Bayesian Machine Learning for Shallow and Deep Models

Václav Šmídl,

[1] AI Center, FEL, CTU, Prague,
[2] RICE, FEL, UWB, Pilsen,

February 24, 2023

Real problem to be solved! Example in curve fitting

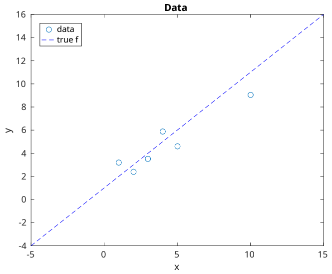
Fit by a linear function:

$$\begin{aligned} y_1 &= ax_1 &+b1, &\quad +e_1 \\ y_2 &= ax_2 &+b1 &\quad +e_2, \\ \vdots & \quad \vdots & \vdots & \quad \vdots \end{aligned}$$

In matrix notation $\theta = [a, b]^T$:

$$\mathbf{y} = X\theta + \mathbf{e},$$
$$\mathbf{e} = \mathbf{y} - X\theta$$

Minimize $\sum_i e_i^2 = \mathbf{e}^T\mathbf{e}$:



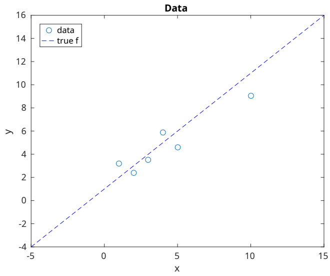
Real problem to be solved! Example in curve fitting

In matrix notation $\theta = [a, b]^T$:

$$\mathbf{y} = X\theta + \mathbf{e},$$
$$\mathbf{e} = \mathbf{y} - X\theta$$

Minimize $\sum_i e_i^2 = \mathbf{e}^T\mathbf{e}$:

$$\hat{\theta} = (X^TX)^{-1}X^T\mathbf{y}. \quad \hat{\sigma}^2 = \frac{1}{n}\mathbf{e}^T\mathbf{e},$$
$$\hat{y} = X\hat{\theta} + \hat{\sigma}\mathbf{e}$$



Real problem to be solved! Example in curve fitting

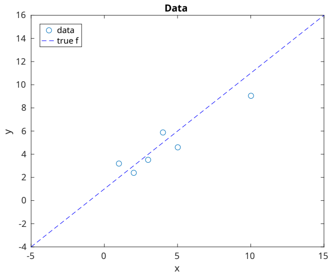
In matrix notation $\theta = [a, b]^T$:

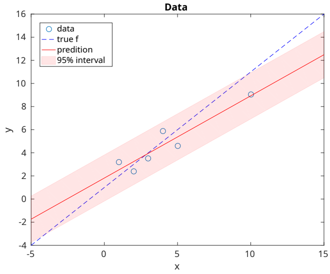
$$\mathbf{y} = X\theta + \mathbf{e},$$
$$\mathbf{e} = \mathbf{y} - X\theta$$

Minimize $\sum_i e_i^2 = \mathbf{e}^T\mathbf{e}$:

$$\hat{\theta} = (X^T X)^{-1} X^T \mathbf{y}. \quad \hat{\sigma}^2 = \frac{1}{n}\mathbf{e}^T\mathbf{e},$$

$$\hat{y} = X\hat{\theta} + \hat{\sigma}\mathbf{e}$$





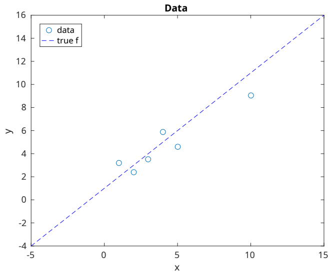
Real problem to be solved! Example in curve fitting

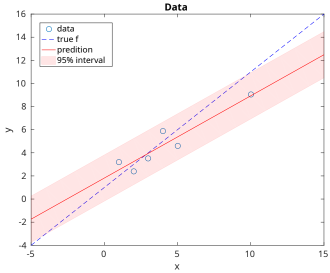
In matrix notation $\theta = [a, b]^T$:

$$\mathbf{y} = X\theta + \mathbf{e},$$
$$\mathbf{e} = \mathbf{y} - X\theta$$

Minimize $\sum_i e_i^2 = \mathbf{e}^T\mathbf{e}$:

$$\hat{\theta} = (X^T X)^{-1} X^T \mathbf{y}. \quad \hat{\sigma}^2 = \frac{1}{n}\mathbf{e}^T\mathbf{e},$$
$$\hat{y} = X\hat{\theta} + \hat{\sigma}\mathbf{e}$$



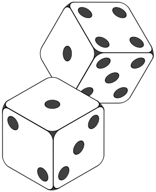


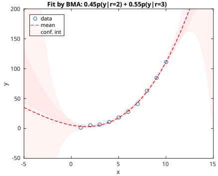
Overconfidence! The answer is correct only asymptotically ($\mathcal{O}(n^{-1})$)

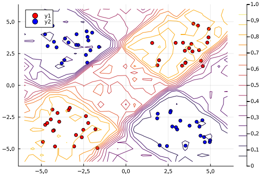
- we never have infinite dataset or large enough
- we need to handle the information with care!

Roadmap:

Theory | Shallow models | Deep models







Bayesian (Laplace) philosophy

Bayesian $=$

Bayesian (Laplace) philosophy

Bayesian = someone who uses **probability** calculus to quantify **uncertainty**.

Bayesian (Laplace) philosophy

Bayesian = someone who uses **probability** calculus to quantify **uncertainty**.

Justification: Uncertainty and randomness have the same effect on decision-making.

Bayesian = someone who uses **probability** calculus to quantify **uncertainty**.

Justification: Uncertainty and randomness have the same effect on decision-making.

Distance of a star: measuring distance to stars has large observation error, say we measure 10ly $\pm$1ly

Bayesian (Laplace) philosophy

Bayesian = someone who uses **probability** calculus to quantify **uncertainty**.

Justification: Uncertainty and randomness have the same effect on decision-making.

Distance of a star: measuring distance to stars has large observation error, say we measure 10ly $\pm$1ly

Can I say that the distance of the star has Normal distribution: $\mathcal{N}(10, 1)$?

