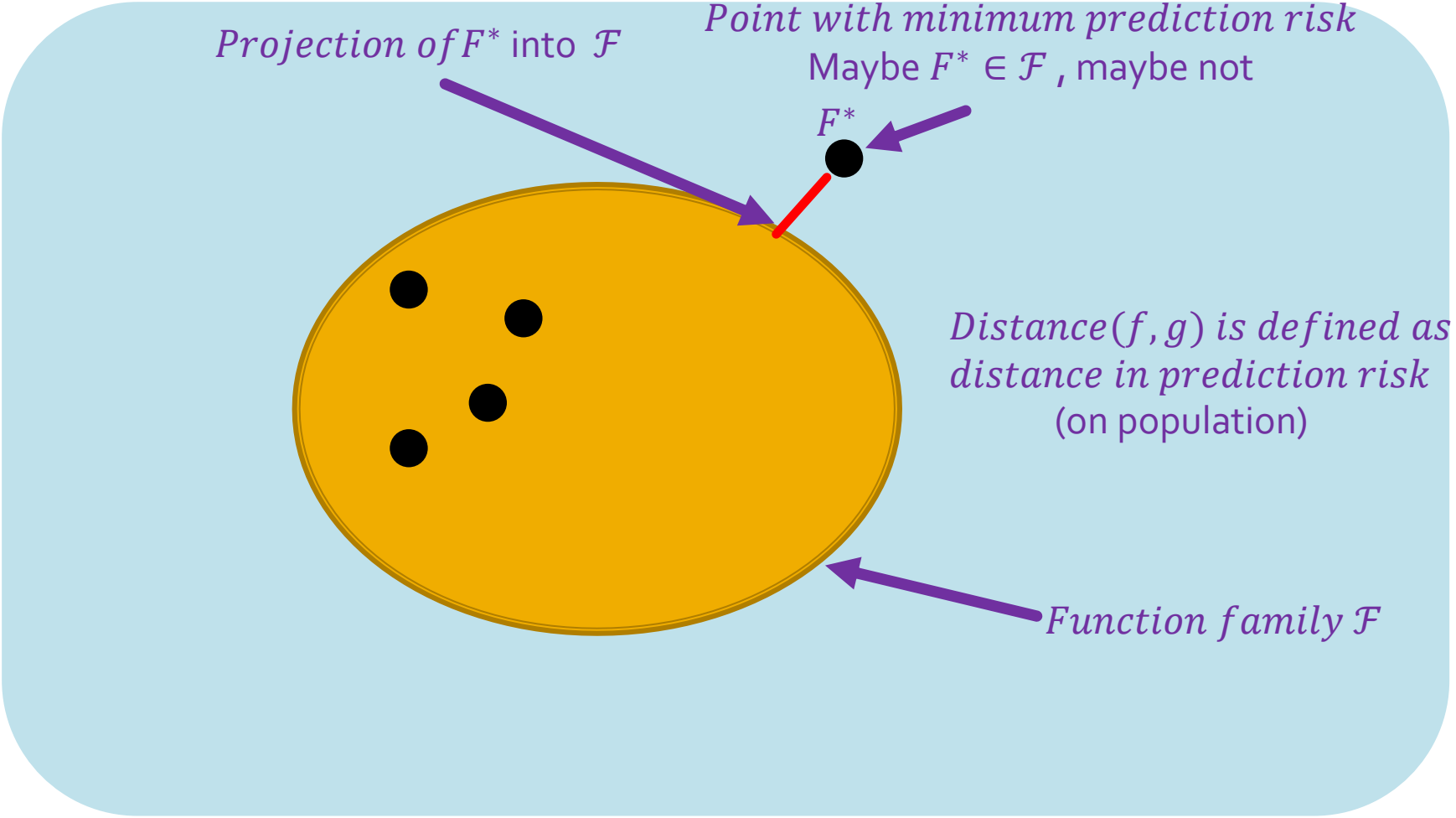
$a = argmin_a \widetilde{S(\boldsymbol{a})}$ *And arised optimization problem can be non-convex.*

