Machine Learning

Jochen Bröcker

University of Reading, UK

May 11, 2023

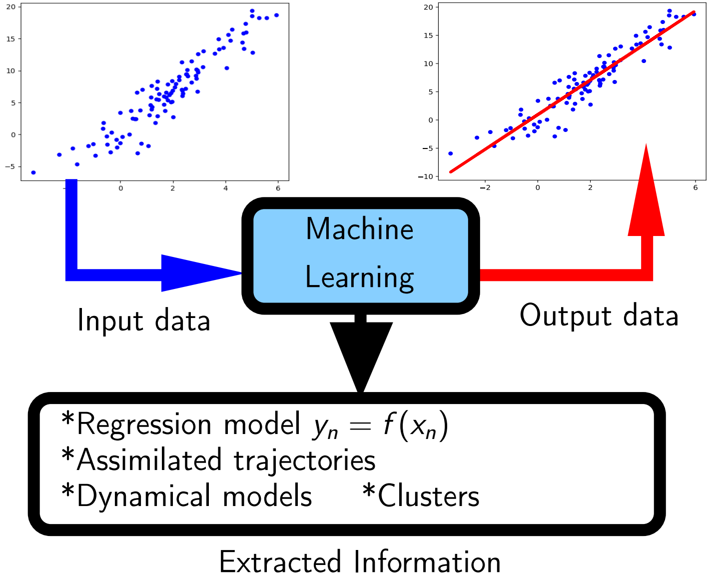
Contents

Contents

Contents

Contents

Problem of machine learning



Problem of machine learning

- Data tells a story
- Information or "gist" of story extracted
- Extracted information is used to re–tell the story
- Errors in re–telling may be used to revise extracted information

Ultimate Goal:
Be able to predict behaviour of unseen data, or "how does the story continue".

Examples of machine learning problems:

- Time series models
- Data assimilation
- Unsupervised learning
- Regression and classification

Problem of machine learning

- Data tells a story
- Information or “gist” of story extracted
- Extracted information is used to re–tell the story
- Errors in re–telling may be used to revise extracted information

Ultimate Goal:
Be able to predict behaviour of unseen data, or “how does the story continue”.

Examples of machine learning problems:

- Time series models
- Data assimilation
- Unsupervised learning
- Regression and classification

Problem of machine learning

- Data tells a story
- Information or "gist" of story extracted
- Extracted information is used to re–tell the story
- Errors in re–telling may be used to revise extracted information

Ultimate Goal:

Be able to predict behaviour of unseen data, or "how does the story continue".

Examples of machine learning problems:

- Time series models
- Data assimilation
- Unsupervised learning
- Regression and classification

Problem of machine learning

- Data tells a story
- Information or "gist" of story extracted
- Extracted information is used to re–tell the story
- Errors in re–telling may be used to revise extracted information

Ultimate Goal:

Be able to predict behaviour of unseen data, or "how does the story continue".

Examples of machine learning problems:

- Time series models
- Data assimilation
- Unsupervised learning
- Regression and classification

Problem of machine learning

- Data tells a story
- Information or "gist" of story extracted
- Extracted information is used to re–tell the story
- Errors in re–telling may be used to revise extracted information

Ultimate Goal:

Be able to predict behaviour of unseen data, or "how does the story continue".

Examples of machine learning problems:

- Time series models
- Data assimilation
- Unsupervised learning
- Regression and classification

Problem of machine learning

- Data tells a story
- Information or “gist” of story extracted
- Extracted information is used to re–tell the story
- Errors in re–telling may be used to revise extracted information

Ultimate Goal:

Be able to predict behaviour of unseen data, or “how does the story continue”.

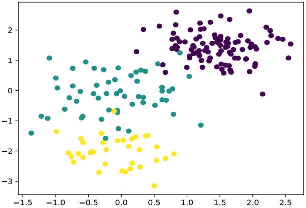
Examples of machine learning problems:

- Time series models
- Data assimilation
- Unsupervised learning
- Regression and classification

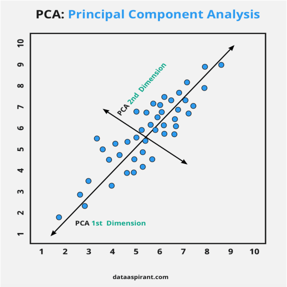
Examples for unsupervised learning methods

...apply to data set $D = \{\mathbf{x}_n \in F, n = 1, 2, \ldots\}$, where F is potentially very high dimensional.