- No: the distance is not random
- Yes: you are a Bayesian seeing distance as a **degree of belief**,

Probability of an event

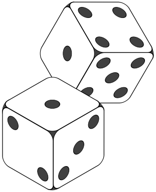
Probability=Frequency of an event:

$$P(x) = \frac{\#\,\text{realizations}}{\#\,\text{trials}}$$

Probability of an event

Probability=Frequency of an event:

$$P(x) = \frac{\#\,\text{realizations}}{\#\,\text{trials}}$$

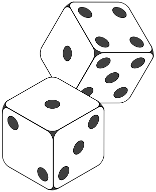


$P(x = 1) = \frac{1}{6}$

Probability of an event

Probability=Frequency of an event:

$$P(x) = \frac{\#\,\text{realizations}}{\#\,\text{trials}}$$

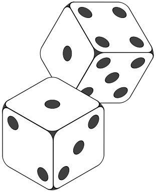


$P(x = 1) = \frac{1}{6}$

Probability of an event

Probability=Frequency of an event:

$$P(x) = \frac{\#\,\text{realizations}}{\#\,\text{trials}}$$



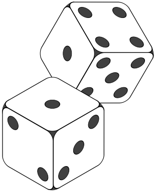
$P(x=1) = \frac{1}{6}$

Probability of an event

Frequentist:

Probability=Frequency of an event:

$$P(x) = \frac{\#\,\text{realizations}}{\#\,\text{trials}}$$



$P(x = 1) = \frac{1}{6}$

Bayesian:

Frequency:

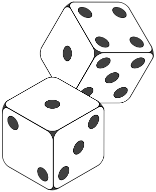
$$P(\text{Sparta beats Slavia}) = \frac{133}{294} \approx 45\%$$

Probability of an event

Frequentist:

Probability=Frequency of an event:

$$P(x) = \frac{\#\,\text{realizations}}{\#\,\text{trials}}$$



$P(x = 1) = \frac{1}{6}$

Bayesian:

Frequency:

$$P(\text{Sparta beats Slavia}) = \frac{133}{294} \approx 45\%$$

Degree (state) of belief:

$$P(x|d) = \frac{P(d|x)P(x)}{\sum_x P(d|x)P(x)}$$

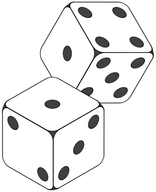
$P(\text{Sparta vs. Slavia} = 1) = 1/1.8$

Probability of an event

Frequentist:

Probability=Frequency of an event:

$$P(x) = \frac{\#\,\text{realizations}}{\#\,\text{trials}}$$



$$P(x = 1) = \tfrac{1}{6}$$

Bayesian:

Frequency:

$$P(\text{Sparta beats Slavia}) = \frac{133}{294} \approx 45\%$$

Degree (state) of belief:

$$P(x|d) = \frac{P(d|x)P(x)}{\sum_x P(d|x)P(x)}$$

$$P(\text{Sparta vs. Slavia} = 1) = 1/1.8$$

Same probability calculus

Different [1] role of prior $P(x)$, applications and methods

1. Product rule (Chain rule)

$$P(X, Y) \quad = \quad P(X|Y)P(Y),$$

2. Sum rule (Marginalization)

$$P(X) \quad = \quad \sum_Y P(X, Y)$$

All you need is rules: Rules of probability

1. Product rule (Chain rule)

$$\begin{aligned} P(X,Y) &= P(X|Y)P(Y), \\ &= P(X)P(Y|X) \end{aligned}$$

2. Sum rule (Marginalization)

$$\begin{aligned} P(X) &= \textstyle\sum_Y P(X,Y) \\ P(Y) &= \textstyle\sum_X P(X,Y) \end{aligned}$$

Derived $P(X) = \sum_Y P(X|Y)P(Y)$

Continuous distributions: $p(x) = dF(x)$ (engineering notation)

2. Sum for Continuous distribution

$$p(x) = \int p(x,y)dy$$

Bayes Rule

From chain rule:

$$P(X|Y)P(Y) = P(Y|X)P(X).$$

$$P(X|Y) = \frac{P(Y|X)P(X).}{P(Y)}$$

Bayes Rule

From chain rule:

$$P(X|Y)P(Y) = P(Y|X)P(X).$$

$$P(X|Y) = \frac{P(Y|X)P(X).}{P(Y)}$$

Application: θ is a parameter, D is a random observation

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

Bayes Rule

From chain rule:

$$P(X|Y)P(Y) = P(Y|X)P(X).$$
$$P(X|Y) = \frac{P(Y|X)P(X).}{P(Y)}$$

Application: θ is a parameter, D is a random observation

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

Philosophical issue:

Frequentists: parameter is NOT a random quantity, $p(\theta)$ should not exist.

Bayesian: $p(\theta|D)$ is our degree of belief in parameter values.

Consequences of being a Bayesian

Uncertain quantities are modeled by probability.

Incremental learning: two data sets $\mathcal{D}_1$ and $\mathcal{D}_2$. Learning from the first

$$p(\theta|\mathcal{D}_1) \propto p(\mathcal{D}_1|\theta)p(\theta)$$

and later from the second:

$$p(\theta|\mathcal{D}_1, \mathcal{D}_2) \propto p(\mathcal{D}_2|\theta)p(\theta|\mathcal{D}_1).$$

Model selection: we have multiple possible models $\mathcal{M}_1 \dots \mathcal{M}_n$ and do not know which is correct. Model is uncertain.

Consequences of being a Bayesian

Uncertain quantities are modeled by probability.

Incremental learning: two data sets $\mathcal{D}_1$ and $\mathcal{D}_2$. Learning from the first

$$p(\theta|\mathcal{D}_1) \propto p(\mathcal{D}_1|\theta)p(\theta)$$

and later from the second:

$$p(\theta|\mathcal{D}_1, \mathcal{D}_2) \propto p(\mathcal{D}_2|\theta)p(\theta|\mathcal{D}_1).$$

Model selection: we have multiple possible models $\mathcal{M}_1 \ldots \mathcal{M}_n$ and do not know which is correct. Model is uncertain.
Unknown variable $m \in \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ and seek $p(m)$.

Consequences of being a Bayesian

Uncertain quantities are modeled by probability.

