TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Machine Learning Capstone project: Hanoi house price prediction

Instructor: Assoc. Prof. Than Quang Khoat

Members

- Nguyen Trung Hieu – 20204877
- Nguyen Lan Cuong – 20204872
- Tran Duc Tri – 20204893
- Nguyen Duc Thanh – 20203913
- Cu Duy Hiep - 20200212

Content

1. Introduction.
2. Data analysis and data preprocessing
3. Machine learning approaches
4. Experiment
5. Conclusion

1. Introduction

- Predicting the price of real estate is a very intriguing problem that attracts many scientists, economists, politicians trying to tackle them because of its importance in securing financial security, controlling the economy, and guaranteeing the social security.
- With the explosion of Artificial Intelligence, especially in the field of Machine Learning(ML), many people try to apply ML algorithms into solving this problem

1. Introduction

In this project, we will try to predict the house price in Hanoi using some of ML methods including:

- Random Forest
- Kernel Ridge Regression
- Gaussian Process
- Ensemble neural network

Content

1. Introduction.
2. Data analysis and data preprocessing
3. Machine learning approaches
4. Experiment
5. Conclusion

Data analysis

- Hanoi Housing Dataset 2020 is a raw dataset containing 82.5 thousand records with 12 variables: Date, Address, District, Ward, House type, Legal status, Number of floors, Number of bedrooms, Area, Length, Width, Price per square meters.

Ngày	Địa chỉ	Quận	Huyện	Loại hình nhà ở	Giấy tờ pháp lý	Số tầng	Số phòng ngủ	Diện tích	Dài	Rộng	Giá/m2
2020-08-05	Đường Hoàng Quốc	Quận Cầu Giấy	Phường Nghĩa Đô	Nhà ngõ, hẻm	Đã có sổ	4	5 phòng	46 m^2	NaN	NaN	86,96 triệu/m^2
2020-08-05	Đường Kim Giang, F	Quận Thanh Xuân	Phường Kim Giang	Nhà mặt phố, mặt tiề	NaN	NaN	3 phòng	37 m^2	NaN	NaN	116,22 triệu/m^2
2020-08-05	phố minh khai, Phườ	Quận Hai Bà Trưng	Phường Minh Khai	Nhà ngõ, hẻm	Đã có sổ	4	4 phòng	40 m^2	10 m	4 m	65 triệu/m^2
2020-08-05	Đường Võng Thị, Ph	Quận Tây Hồ	Phường Thụy Khuê	Nhà ngõ, hẻm	Đã có sổ	NaN	6 phòng	51 m^2	12.75 m	4 m	100 triệu/m^2
2020-08-05	Đường Kim Giang, F	Quận Thanh Xuân	Phường Kim Giang	Nhà ngõ, hẻm	NaN	NaN	4 phòng	36 m^2	9 m	4 m	86,11 triệu/m^2
2020-08-05	Đường Yên Hòa, Ph	Quận Cầu Giấy	Phường Yên Hoà	Nhà ngõ, hẻm	Đã có sổ	NaN	nhiều hơn 10 phòng	46 m^2	12.1 m	3.8 m	104,35 triệu/m^2
2020-08-05	Đường Tây Sơn, Ph	Quận Đống Đa	Phường Trung Liệt	Nhà ngõ, hẻm	NaN	NaN	3 phòng	52 m^2	NaN	4.5 m	112,5 triệu/m^2
2020-08-05	Đường Lò Đúc, Phư	Quận Hai Bà Trưng	Phường Đống Mác	Nhà mặt phố, mặt tiề	Đã có sổ	6	5 phòng	32 m^2	NaN	6.8 m	184,38 triệu/m^2
2020-08-05	Đường Xuân La, Phư	Quận Tây Hồ	Phường Xuân La	Nhà ngõ, hẻm	NaN	NaN	4 phòng	75 m^2	12 m	6.5 m	120 triệu/m^2
2020-08-05	Đường 19/5, Phườn	Quận Hà Đông	Phường Văn Quán	Nhà ngõ, hẻm	Đã có sổ	4	3 phòng	41 m^2	NaN	3.5 m	64,63 triệu/m^2
2020-08-05	Đường Tựu Liệt, Th	Huyện Thanh Trì	Thị trấn Văn Điển	Nhà ngõ, hẻm	Đã có sổ	NaN	3 phòng	35 m^2	NaN	NaN	45,71 triệu/m^2
2020-08-05	Đường Định Công H	Quận Hoàng Mai	Phường Định Công	Nhà ngõ, hẻm	Đã có sổ	5	4 phòng	30 m^2	NaN	NaN	83,33 triệu/m^2
2020-08-05	Đường Bồ Đề, Phườ	Quận Long Biên	Phường Bồ Đề	Nhà ngõ, hẻm	Đã có sổ	NaN	4 phòng	52 m^2	13 m	4 m	93,27 triệu/m^2