Clustering Group data into representative "clusters". Cluster centres represent points in the cluster



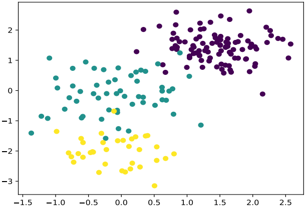
Principal Component Analysis Find principal axes of minimal ellipsoid encompassing the data. Then chose subspace spanned by axes with large projection, delete remaining axes.



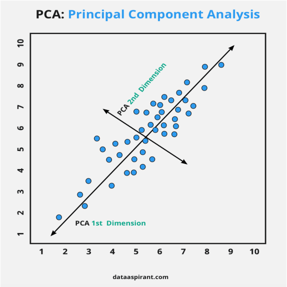
Examples for unsupervised learning methods

. . . apply to data set $D = \{\mathbf{x}_n \in F, n = 1, 2, \ldots\}$, where F is potentially very high dimensional.

Clustering Group data into representative "clusters". Cluster centres represent points in the cluster

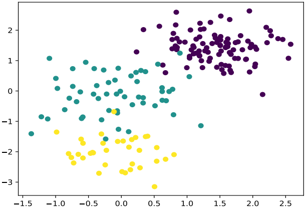


Principal Component Analysis Find principal axes of minimal ellipsoid encompassing the data. Then chose subspace spanned by axes with large projection, delete remaining axes.

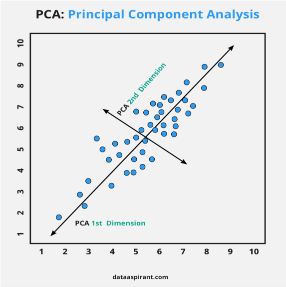


Examples for unsupervised learning methods

...apply to data set $D = \{\mathbf{x}_n \in F, n = 1, 2, \ldots\}$, where F is potentially very high dimensional.



Clustering Group data into representative "clusters". Cluster centres represent points in the cluster



Principal Component Analysis Find principal axes of minimal ellipsoid encompassing the data. Then chose subspace spanned by axes with large projection, delete remaining axes.

General framework for unsupervised learning methods

Given data points $\mathbf{x}_1, \mathbf{x}_2, \ldots$ in "large" (or high dimensional) space F, find a "small" (or low dimensional) subset $F_0 \subset F$ and a map

$$f : F \to F_0 \subset F$$

which "approximates the identity", i.e.

$$r_N = \sum_{n=1}^{N} d(x_n, f(x_n))$$

is small (and d is an appropriate measure of distance).

Trade–Off

A larger F_0 gives a smaller error r_N, but implies a higher complexity of f.

General framework for unsupervised learning methods

Given data points $\mathbf{x}_1, \mathbf{x}_2, \ldots$ in "large" (or high dimensional) space F, find a "small" (or low dimensional) subset $F_0 \subset F$ and a map

$$f : F \to F_0 \subset F$$

which "approximates the identity", i.e.

$$r_N = \sum_{n=1}^{N} d(x_n, f(x_n))$$

is small (and d is an appropriate measure of distance).

Trade–Off

A larger F_0 gives a smaller error r_N, but implies a higher complexity of f.

General framework for unsupervised learning methods

Given data points $\mathbf{x}_1, \mathbf{x}_2, \dots$ in "large" (or high dimensional) space F, find a "small" (or low dimensional) subset $F_0 \subset F$ and a map

$$f : F \to F_0 \subset F$$

which "approximates the identity", i.e.

$$r_N = \sum_{n=1}^{N} d(x_n, f(x_n))$$

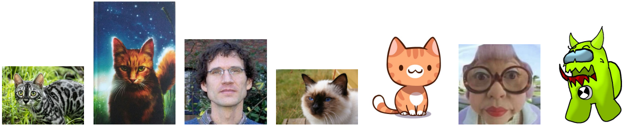
is small (and d is an appropriate measure of distance).

Trade–Off

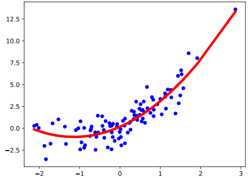
A larger F_0 gives a smaller error r_N, but implies a higher complexity of f.