Incremental learning: two data sets $\mathcal{D}_1$ and $\mathcal{D}_2$. Learning from the first

$$p(\theta|\mathcal{D}_1) \propto p(\mathcal{D}_1|\theta)p(\theta)$$

and later from the second:

$$p(\theta|\mathcal{D}_1, \mathcal{D}_2) \propto p(\mathcal{D}_2|\theta)p(\theta|\mathcal{D}_1).$$

Model selection: we have multiple possible models $\mathcal{M}_1 \dots \mathcal{M}_n$ and do not know which is correct. Model is uncertain.
Unknown variable $m \in \{\mathcal{M}_1, \dots, \mathcal{M}_n\}$ and seek $p(m)$.

Nuisance parameters: to define parameters θ we often need "regularization" parameters, η (hyper-parameters). Hyper-parameters are uncertain.

Consequences of being a Bayesian

Uncertain quantities are modeled by probability.

Incremental learning: two data sets $\mathcal{D}_1$ and $\mathcal{D}_2$. Learning from the first

$$p(\theta|\mathcal{D}_1) \propto p(\mathcal{D}_1|\theta)p(\theta)$$

and later from the second:

$$p(\theta|\mathcal{D}_1, \mathcal{D}_2) \propto p(\mathcal{D}_2|\theta)p(\theta|\mathcal{D}_1).$$

Model selection: we have multiple possible models $\mathcal{M}_1 \ldots \mathcal{M}_n$ and do not know which is correct. Model is uncertain.
Unknown variable $m \in \{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$ and seek $p(m)$.

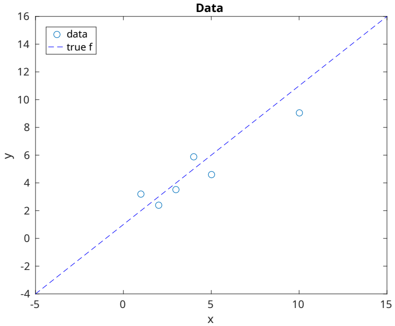
Nuisance parameters: to define parameters θ we often need "regularization" parameters, η (hyper-parameters). Hyper-parameters are uncertain.
We seek $p(\theta, \eta|\mathcal{D})$, or marginal $p(\theta|\mathcal{D}) = \int p(\theta, \eta|\mathcal{D})d\eta$

Example: curve fitting

Linear regression:

$$\mathbf{y} = X\theta + \mathbf{e},$$
$$p(\mathbf{y}|X,\theta) = \mathcal{N}(X\theta, \sigma I)$$

Estimating the parameter



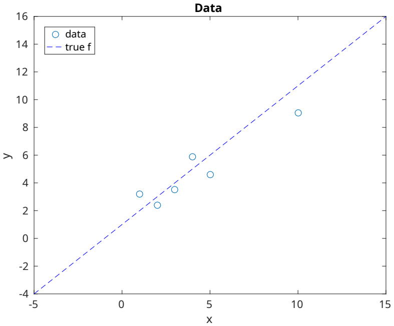
Example: curve fitting

Linear regression:

$$\mathbf{y} = X\theta + \mathbf{e},$$
$$p(\mathbf{y}|X,\theta) = \mathcal{N}(X\theta, \sigma I)$$

Estimating the parameter

$$\begin{aligned} p(\theta|X,\mathbf{y}) &\propto p(\mathbf{y}|X,\theta)p(\theta) \\ &= \mathcal{N}(\mu_\theta, \Sigma_\theta) \\ \mu_\theta &= (X^T X)^{-1} X^T \mathbf{y}. \\ \Sigma_\theta &= (X^T X)^{-1}. \end{aligned}$$



Example: curve fitting

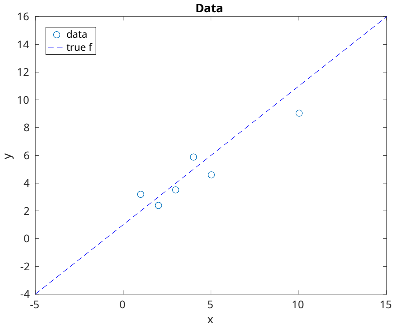
Linear regression:

$$\mathbf{y} = X\theta + \mathbf{e},$$
$$p(\mathbf{y}|X,\theta) = \mathcal{N}(X\theta, \sigma I)$$

Estimating the parameter

$$\begin{aligned} p(\theta|X,\mathbf{y}) &\propto p(\mathbf{y}|X,\theta)p(\theta) \\ &= \mathcal{N}(\mu_\theta, \Sigma_\theta) \\ \mu_\theta &= (X^T X)^{-1} X^T \mathbf{y}. \\ \Sigma_\theta &= (X^T X)^{-1}. \end{aligned}$$

Isn't it the same as before? What is the use for Σ_θ?

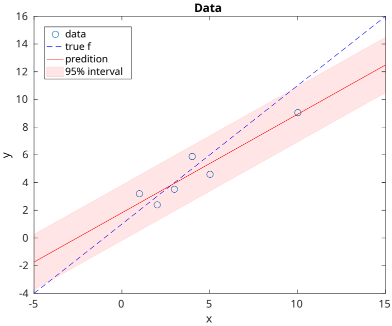


Prediction

Prediction with LS estimate:

$$\hat{y} = X\hat{\theta} + e.$$

Known variance of e.
Why it does not extrapolate well?



Prediction

Prediction with LS estimate:

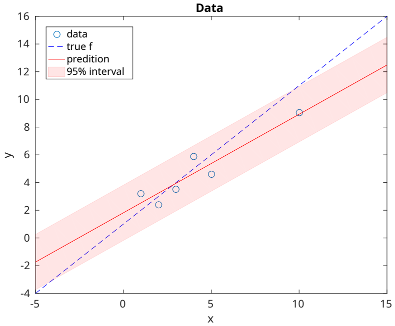
$$\hat{y} = X\hat{\theta} + e.$$

Known variance of e.
Why it does not extrapolate well?

Bayesian explanation
Prediction

$$\hat{y} \sim p(y'|\hat{\theta}),$$

assumes **certainty** in estimate of $\hat{\theta}$.



Prediction

Prediction with LS estimate:

$$\hat{y} = X\hat{\theta} + e.$$

Known variance of e.
Why it does not extrapolate well?