Data analysis

Remove outer district

> 8 floors

<30 and >300

Ngày	Địa chỉ	Quận	Huyện	Loại hình nhà ở	Giấy tờ pháp lý	Số tầng	Số phòng ngủ	Diện tích	Dài	Rộng	Giá/m2
2020-08-05	Đường Hoàng Quốc	Quận Cầu Giấy	Phường Nghĩa Đô	Nhà ngõ, hẻm	Đã có sổ	4	5 phòng	46 m²	NaN	NaN	86,96 triệu/m²
2020-08-05	Đường Kim Giang, F	Quận Thanh Xuân	Phường Kim Giang	Nhà mặt phố, mặt tiề	NaN	NaN	3 phòng	37 m²	NaN	NaN	116,22 triệu/m²
2020-08-05	phố minh khai, Phườ	Quận Hai Bà Trưng	Phường Minh Khai	Nhà ngõ, hẻm	Đã có sổ	4	4 phòng	40 m²	10 m	4 m	65 triệu/m²
2020-08-05	Đường Võng Thị, Ph	Quận Tây Hồ	Phường Thụy Khuê	Nhà ngõ, hẻm	Đã có sổ	NaN	6 phòng	51 m²	12.75 m	4 m	100 triệu/m²
2020-08-05	Đường Kim Giang, F	Quận Thanh Xuân	Phường Kim Giang	Nhà ngõ, hẻm	NaN	NaN	4 phòng	36 m²	9 m	4 m	86,11 triệu/m²
2020-08-05	Đường Yên Hòa, Ph	Quận Cầu Giấy	Phường Yên Hoà	Nhà ngõ, hẻm	Đã có sổ	NaN	nhiều hơn 10 phòng	46 m²	12.1 m	3.8 m	104,35 triệu/m²
2020-08-05	Đường Tây Sơn, Ph	Quận Đống Đa	Phường Trung Liệt	Nhà ngõ, hẻm	NaN	NaN	3 phòng	52 m²	NaN	4.5 m	112,5 triệu/m²
2020-08-05	Đường Lò Đúc, Phư	Quận Hai Bà Trưng	Phường Đống Mác	Nhà mặt phố, mặt tiề	Đã có sổ	6	5 phòng	32 m²	NaN	6.8 m	184,38 triệu/m²
2020-08-05	Đường Xuân La, Ph	Quận Tây Hồ	Phường Xuân La	Nhà ngõ, hẻm	NaN	NaN	4 phòng	75 m²	12 m	6.5 m	120 triệu/m²
2020-08-05	Đường 19/5, Phườn	Quận Hà Đông	Phường Văn Quán	Nhà ngõ, hẻm	Đã có sổ	4	3 phòng	41 m²	NaN	3.5 m	64,63 triệu/m²
2020-08-05	Đường Tựu Liệt, Th	Huyện Thanh Trì	Thị trấn Văn Điển	Nhà ngõ, hẻm	Đã có sổ	NaN	3 phòng	35 m²	NaN	NaN	45,71 triệu/m²
2020-08-05	Đường Định Công H	Quận Hoàng Mai	Phường Định Công	Nhà ngõ, hẻm	Đã có sổ	5	4 phòng	30 m²	NaN	NaN	83,33 triệu/m²
2020-08-05	Đường Bồ Đề, Phườ	Quận Long Biên	Phường Bồ Đề	Nhà ngõ, hẻm	Đã có sổ	NaN	4 phòng	52 m²	13 m	4 m	93,27 triệu/m²

Remove record has null value

Data preprocessing

- *With ward data, we use Binary encoding to process.*
- Transform categorical value to ordinal value with is the number of unique wards.
- Transform the ordinal value to binary, which is a sequence of 1 and 0 lengths.
- *With house type data and legal status data, we will use ordinal encoding for 2 columns: house types and legal status*

Data preprocessing

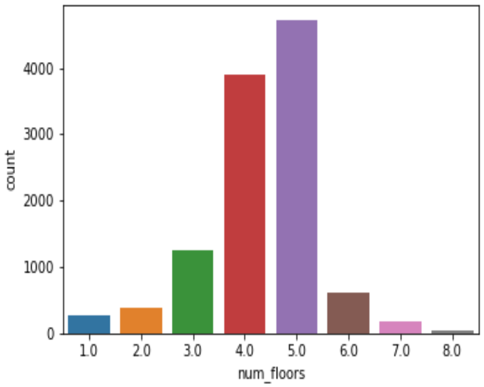


Fig 1: Number of floors count plot

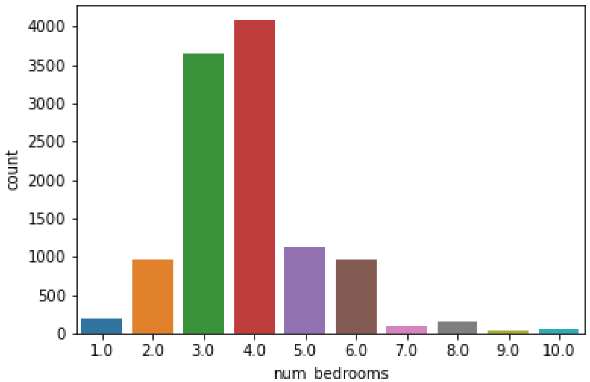


Fig 2: Number of bedrooms count plot

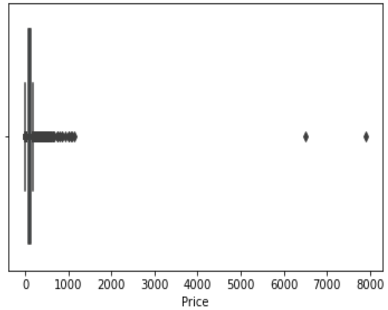


Fig 3: Boxplot of Price

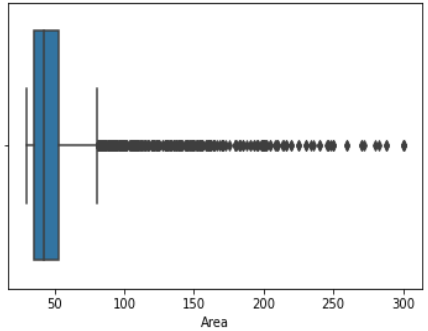


Fig 4: Boxplot of Area

Data preprocessing

IQR outlier detection:

We see that the Price variable have some extreme outliers, so we need to remove outliers of price. We run IQR outlier detection on price column by removing the value x that:

- $x < Q_1 - 1.5 * IQR \; or \; x > Q_3 + 1.5 * IQR$

where Q_1: 25^{th} percentiles, Q_3: 75^{th} percentiles, $IQR = Q_3 - Q_1$.