Examples for regression and classification

Classification: Identify all pictures with cats (or tumors, or . . .)



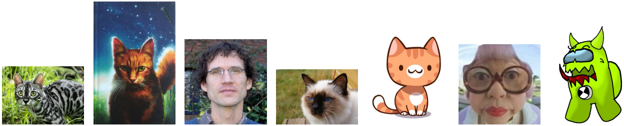
Regression: Identify functional relationship



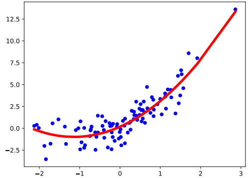
Multilabel regression, probabilistic regression, . . .

Examples for regression and classification

Classification: Identify all pictures with cats (or tumors, or . . .)



Regression: Identify functional relationship



Multilabel regression, probabilistic regression, . . .

The main ingredients of regression

and classification

- Two spaces F, G with *feature space* F potentially very large and *target space* G very small (i.e. $\mathbb{R}$ or finite set);
- a *training data* set T of *feature value pairs* $(\mathrm{x}_n, y_n), n = 1, \ldots, N$ with *features* $\mathrm{x}_n \in F$ and *targets* $y_n \in G$;
- a *model class* $\mathcal{F}$ of functions $f : F \to G$;
- a *loss function* $L : G \times G \to \mathbb{R}_{\geq 0}$ with the property that $L(y, y) = 0$ for all $y \in G$;
- a measure of complexity $\kappa : \mathcal{F} \to \mathbb{R}_{\geq 0}$

The value $L(y, f(\mathrm{x}))$ measures the error of the function $f \in \mathcal{F}$ in mapping the feature x onto the target y.
The value $\kappa(f)$ measures the "complexity" (i.e. irregularity, number of parameters) of the function $f \in \mathcal{F}$.

The main ingredients of regression

and classification

- Two spaces F, G with *feature space* F potentially very large and *target space* G very small (i.e. $\mathbb{R}$ or finite set);
- a *training data* set T of *feature value pairs* $(\mathsf{x}_n, y_n), n = 1, \dots, N$ with *features* $\mathsf{x}_n \in F$ and *targets* $y_n \in G$;
- a *model class* $\mathcal{F}$ of functions $f : F \to G$;
- a *loss function* $L : G \times G \to \mathbb{R}_{\geq 0}$ with the property that $L(y, y) = 0$ for all $y \in G$;
- a measure of complexity $\kappa : \mathcal{F} \to \mathbb{R}_{\geq 0}$

The value $L(y, f(\mathsf{x}))$ measures the error of the function $f \in \mathcal{F}$ in mapping the feature x onto the target y.
The value $\kappa(f)$ measures the "complexity" (i.e. irregularity, number of parameters) of the function $f \in \mathcal{F}$.

The main ingredients of regression

and classification

- Two spaces F, G with *feature space* F potentially very large and *target space* G very small (i.e. $\mathbb{R}$ or finite set);
- a *training data* set T of *feature value pairs* $(\mathsf{x}_n, y_n), n = 1, \ldots, N$ with *features* $\mathsf{x}_n \in F$ and *targets* $y_n \in G$;
- a *model class* $\mathcal{F}$ of functions $f : F \to G$;
- a *loss function* $L : G \times G \to \mathbb{R}_{\geq 0}$ with the property that $L(y, y) = 0$ for all $y \in G$;
- a measure of complexity $\kappa : \mathcal{F} \to \mathbb{R}_{\geq 0}$

The value $L(y, f(\mathsf{x}))$ measures the error of the function $f \in \mathcal{F}$ in mapping the feature x onto the target y.
The value $\kappa(f)$ measures the "complexity" (i.e. irregularity, number of parameters) of the function $f \in \mathcal{F}$.

The main ingredients of regression

and classification

- Two spaces F, G with *feature space* F potentially very large and *target space* G very small (i.e. $\mathbb{R}$ or finite set);
- a *training data* set T of *feature value pairs* $(\mathsf{x}_n, y_n), n = 1, \ldots, N$ with *features* $\mathsf{x}_n \in F$ and *targets* $y_n \in G$;
- a *model class* $\mathcal{F}$ of functions $f : F \to G$;
- a *loss function* $L : G \times G \to \mathbb{R}_{\geq 0}$ with the property that $L(y, y) = 0$ for all $y \in G$;
- a measure of complexity $\kappa : \mathcal{F} \to \mathbb{R}_{\geq 0}$

The value $L(y, f(\mathsf{x}))$ measures the error of the function $f \in \mathcal{F}$ in mapping the feature x onto the target y.
The value $\kappa(f)$ measures the "complexity" (i.e. irregularity, number of parameters) of the function $f \in \mathcal{F}$.