Bayesian explanation
Prediction

$$\hat{y} \sim p(y'|\hat{\theta}),$$

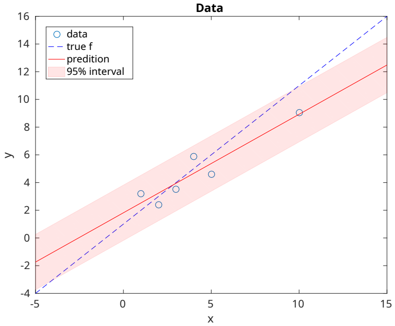
assumes **certainty** in estimate of $\hat{\theta}$.

- All that is certain is the data!

$$\hat{y} \sim p(y'|y, X)$$

- Working out the rules:

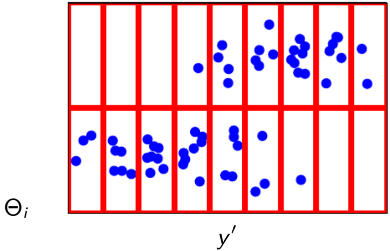
$$p(y'|y, X) = \int p(y'|\theta)p(\theta|y, X)d\theta$$



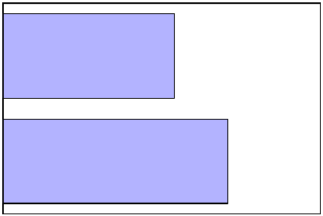
Intuition behind marginalizaton

Definitely not exact math! $\theta \in \{\Theta_1, \Theta_2\}$

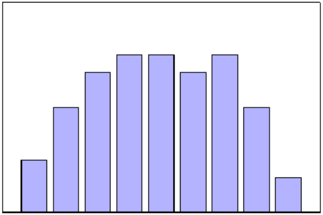
$p(y', \theta | y, X)$



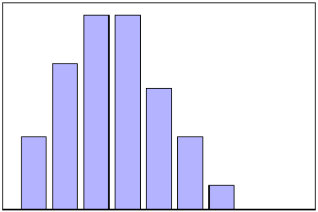
$p(\theta | y, X)$



$p(y' | y, X)$



$p(y' | \theta = \Theta_2)$



Bayesian Prediction

- Bayesian prediction:

$$p(y'|y, X) = \int p(y'|\theta)p(\theta|y, X)d\theta$$

- Posterior probability

$$p(\theta|y, X) \propto p(y|\theta, X)p(\theta)$$

for choices:

$$\begin{aligned} p(y|\theta, X) &= \mathcal{N}(X\theta, 1), \\ \log p(y|\theta, X) &= -\frac{1}{2}(y - X\theta)^\top(y - X\theta) + c, \end{aligned}$$

Bayesian Prediction

- Bayesian prediction:

$$p(y'|y, X) = \int p(y'|\theta)p(\theta|y, X)d\theta$$

- Posterior probability

$$p(\theta|y, X) \propto p(y|\theta, X)p(\theta)$$

for choices:

$$\begin{aligned} p(y|\theta, X) &= \mathcal{N}(X\theta, 1), \\ \log p(y|\theta, X) &= -\frac{1}{2}(y - X\theta)^\top(y - X\theta) + c, \end{aligned}$$

- Solution

$$\begin{aligned} p(\theta|y, X) &= \mathcal{N}(\hat{\theta}, S_n), \\ \hat{\theta} &= (X'X)^{-1}X'y, \quad S_n = (X'X)^{-1}. \end{aligned}$$

Bayesian Prediction

- Bayesian prediction:

$$p(y'|y,X) = \int p(y'|\theta)p(\theta|y,X)d\theta$$

- Posterior probability

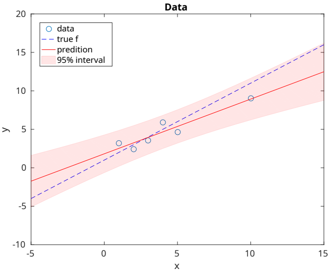
$$p(\theta|y,X) \propto p(y|\theta,X)p(\theta)$$

for choices:

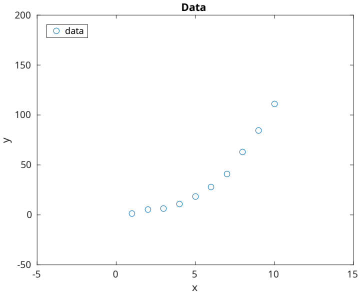
$$p(y|\theta,X) = \mathcal{N}(X\theta, 1),$$
$$\log p(y|\theta,X) = -\frac{1}{2}(y - X\theta)^\top (y - X\theta) + c,$$

- Solution

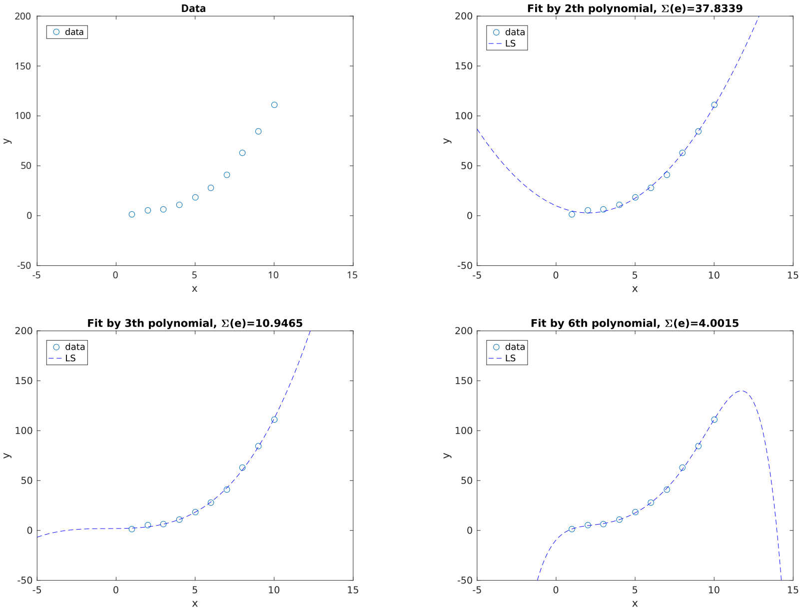
$$\begin{aligned} p(\theta|y,X) &= \mathcal{N}(\hat{\theta}, S_n), \\ \hat{\theta} &= (X'X)^{-1}X'y, \quad S_n = (X'X)^{-1}. \\ p(y'|y,X) &= \mathcal{N}(X\hat{\theta}, 1 + [1,x]S_n[1,x]^\top) \end{aligned}$$



Challenge: curve fitting



Challenge: curve fitting



What is wrong with minimization?