***"it's more easy to define things then to solve things" – J.Friedman***

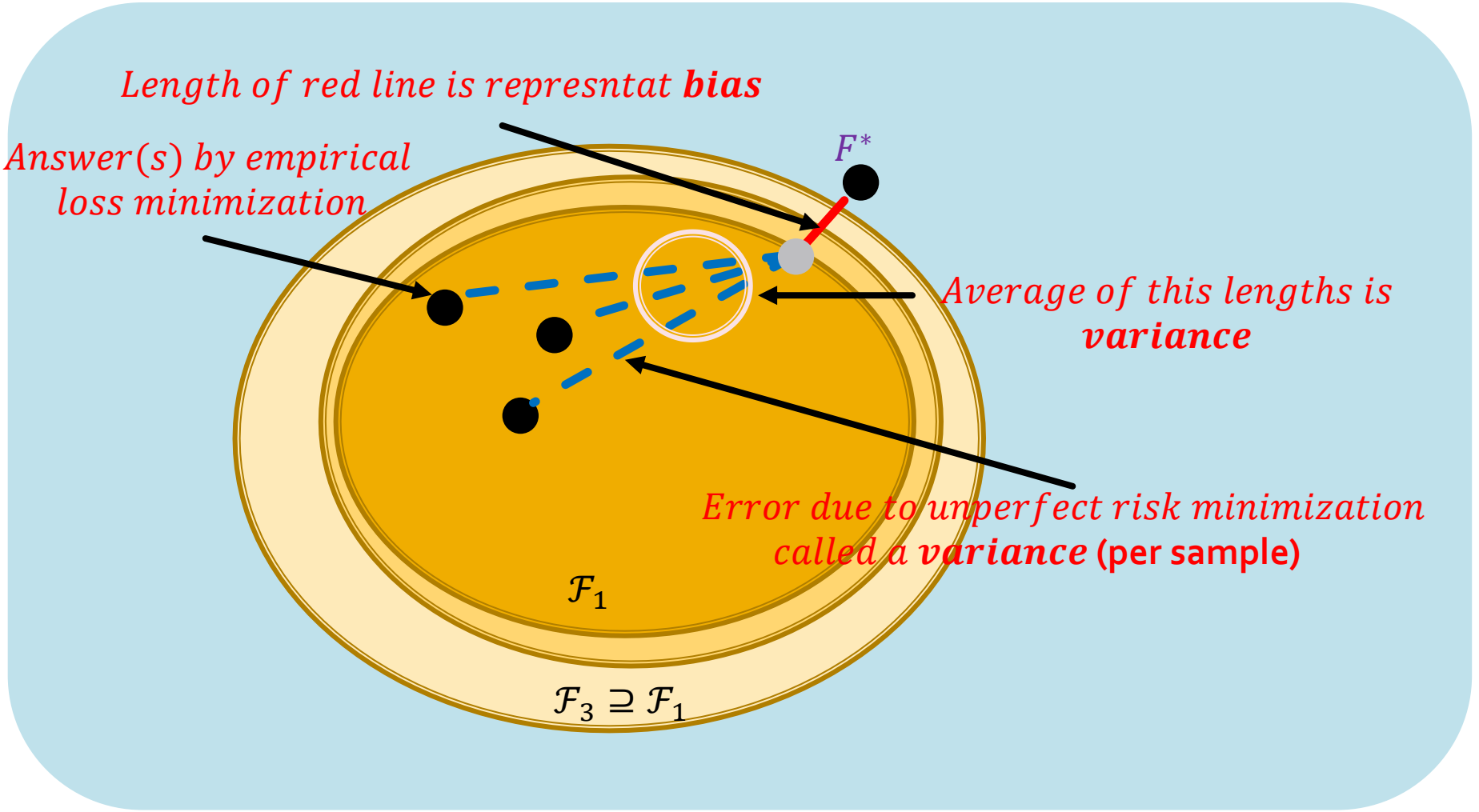
# Some engineering problems around supervised ML

- Usually we fit not to real samples of signal, but really into (signal + noise)
- We in general shouldn't believe a lot to provided data from user
- Complexity(size) of function space from which we will pick $X \rightarrow Y$ can affect into quality of model. If we can perfectly "fit  provied data" it's reasonable to assume that function space is BIG. And it lead to some troubles as we will see

# Bias-Variance tradeoff



# Bias-Variance tradeoff



# Bias-Variance tradeoff

So there are always two kind of errors in this business:
**Variance error** and **Bias error**:

**Reason of variance**
we do not know population and we use only data. Variance can also "be corrected" by more amount of data.
***It is about uncertanty which function is the best in case of limited amount of data.***

**Reason of bias**
Is that our function class not necessary contain target function. Bias can be fixed by consider more big function space.

In Machine Learning supervised methods:
Complete picture is that we consider series of nested function families and try to pick the best function via cross-validation.

# Building blocks fore store examples - Data Matrix

$x_{ij}$ - value of j-th attribute for i-th object.

| Attribute #1 | Attribute #2 | …Attribute #n | Y |
|---|---|---|---|
| $x_{11}$ | $x_{12}$ | $x_{1n}$ | $y_1$ |
| … | … | … | … |
| $x_{N1}$ | $x_{N2}$ | $x_{Nn}$ | $y_N$ |

This matrix is called "design matrix"|"data matrix"|"flat file" | "spreadsheet"

$n$ is number of columns –
(a.k.a number of "**measurement**"|"**attributes**"|"**variable**" | "**fields**" )

$N$ is number of rows –
(a.k.a. "**objects**"|"**samples**"|"**observations**"|"**examples**" in database)

It's only terminology, but still each community use it's own dialect

# About single layer neural network

- They are general in terms that they can approximate continuous function via appending extra units in hidden layer.
- Theorem said that you should have no deep layers if number of data is infinite, but it's not infinite.

# Regularization strategies

- *Regularization is a common scalarization technic for solve problem with two objectives*
- **This word have now different flavor in machine learning and in deep learning**.
- Regularization now is any activity to "*not perfect*" fit into train data.
- So even "*incorrect*" behavior of some part of algorithm for find model can be considered as regularization
- And it is very important because real goal of Machine Learning is to work well in future data, not in train date that have been given.

# Usual cross-validation

- In usual cross-validation we randomly split our all available data into two sets: **train(70%-80% of data)** and **test(30%-20% of data)**.

- We have a finite number of Predictors (or Models). There are three typical sources of this models:
  - 1. We have one predictor schema which use *n* predictor variables/features, but we want to perform feature selection, maybe to decrease number of used predictor variables (or number of features). We in some way received all possible subset $2^n$ , or some small part of it. Now we evaluate which model is better.

  - 2. We have several meta-parameters (or hyper-parameters or tuning parameters) for our predictor schema in ***objective which we minimize*** to obtain model that we will exploit in future.If they are categorical – we just enumerate them.

# Problem with usual cross-validation

- Models which had problem with high variance by itself, will have even more high variance during using this methodology, because train data is smaller. If there is small amount of data, then maybe we even can not fit all parameters of the model.