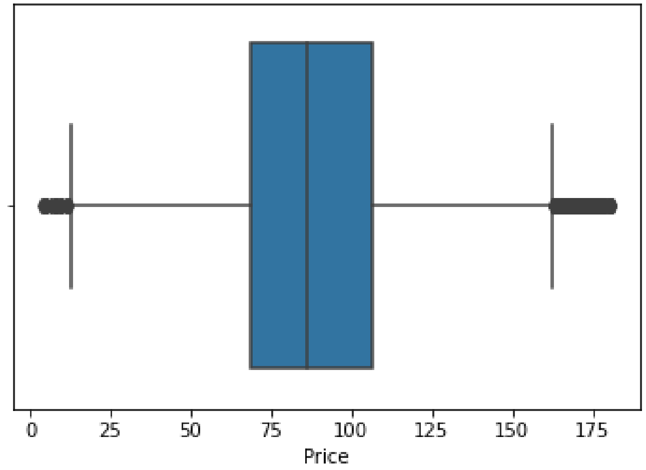


Fig 5: Boxplot on Price after running IQR

Content

1. Introduction.
2. Data analysis and data preprocessing
3. Machine learning approaches
4. Experiment
5. Conclusion

3.1. Random forest regressor

3.1.1. Decision tree regressor

- CART (Classification and Regression Trees) algorithm to handle both numeric and categorical data. The idea is choosing an attribute at each sub-tree to split into left and right branch
- For our house price problem, we use the loss function:

$$H(j_i, t_i) = \frac{\left|D_i^L(j,t)\right|}{|D_i|} l\left(D_i^L(j,t)\right) + \frac{\left|D_i^R(j,t)\right|}{|D_i|} l\left(D_i^R(j,t)\right)$$

$$l(D) = \frac{1}{|D|} \sum_{i \in D} (y_i - \bar{y})^2$$

3.1. Random forest regressor

3.1.2. Random forest regressor

We grow K decision trees when learning random forests:

- For each tree, we generate a dataset to train by sampling with replacement from .
- When we grow each individual tree, when choosing attributes and threshold to split a node, we only consider from a subset of attributes.
- We can limit the depth of each tree with a hyperparameter.

The final output is the average results obtained from those trees

3.2. Kernel Ridge Regression

First, we recall the method of ridge regression:

- Suppose we have the set of inputs {(**X**$_i$, y$_i$)}. We need to minimize :

$$L = \sum_i (\boldsymbol{w}^T \mathbf{X}_i - y_i)^2 + \lambda \boldsymbol{w}^T \cdot \boldsymbol{w}$$

$$\frac{\partial L}{\partial \boldsymbol{w}} = \sum_i 2\ \mathbf{X}_i \left(\boldsymbol{w}^T \boldsymbol{X}_i - y_i \right) + \mathbf{2}\lambda \boldsymbol{w}$$

3.2. Kernel Ridge Regression

- We can express w as a linear combination of all input vectors (using induction hypothesis):

$$\boldsymbol{w} = \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{n}} \alpha_i \ \boldsymbol{X_i}$$

- We also have the solution of the ridge regression:

$$\boldsymbol{w} = (\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}} + \lambda\mathbf{I})^{-\mathbf{1}}\boldsymbol{X}\boldsymbol{y}^{\boldsymbol{T}}$$

- From two equations above, we can write

$$\alpha = (\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X} + \ \beta^2\mathbf{I})^{-\mathbf{1}}\boldsymbol{y}^{\boldsymbol{T}}$$

3.2. Kernel Ridge Regression

Now, we can let $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{K}$ and replace K by some kernel matrix such as :

- Linear kernel: $\boldsymbol{K}(x_1, x_2) = x_1^T x_2$
- Laplacian kernel: $\boldsymbol{K}(x_1, x_2) = \exp\left(-\gamma\left|\left|x_1 - x_2\right|\right|_1\right)$
- Polynomial kernel: $\boldsymbol{K}(x_1, x_2) = (\gamma x_1^T x_2 + 1)^3$
- Radial basis function kernel: $\boldsymbol{K}(x_1, x_2) = exp(-\gamma\left|\left|x_1 - x_2\right|\right|_2^2)$

3.3. Gaussian Process

In general, the posterior predictive distribution is given by the following formula:

$$P(Y\,|D,X) = \int_w P(Y,\mathbf{w}\,|D,X)\,\boldsymbol{d}\mathbf{w} = \int_w P(Y|\mathbf{w},D,X)\boldsymbol{P}(\mathbf{w}|D)\boldsymbol{d}\mathbf{w}$$

- Ridge regression : Gaussian prior and likelihood, MAP
- Gaussian process: Learn all the function

3.3. Gaussian Process

3.3.1. Gaussian Process definition

- We can define the mean function m(x) and their covariance matrix k(x, x') of a real process f(x) as:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}) + m(\mathbf{x}))]$$