1. The error of the fit is minimized
   - over-fitting,
2. Model complexity is not taken into account
3. How the humans decide?

What is wrong with minimization?

1. The error of the fit is minimized
    - over-fitting,
2. Model complexity is not taken into account
3. How the humans decide?

- Potentially many answers
    - penalization / regularization terms,
    - information criteria
    - cross validation testing / training data,

What is wrong with minimization?

1. The error of the fit is minimized
   - over-fitting,
2. Model complexity is not taken into account
3. How the humans decide?

- Potentially many answers
  - penalization / regularization terms,
  - information criteria
  - cross validation testing / training data,

- Bayesian answer:
  - admit that the model order is **unknown**.

Bayesian Model Selection

- **Unknown** quantity: model order r has distribution $p(r|y, X)$
- Known data: $\mathbf{y}, X$ with model $p(\mathbf{y}|\theta, X, r) = N(X\theta, 1)$,

Looking for $p(r|\mathbf{y}, X)$:

1. Bayes rule

$$p(r|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, r)p(r)}{\sum_r p(\mathbf{y}|X, r)p(r)}, \qquad p(r) = ?$$

Bayesian Model Selection

- **Unknown** quantity: model order r has distribution $p(r|y, X)$
- Known data: $\mathbf{y}, X$ with model $p(\mathbf{y}|\theta, X, r) = N(X\theta, 1)$,

Looking for $p(r|\mathbf{y}, X)$:

1. Bayes rule
$$p(r|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, r)p(r)}{\sum_r p(\mathbf{y}|X, r)p(r)}, \qquad p(r) =?$$
2. Marginalization
$$p(\mathbf{y}|X, r) = \int p(\mathbf{y}, \theta|X, r)d\theta$$

Bayesian Model Selection

- **Unknown** quantity: model order r has distribution $p(r|y, X)$
- Known data: $\mathbf{y}, X$ with model $p(\mathbf{y}|\theta, X, r) = N(X\theta, 1)$,

Looking for $p(r|\mathbf{y}, X)$:

1. Bayes rule
$$p(r|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, r)p(r)}{\sum_r p(\mathbf{y}|X, r)p(r)}, \qquad p(r) = ?$$
2. Marginalization
$$p(\mathbf{y}|X, r) = \int p(\mathbf{y}, \theta|X, r)d\theta$$
3. Chain rule
$$p(\mathbf{y}, \theta|X, r) = p(\mathbf{y}|\theta, X, r)p(\theta|r), \qquad p(\theta|r) = ?$$

Bayesian Model Selection

- **Unknown** quantity: model order r has distribution $p(r|y, X)$
- Known data: $\mathbf{y}, X$ with model $p(\mathbf{y}|\theta, X, r) = N(X\theta, 1)$,

Looking for $p(r|\mathbf{y}, X)$:

1. Bayes rule

$$p(r|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, r)p(r)}{\sum_r p(\mathbf{y}|X, r)p(r)}, \qquad p(r) = 1/r_{max}$$

2. Marginalization

$$p(\mathbf{y}|X, r) = \int p(\mathbf{y}, \theta|X, r)d\theta$$

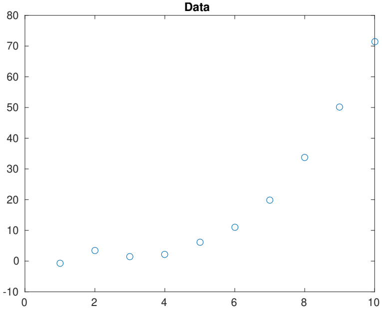
3. Chain rule

$$p(\mathbf{y}, \theta|X, r) = p(\mathbf{y}|\theta, X, r)p(\theta|r), \qquad p(\theta|r) = N(0, \alpha I)$$

Solution:

$$p(r|\mathbf{y}, X, \alpha) \propto \left|X^T X + \alpha I\right|^{-1/2} \exp\left(-\frac{1}{2}\hat{\theta}\left(X^T X + \alpha I\right)\hat{\theta}\right)$$

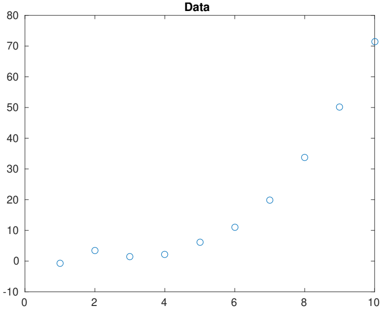
Application of the polynomial



α	1e-8	1e-6	1e-4	"best"
$P(x=2)$	44%	8%	1%	44%
$P(x=3)$	55%	92%	99%	55%
$P(x=4)$	0%	0%	0%	0%

How to choose α?

Application of the polynomial

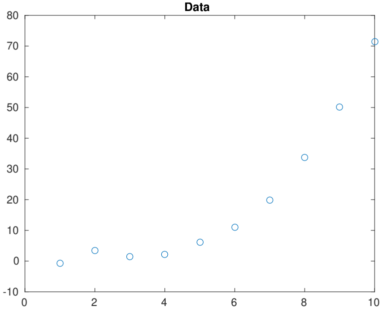


α	1e-8	1e-6	1e-4	"best"
$P(x=2)$	44%	8%	1%	44%
$P(x=3)$	55%	92%	99%	55%
$P(x=4)$	0%	0%	0%	0%

How to choose α?