The main ingredients of regression
and classification

- Two spaces F, G with *feature space* F potentially very large and *target space* G very small (i.e. $\mathbb{R}$ or finite set);
- a *training data* set T of *feature value pairs* $(\mathsf{x}_n, y_n), n = 1, \ldots, N$ with *features* $\mathsf{x}_n \in F$ and *targets* $y_n \in G$;
- a *model class* $\mathcal{F}$ of functions $f : F \to G$;
- a *loss function* $L : G \times G \to \mathbb{R}_{\geq 0}$ with the property that $L(y, y) = 0$ for all $y \in G$;
- a measure of complexity $\kappa : \mathcal{F} \to \mathbb{R}_{\geq 0}$

The value $L(y, f(\mathsf{x}))$ measures the error of the function $f \in \mathcal{F}$ in mapping the feature x onto the target y.
The value $\kappa(f)$ measures the "complexity" (i.e. irregularity, number of parameters) of the function $f \in \mathcal{F}$.

The main ingredients of regression

and classification

- Two spaces F, G with *feature space* F potentially very large and *target space* G very small (i.e. $\mathbb{R}$ or finite set);
- a *training data* set T of *feature value pairs* $(\mathbf{x}_n, y_n), n = 1, \ldots, N$ with *features* $\mathbf{x}_n \in F$ and *targets* $y_n \in G$;
- a *model class* $\mathcal{F}$ of functions $f : F \to G$;
- a *loss function* $L : G \times G \to \mathbb{R}_{\geq 0}$ with the property that $L(y, y) = 0$ for all $y \in G$;
- a measure of complexity $\kappa : \mathcal{F} \to \mathbb{R}_{\geq 0}$

The value $L(y, f(\mathbf{x}))$ measures the error of the function $f \in \mathcal{F}$ in mapping the feature $\mathbf{x}$ onto the target y.

The value $\kappa(f)$ measures the "complexity" (i.e. irregularity, number of parameters) of the function $f \in \mathcal{F}$.

The main ingredients of regression
and classification

- Two spaces F, G with *feature space* F potentially very large and *target space* G very small (i.e. $\mathbb{R}$ or finite set);
- a *training data* set T of *feature value pairs* $(\mathbf{x}_n, y_n), n = 1, \ldots, N$ with *features* $\mathbf{x}_n \in F$ and *targets* $y_n \in G$;
- a *model class* $\mathcal{F}$ of functions $f : F \to G$;
- a *loss function* $L : G \times G \to \mathbb{R}_{\geq 0}$ with the property that $L(y, y) = 0$ for all $y \in G$;
- a measure of complexity $\kappa : \mathcal{F} \to \mathbb{R}_{\geq 0}$

The value $L(y, f(\mathbf{x}))$ measures the error of the function $f \in \mathcal{F}$ in mapping the feature $\mathbf{x}$ onto the target y.
The value $\kappa(f)$ measures the "complexity" (i.e. irregularity, number of parameters) of the function $f \in \mathcal{F}$.

The loss minimisation principle

Better: structural loss minimisation principle

Aim:
Find functional relationship $f \in \mathcal{F}$ between features and targets.

Loss minimisation principle:
Find $f_T \in \mathcal{F}$ by minimising *training error*

$$E_T := \frac{1}{N} \sum_{n=1}^{N} L(y_n, f(\mathrm{x}_n))$$

over $f \in \mathcal{F}$, subject to a constraint $\kappa(f) \leq c$.

Note: f_T depends on the training set T and also on c.

The loss minimisation principle

Better: structural loss minimisation principle

Aim:

Find functional relationship $f \in \mathcal{F}$ between features and targets.

Loss minimisation principle:

Find $f_T \in \mathcal{F}$ by minimising *training error*

$$E_T := \frac{1}{N} \sum_{n=1}^{N} L(y_n, f(\mathbf{x}_n))$$

over $f \in \mathcal{F}$, subject to a constraint $\kappa(f) \leq c$.