- Without loss of generality, we usually take the mean function to be zero.
- In detail, if we choose several input points, **we can generate a random Gaussian vector with covariance matrix**

$$\mathbf{f}_* \sim N\left(0, K(\boldsymbol{X}_*, \boldsymbol{X}_*)\right)$$

3.3. Gaussian Process

3.3.2. Gaussian Process prediction

- Prediction for noise – free observations

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim N(0, \begin{bmatrix} K(\boldsymbol{X}, \boldsymbol{X}) & K(\boldsymbol{X}, \boldsymbol{X}_*) \\ K(\boldsymbol{X}_*, \boldsymbol{X}) & K(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix})$$

The distribution of conditioning on the observations can be written as follows**:**

$$\mathbf{f}_* | \boldsymbol{X}_*, \boldsymbol{X}, \mathbf{f} \sim N(K(\boldsymbol{X}_*, \boldsymbol{X})K(\boldsymbol{X}, \boldsymbol{X})^{-1}\mathbf{f}, K(\boldsymbol{X}_*, \boldsymbol{X}_*) - K(\boldsymbol{X}_*, \boldsymbol{X})K(\boldsymbol{X}, \boldsymbol{X})^{-1} K(\boldsymbol{X}, \boldsymbol{X}_*))$$

3.3. Gaussian Process

3.3.2. Gaussian Process prediction

- Prediction of noisy observation

$$y = f(\mathbf{x}) + \varepsilon \qquad cov(y) = K(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\mathbf{I}$$

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix} \sim N(0, \begin{bmatrix} K(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\mathbf{I} & K(\boldsymbol{X},\boldsymbol{X}_*) \\ K(\boldsymbol{X}_*,\boldsymbol{X}) & K(\boldsymbol{X}_*,\boldsymbol{X}_*)\end{bmatrix})$$

We derive the mean and variance of $\mathbf{f}_*$

$$\mathbb{E}[\mathbf{f}_*] = \boldsymbol{k}_*{}^{\boldsymbol{T}}(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\mathbf{I})^{-1}\boldsymbol{y}$$

$$\mathbb{V}[\mathbf{f}_*] = \boldsymbol{k}(\mathbf{x}_*,\mathbf{x}_*) - \boldsymbol{k}_*{}^{\boldsymbol{T}}(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\mathbf{I})^{-1}\boldsymbol{k}_*$$

3.3. Gaussian Process

3.3.2. Gaussian Process prediction

- Cholesky factorization:

When **A** is a real symmetric positive-definite matrix, we can decompose it as

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T$$

Where **L** is a real lower triangular matrix with positive diagonal entries.

- Algorithm for Gaussian Process regression

Input: X **(inputs),** y **(targets),** k **(covariance function),** σ^2 **(noise level),** $\mathbf{x}_*$**(test input)**

1. $L := cholesky((K(X, X) + \sigma^2 \mathbf{I}))$
2. $\alpha := L^T \backslash (L \backslash y)$
3. $\mathbb{E}[\mathbf{f}_*] := k_*^T \alpha$
4. $\mathbf{v} := L \backslash k_*$
5. $\mathbb{V}[\mathbf{f}_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^T \mathbf{v}$
6. $\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^T \alpha - \sum_i \log L_{ij} - \frac{n}{2}\log 2\pi$
7. **Return** $\mathbb{E}[\mathbf{f}_*]$(mean), $\mathbb{V}[\mathbf{f}_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood).

Where $\log p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|X)\, d\mathbf{f}$

3.3. Gaussian Process

3.3.3. The covariance function

- There are three main ways to choose a good covariance function:

✓ Expert knowledge (awesome to have, difficult to get)

✓ Bayesian model selection (more possibly to face intractable integrals)

✓ Cross-validation (time consuming but easy to implement)

- We can choose the covariance matrix as a kernel matrix since there are many similar properties

3.3. Gaussian Process

3.3.3. The covariance function

- RBF kernel: $$k(x_i, x_j) = exp(-\frac{d(x_i, x_j)^2}{2l^2})$$

- Dot product and white kernel:

$$k(x_i, x_j) = \sigma^2 + x_i \cdot x_j$$

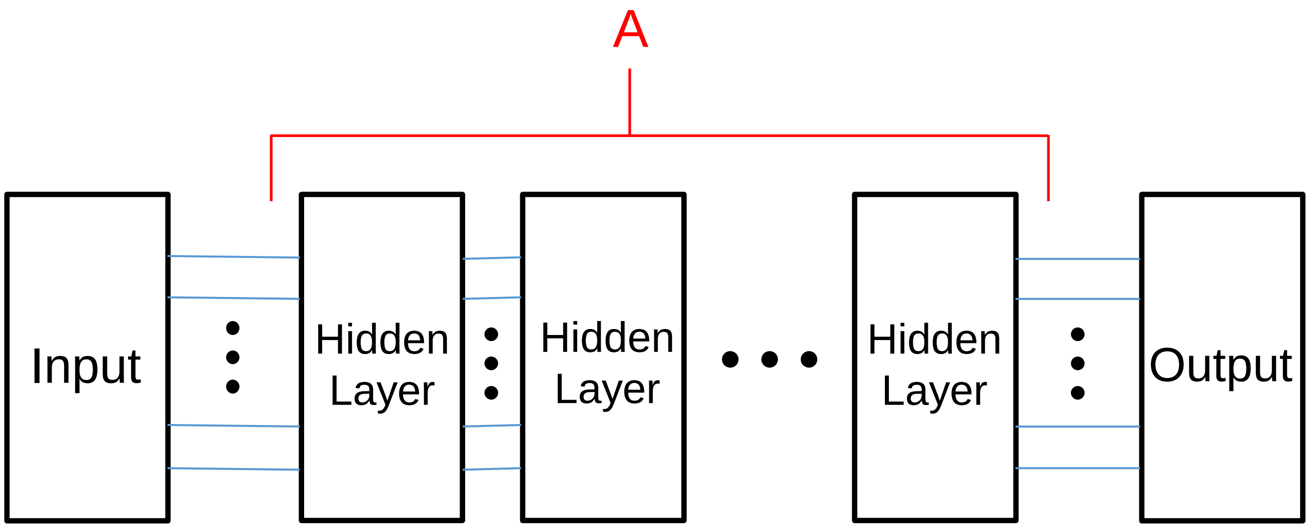
  - Dot product kernels:
  - White kernels:

$$k(x_i, x_j) = \begin{cases} noiselevel & if\ x_i == x_j \\ 0 & otherwise \end{cases}$$

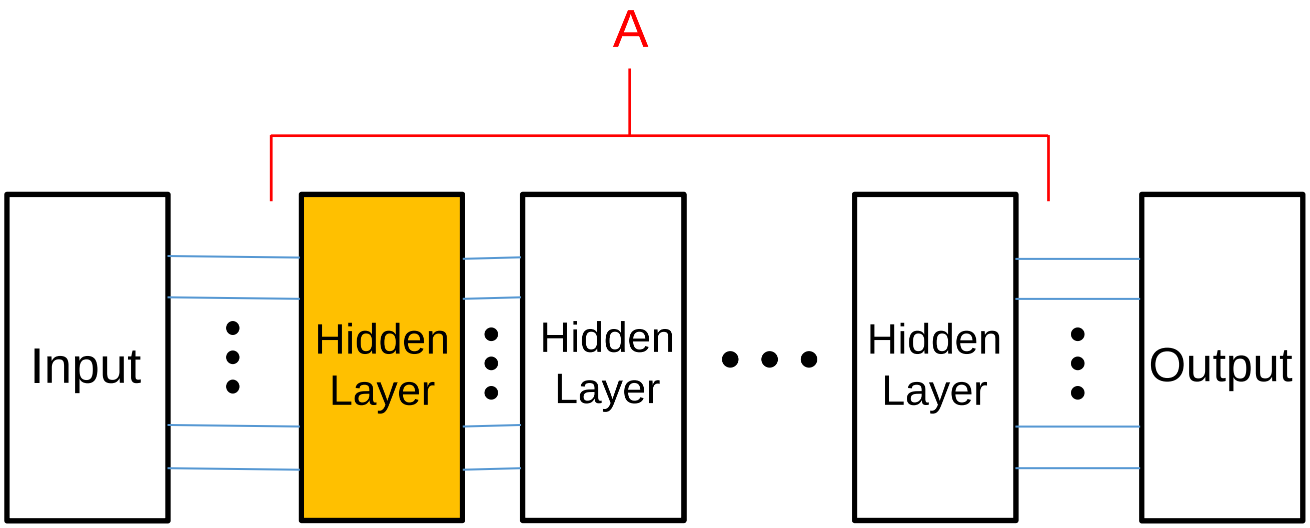
- Matérn kernel: $$k(x_i, x_j) = \frac{1}{\Gamma(\upsilon)2^{\upsilon-1}}\left(\frac{\sqrt{2\upsilon}}{l}d(x_i, x_j)\right)^{\upsilon} K_{\upsilon}(\frac{\sqrt{2\upsilon}}{l}d(x_i, x_j))$$

- Rational Quadratic kernel $$k(x_i, x_j) = (1 + \frac{d(x_i, x_j)^2}{2\alpha l^2})^{-\alpha}$$