- assume α an unknown **hyperparametr**
- **uncertainty => hierarchical** prior $p(\alpha) = \Gamma(\gamma, \delta)$.
- solve $p(r|y, X) = \int p(r|y, x, \alpha)p(\alpha)d\alpha$

Application of the polynomial

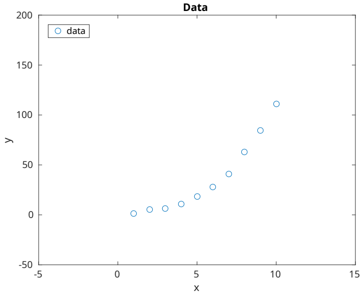


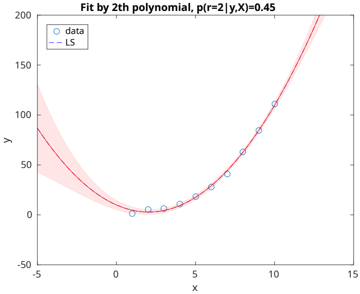
α	1e-8	1e-6	1e-4	"best"
$P(x=2)$	44%	8%	1%	44%
$P(x=3)$	55%	92%	99%	55%
$P(x=4)$	0%	0%	0%	0%

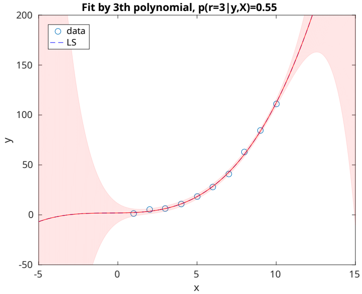
How to choose α?

- assume α an unknown **hyperparametr**
- **uncertainty => hierarchical** prior $p(\alpha) = \Gamma(\gamma, \delta)$.
- solve $p(r|y, X) = \int p(r|y, x, \alpha)p(\alpha)d\alpha$
- works for $\gamma = \delta = 0$ which is Jeffrey's improper prior $p(\alpha) \propto 1/\alpha$,
  - Recursion ends! no need for next hierarchy.

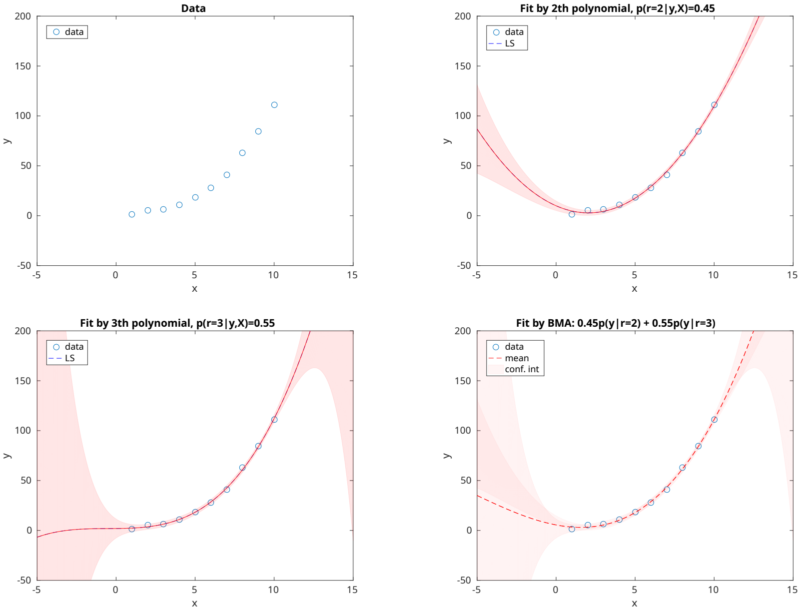
Bayesian prediction:







Bayesian prediction:

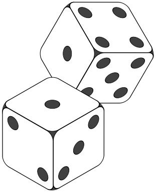


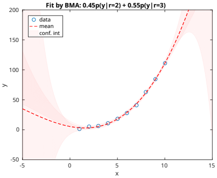
Roadmap

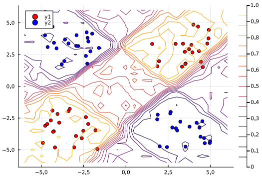
Theory

Shallow models

Deep models







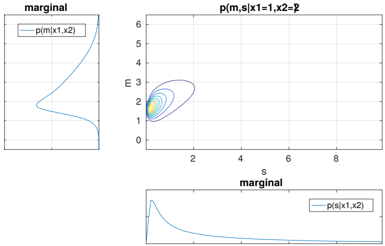
OK, I trust you, lets use it for my fancy model!

Not so fast!

OK, I trust you, lets use it for my fancy model!

Not so fast!

- Posterior density of linear model is tractable only for $p(\theta|\sigma)$, not for $p(\theta) = \mathcal{N}(0, \tau)$!
- Non-linear models are out of question.



OK, I trust you, lets use it for my fancy model!

Not so fast!

- Posterior density of linear model is tractable only for $p(\theta|\sigma)$, not for $p(\theta) = \mathcal{N}(0, \tau)$!
- Non-linear models are out of question.

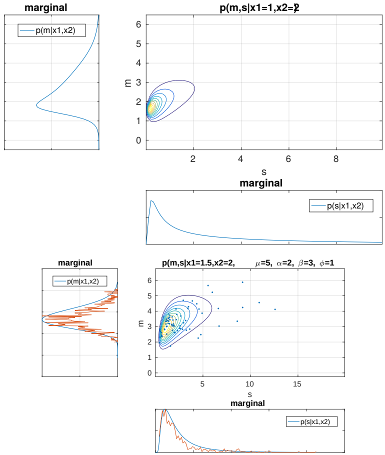
Monte-Carlo for the rescue!

- probability density approximated by empirical densities

$$p(\theta|y, X) \approx \frac{1}{J}\sum_{j=1}^{J}(\theta - \theta^{(j)})$$

with trivial integration

$$p(y'|y, X) = \frac{1}{J}\sum_{j=1}^{J} p(y|X, \theta^{(j)})$$



I am for it, sample it for me!

Sure. Meet probabilistic programming.

I am for it, sample it for me!

Sure. Meet probabilistic programming.

STAN: https://mc-stan.org/

- HMC, NUTS
- Variational inference
- Matlab, R, Mathematica, Python, ...