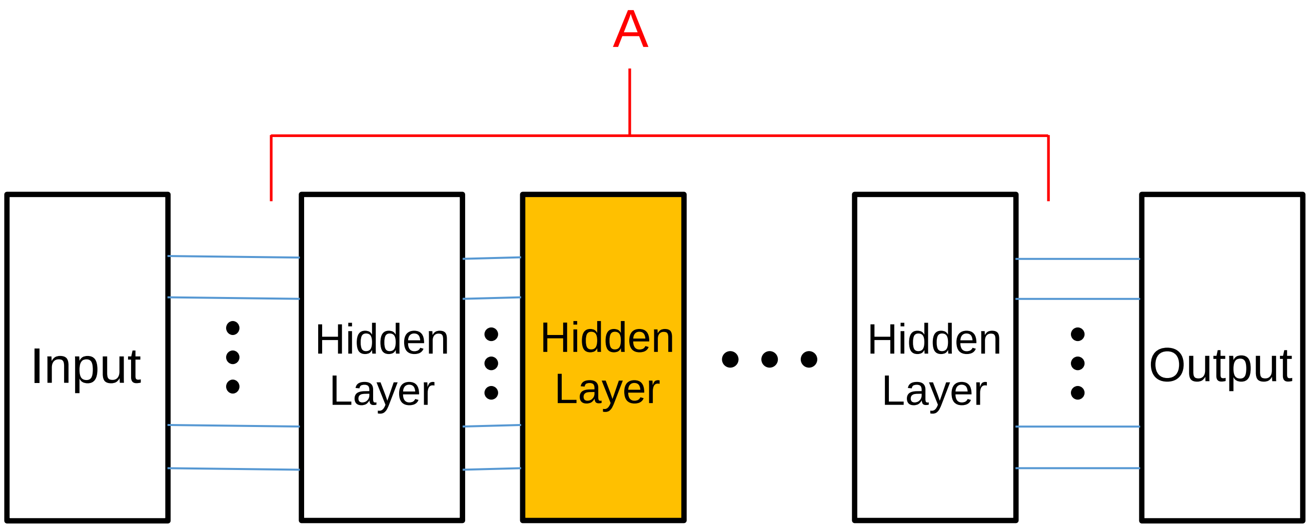
3.4. Ensemble neural network



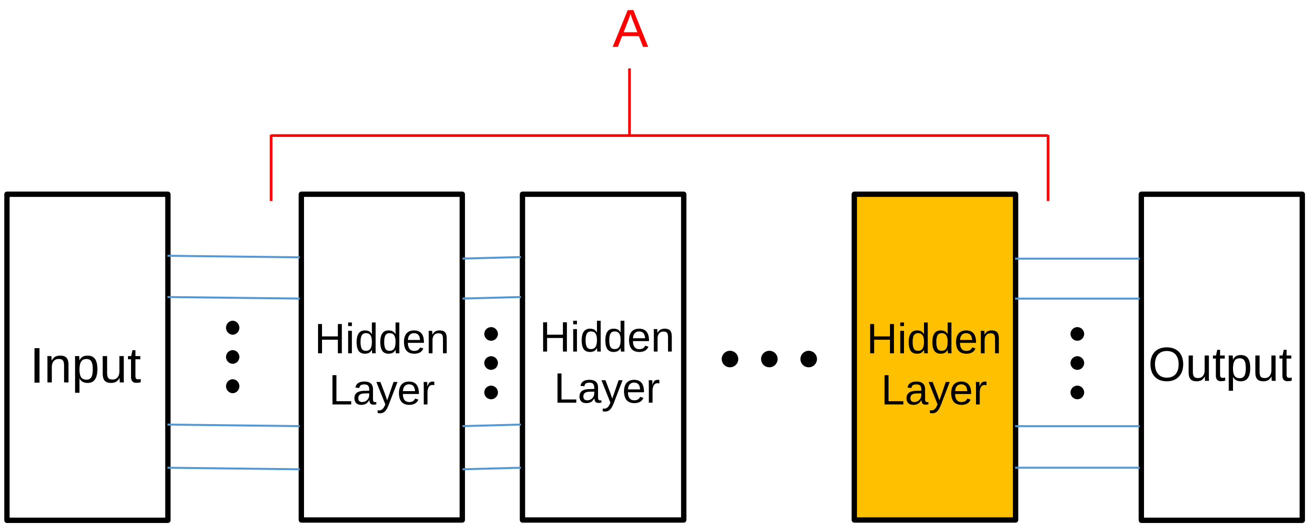
3.4. Ensemble neural network



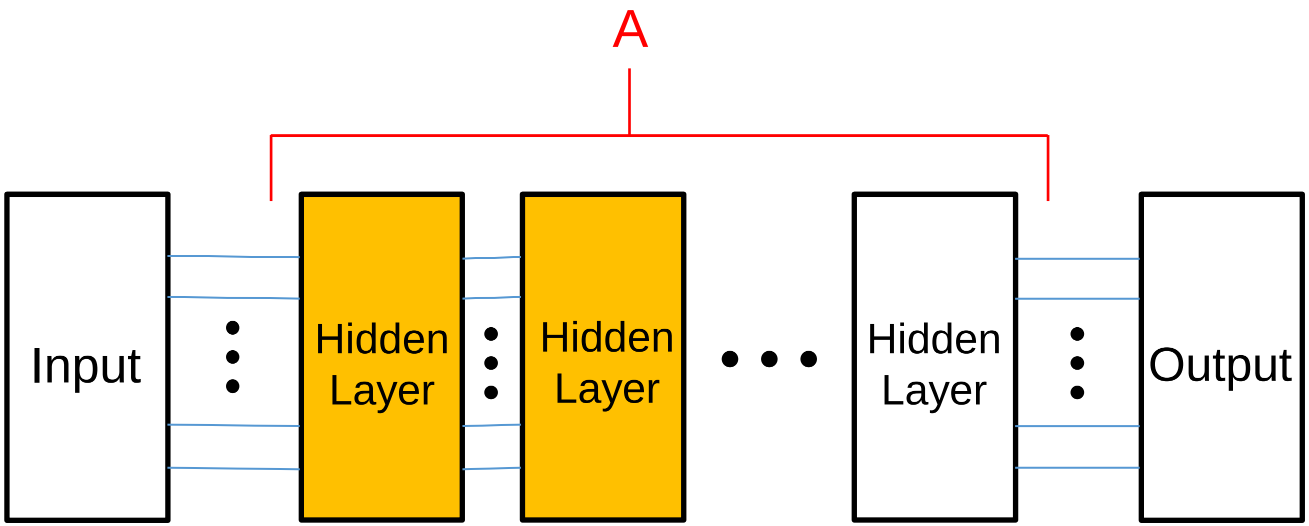
3.4. Ensemble neural network



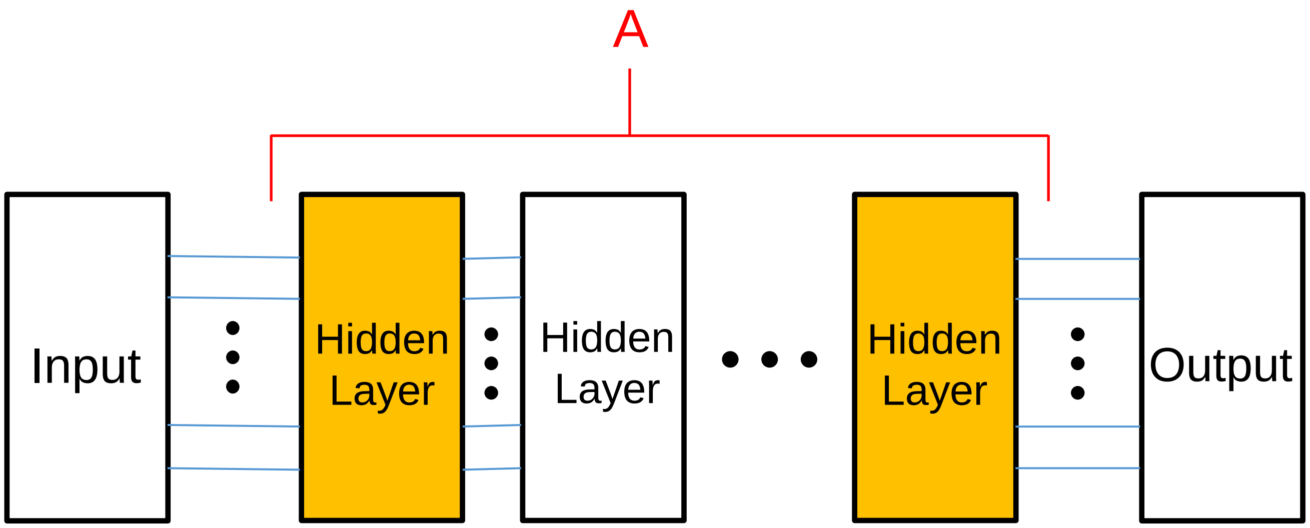
3.4. Ensemble neural network



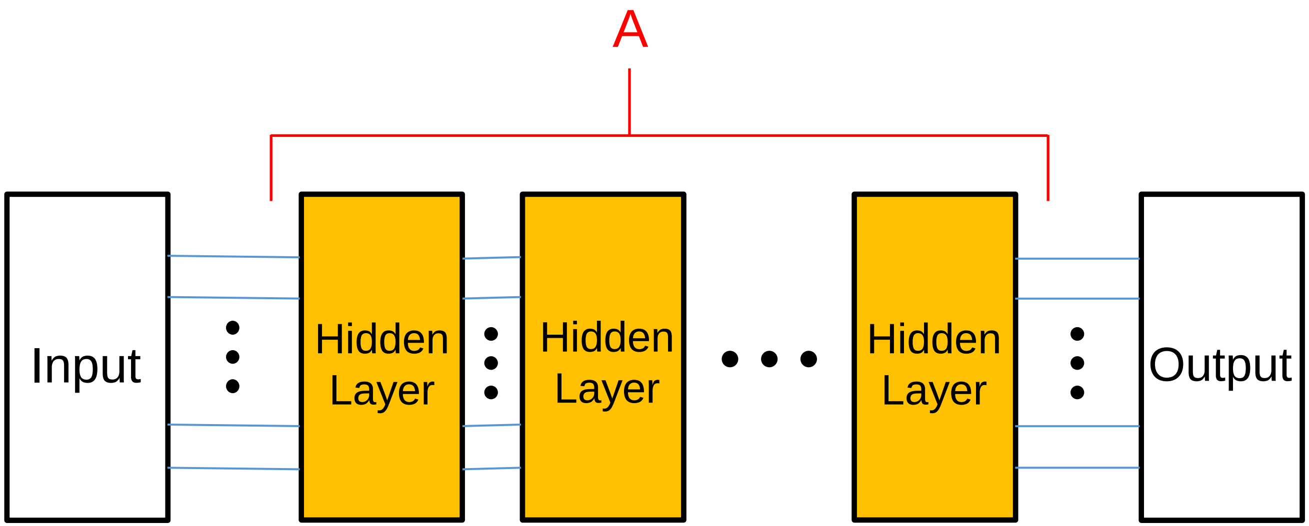
3.4. Ensemble neural network



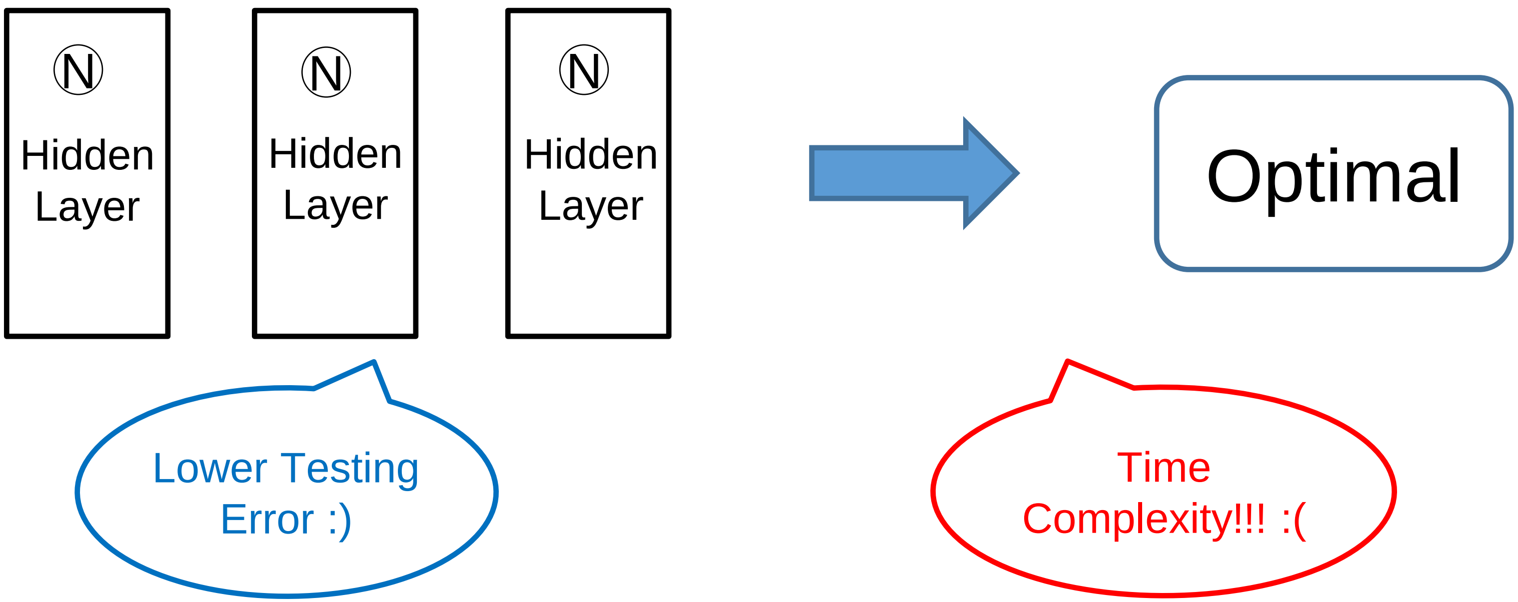
3.4. Ensemble neural network



3.4. Ensemble neural network



3.4. Ensemble neural network



Ⓝ: Neuron

3.4. Ensemble neural network

- Adam optimization

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Source: Diederik Kingma, Jimmy Ba **Adam: A Method for Stochastic Optimization (2014)** arXiv:1412.6980

Content

1. Introduction.
2. Data analysis and data preprocessing
3. Machine learning approaches
4. Experiment
5. Conclusion

4. Experiments

Experiments setup

- We split the data into 2 parts: training data and testing data, with ratio 8:2
- We use 4 folds cross – validation on the training data set
- We use 3 different metrics for model assessment:

+ Mean absolute percentage error (MAPE).
+ Mean absolute error (MAE)
+ Root mean squared error (MSE)

4. Experiments

Random forest regressor model

	N = 50	N = 100	N = 150	N = 200	N = 250	N = 300	N = 350	N = 400
MAPE	0.277	0.276	0.276	0.276	0.276	0.276	0.276	**0.276**
RMSE	25.513	25.461	25.440	25.427	25.429	25.424	25.426	**25.422**
MAE	19.014	18.976	18.959	18.953	18.960	18.952	18.951	**18.949**

Table 1: Random Forest performance on different number of trees

4. Experiments

Random forest regressor model

	d = 5	d = 8	d = 11	d = 14	d = 17	d = 20	d = 23	d = 26	d = inf
MAPE	0.309	0.290	0.274	0.268	0.267	**0.267**	0.267	0.267	0.267
RMSE	27.530	26.530	25.280	24.904	24.864	**24.854**	24.856	24.865	24.865
MAE	21.003	19.846	18.874	18.846	18.389	**18.379**	18.380	18.385	18.384

Table 2: Random Forest performance on different maximum depth of each tree

4. Experiments

Kernel ridge regression

	Linear	Laplacian	RBF	Polynomial
MAPE	0.307	**0.268**	0.276	0.279
RMSE	28.12	**26.08**	26.48	26.38
MAE	28.12	**19.31**	19.48	19.61

Table 3: Kernel ridge performance on different kernels

	0.001	0.025	0.05	0.1	0.25	0.5	1.0	2.0
MAPE	0.298	0.281	**0.281**	0.281	0.282	0.283	0.284	0.286
RMSE	29.13	26.59	**26.34**	26.35	26.35	26.35	26.42	26.55