Turing.jl: https://github.com/TuringLang/Turing.jl

- HMC, NUTS, SMC, **PG**
- Julia

PyMC3:

- Python

```
addpath('MatlabStan')
linmodel = {
'model {'
    '  y ~ normal(alpha*(1 - exp(-beta * x))+gamma, sigma);'
'}'
'parameters {'
    '  real<lower=0> alpha;'
    '  real<lower=0> beta;'
    '  real gamma;'
    '  real<lower=0,upper=0.1> sigma;'
'}'
'data {'
    '  int<lower=0> N;'
    '  vector[N] x;'
    '  vector[N] y;'
'}'
};
```

```
@model gdemo(x) = begin
  s ~ InverseGamma(2,3)
  m ~ Normal(0, sqrt(s))
  x[1] ~ Normal(m, sqrt(s))
  x[2] ~ Normal(m, sqrt(s))
  return s, m
end

chain = sample(gdemo([1.5, 2.0]), SGLD(10000, 0.5))
```

Even Neural networks?

Consider a classification problem of
2d input space.

$$y = f_\theta([x_1, x_2])$$
$$y \in \{0, 1\}$$

with MLP (2->3->2->1)

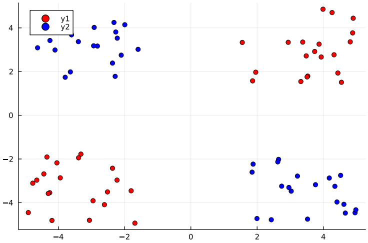
$$\hat{y} = \sigma(W_3 \text{th}(W_2 \text{th}(W_1 x + b_1) + b_2) + b_3)$$
$$\theta = [W_1, b_1, W_2, b_2, W_3, b_3]$$

with prediction error:

$$CE(\hat{y}, y) = -\left(y \log \hat{y} + (1 - y) \log(1 - \hat{y})\right)$$

training using gradient descent.

Even Neural networks?



Consider a classification problem of 2d input space.

$$y = f_\theta([x_1, x_2])$$
$$y \in \{0, 1\}$$

with MLP (2->3->2->1)

$$\hat{y} = \sigma(W_3 \text{th}(W_2 \text{th}(W_1 x + b_1) + b_2) + b_3)$$
$$\theta = [W_1, b_1, W_2, b_2, W_3, b_3]$$

with prediction error:

$$CE(\hat{y}, y) = -\left(y \log \hat{y} + (1 - y) \log(1 - \hat{y})\right)$$

training using gradient descent.

- prediction $\hat{y} \in (0, 1)$ – is it a probability?
- probability of observation

$$p(y|\theta, x) = \mathcal{B}e(f_\theta(x))$$

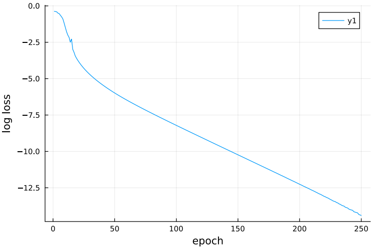
Standard NN: Gradient descent

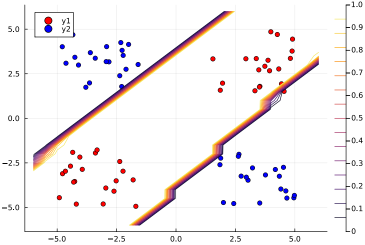
Training with GD

- today with ADAM

Contour of network output on scale (0,1)

- can we trust it?





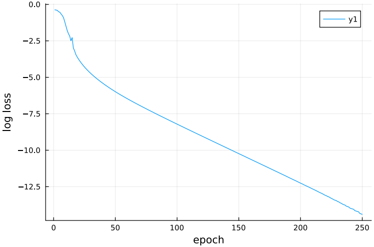
Standard NN: Gradient descent

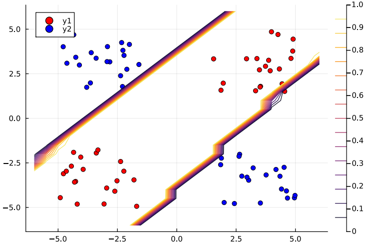
Training with GD

- today with ADAM

Contour of network output on scale (0,1)

- can we trust it?
- what is wrong?
  - Trust in one parametric value, $\hat{\theta}$.
  - insufficient data
  - is it not probabilistic?

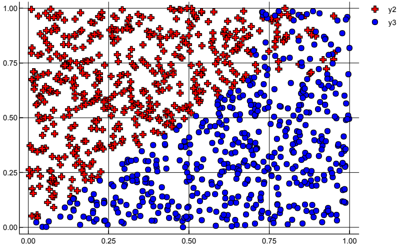


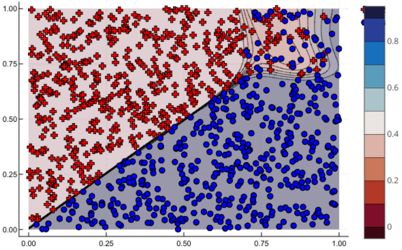


Two different uncertainties

Uncertainty:

- aleatoric – in the data – ML estimation handle well
- epistemic – missing data – Bayes handles well





Bayes in NN

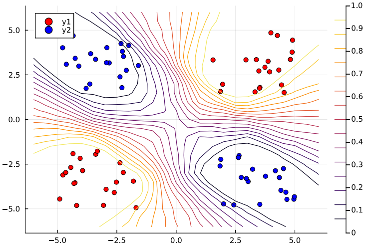
Sample θ.

Use probabilistic programming:

```julia
@model function bayes_nn(xs, ts, nparameters, reconstruct; alpha=0.09)
    # Create the weight and bias vector.
    parameters ~ MvNormal(Zeros(nparameters), I / alpha)

    # Construct NN from parameters
    nn = reconstruct(parameters)
    # Forward NN to make predictions
    preds = nn(xs)

    # Observe each prediction.
    for i in 1:length(ts)
        ts[i] ~ Bernoulli(preds[i])
    end
end;
```



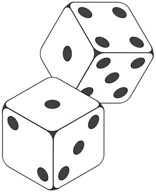
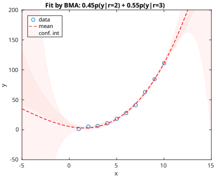
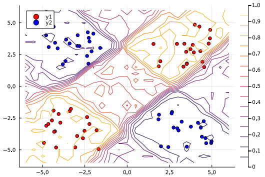
NUTS sampler generates 5000 estimates (NN).

Prediction

$$y = \frac{1}{5000} \sum_j f(x, \theta^{(j)})$$

Took 20min to sample.