MAE	20.93	19.73	**19.69**	19.70	19.73	19.82	19.91	20.02

Table 4: Kernel ridge performance on different alpha

4. Experiments

Gaussian process

	Rational Quadratic	DotProduct	RBF	Matérn
MAPE	**0.269**	0.306	0.284	0.279
RMSE	**26.03**	27.84	26.53	26.30
MAE	**19.16**	21.16	19.82	19.42

Table 5: Gaussian process performance on different kernels

4. Experiments

Ensemble neural network

	n = 5	n = 10	n = 15	n = 20	n = 25	n = 30
MAPE	0.260	0.245	0.241	0.233	**0.231**	0.236
RMSE	27.52	26.29	26.39	25.44	**25.27**	25.87
MAE	20.84	19.66	19.61	18.77	**18.53**	19.10

Table 6: Ensemble neural network on different number of neurons

Model training and comparison

	Random forest		Ridge Kernel		Gaussian process		Ensemble NN	
	Train	**Test**	Train	Test	**Train**	Test	Train	Test
MAPE	0.196	**0.225**	0.216	0.233	**0.122**	0.233	0.227	0.227
RMSE	18.37	**24.32**	20.46	25.64	**11.73**	25.43	22.27	25.23
MAE	13.58	**17.67**	17.16	18.77	**8.62**	18.42	16.43	18.44

Table 7: Different model performance on train – test set

Content

1. Introduction.
2. Data analysis and data preprocessing
3. Machine learning approaches
4. Experiment
5. Conclusion

THE END!