Roadmap:

Theory	Shallow models	**Deep models**
		

Deep learning: networks are huge

Proper sampling is too expensive!

Something fast and “close enough”:

1. Running the task many times from different initial conditions
   - Deep Ensembles
2. Stochastic Gradient Descent
   - Langevin Dynamics
   - Stepsize Tuners
3. Dropout
   - Dropout Monte Carlo

Stochastic Gradient Descent: faster training

Instead of optimizing loss for all data

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\theta), \quad \mathcal{L}(\theta) = \sum_{i=1}^{n} \mathcal{L}(x_i, y_i, \theta),$$
$$\hat{\theta}^{(\tau+1)} = \hat{\theta}^{(\tau)} - \eta \nabla \mathcal{L}(\hat{\theta}^{(\tau)}),$$

we create a subsample of the indeces in each iteration!

$$\mathcal{I} \subset \{1, \ldots, n\}, |\mathcal{I}| < n,$$
$$\tilde{\mathcal{L}}_\tau = \sum_{i \in \mathcal{I}^\tau} \mathcal{L}(x_i, y_i, \theta)$$

Stochastic GD:

$$\hat{\theta}^{(\tau+1)} = \hat{\theta}^{(\tau)} - \eta \nabla \tilde{L}(\hat{\theta}^{(\tau)}),$$

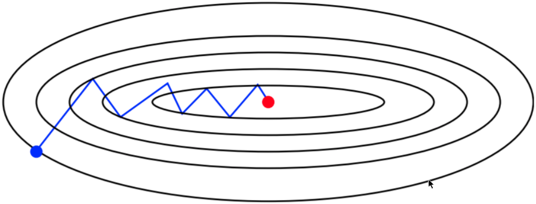
satisfies conditions

$$\nabla_\theta \mathcal{L}(y, x, \theta) = \mathsf{E}\left(\nabla \tilde{\mathcal{L}}(y, x, \theta)\right)$$

and converges to the same solution for decreasing learning rate $\sum_\tau \eta = \infty, \sum_\tau \eta^2 < \infty$.

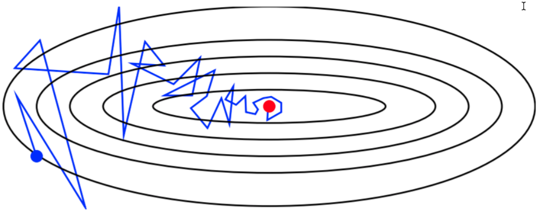
Stochastic Gradient Descent

Deterministic gradient:



Stochastic gradient: will converge only if $\eta_\tau \to 0$.

For constant η_τ it “walks” around optima. **Does it sample in Bayesian sense?**



SGD is Approximate Bayesian Inference

SDG is a discretization of approximation of random walk model

$$\nabla\tilde{\mathcal{L}}(\theta) \approx \nabla\mathcal{L}(\theta) + \frac{1}{\sqrt{S}}\Delta, \qquad \Delta \sim \mathcal{N}(0, C(\theta))$$

If the loss function can be approximated by quadratic function

$$\mathcal{L}(\theta) = \frac{1}{2}\theta^\top A\theta,$$

then posterior factor $q(\theta) = \mathcal{N}(\hat{\theta}, \Sigma)$ satisfies:

$$\Sigma A + A\Sigma = \frac{\eta}{S}C(\theta).$$

Minimizing KL to $p(\theta)$ yields (Mandt, Hoffman, Blei, 2017):

$$\eta^* = \frac{2S}{N}\frac{\dim(\theta)}{\mathrm{tr}(C)}, \text{ or } \qquad H^* = \frac{2S}{N}C^{-1}, \text{ (matrix learning rate)}$$

Can be used to tune learning rate using

$$C_\tau = (1-\kappa_\tau)C_{\tau-1} + \kappa_\tau \mathrm{cov}(\nabla\tilde{\mathcal{L}}).$$

Dropout Monte Carlo

Standard Network Model:

$$
\begin{aligned}
z_i &= \sigma_i \left(W_i x + b_i \right), \quad i = 1 : m-1, \\
y &= \sigma_2 \left(w_m z_m + b_m \right),
\end{aligned}
$$

Dropout Network Model:

$$
\begin{aligned}
z_i &= \sigma_i \left(W_i \left(\xi_i \circ x \right) + b_1 \right), \\
y &= \sigma_2 \left(w_m (\xi_m \circ z_m) + b_m \right)
\end{aligned}
$$

where ξ_i are vectors of zeros and ones sampled from Bernouli distribution.

- samples are drawn in each step of GD!
- Works also for other distributions of ξ
- Dropout is an approximate Bayesian sampler (Gal, Ghahramani, 2016),
  - dropout is switched on in prediction mode!!
  - prediction is repeated N times and averaged

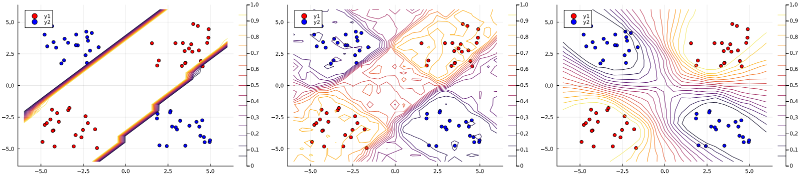
Deterministic:

model = Chain(Dense(2, 3, tanh), Dense(3, 2, tanh), Dense(2, 1, σ))

Dropout:

model = Chain(Dense(2, 10, tanh), Dropout(0.4), Dense(10, 2, tanh), Dropout(0.4), Dense(2, 1, σ))

Max. Likelihood | Dropout(100) | HMC



- Our default model for uncertainty modelling in NLP.

Take home message

- Inference of "best" parameters of your model is reliable only assymptotically
  - you need a LOT of data
- For insufficient data you face the epistemic uncertainty
  - you do not know what you do not know
  - Bayesian approach can handle that at additional cost
- Commodity solutions:
  - Probabilistic programming:
    - Turing.jl, PyMC3, STAN
    - For shallow Models
  - Sampling inside training of NN
    - Dropout MC
    - for deep models
- Nice theory
  - stochastic processes
  - kernel methods