Note: f_T depends on the training set T and also on c.

Assessing performance

General Assumption:

- Feature–target pairs $\{(\mathbf{x}_n, y_n), n = 1, 2, \ldots\}$ are independent and identically distributed random variables
- $y_n = g(x_n) + r_n$ with r_n "noise"
- $L(y, \hat{y}) = (y - \hat{y})^2$ "Quadratic loss"

Test error:

is defined as

$$\mathrm{e}_{\mathrm{test}} := \mathbb{E}(y - f_T(\mathrm{x}))^2$$

where $\mathbb{E}$ is over T and a feature–target pair *not* in T.

Assessing performance

General Assumption:

- Feature–target pairs $\{(\mathbf{x}_n, y_n), n = 1, 2, \ldots\}$ are independent and identically distributed random variables
- $y_n = g(x_n) + r_n$ with r_n "noise"
- $L(y, \hat{y}) = (y - \hat{y})^2$ "Quadratic loss"

Test error:

is defined as

$$e_{\text{test}} := \mathbb{E}(y - f_T(\mathbf{x}))^2$$

where $\mathbb{E}$ is over T and a feature–target pair *not* in T.

Assessing performance

General Assumption:

- Feature–target pairs $\{(\mathbf{x}_n, y_n), n = 1, 2, \ldots\}$ are independent and identically distributed random variables
- $y_n = g(x_n) + r_n$ with r_n "noise"
- $L(y, \hat{y}) = (y - \hat{y})^2$ "Quadratic loss"

Test error:

is defined as

$$e_{\text{test}} := \mathbb{E}(y - f_T(\mathbf{x}))^2$$

where $\mathbb{E}$ is over T and a feature–target pair *not* in T.

Assessing performance

General Assumption:

- Feature–target pairs $\{(\mathbf{x}_n, y_n), n = 1, 2, \ldots\}$ are independent and identically distributed random variables
- $y_n = g(x_n) + r_n$ with r_n "noise"
- $L(y, \hat{y}) = (y - \hat{y})^2$ "Quadratic loss"

Test error:

is defined as

$$\mathbf{e}_{\text{test}} := \mathbb{E}(y - f_T(\mathbf{x}))^2$$

where $\mathbb{E}$ is over T and a feature–target pair *not* in T.

Bias–variance decomposition

Let $\bar{f}(\xi) = \mathbb{E}(f_T(\xi))$ the "average model" for each $\xi \in F$.
Remember $y = g(x) + r$.

$$e_{\text{test}} = \underbrace{\mathbb{E}r^2}_{\text{noise}} + \underbrace{\mathbb{E}(g(\mathrm{x}) - \bar{f}(\mathrm{x}))^2}_{\text{bias}} + \underbrace{\mathbb{E}(f_T(\mathrm{x}) - \bar{f}(\mathrm{x}))^2}_{\text{variance}}$$

Bias–variance decomposition

Let $\bar{f}(\xi) = \mathbb{E}(f_T(\xi))$ the "average model" for each $\xi \in F$.
Remember $y = g(x) + r$.

$$\mathbf{e}_{\text{test}} = \underbrace{\mathbb{E}r^2}_{\text{noise}} + \underbrace{\mathbb{E}(g(\mathbf{x}) - \bar{f}(\mathbf{x}))^2}_{\text{bias}} + \underbrace{\mathbb{E}(f_T(\mathbf{x}) - \bar{f}(\mathbf{x}))^2}_{\text{variance}}$$

Bias variance trade–off and model complexity

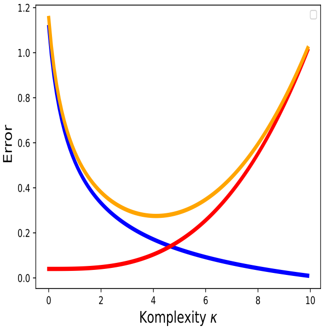
Demonstration later in context of linear models

Typical Bias–Variance Tradeoff

Bias (blue) decreases with k.

Variance (red) increases with k.

Test error (orange) exhibits minimum.



- The complexity κ controls the trade–off.
- *How do we estimate an appropriate value for κ?*
- The training error E_T is a *bad* estimator for the test error e_{test} (typically becomes better with κ due to overfitting).

Bias variance trade–off and model complexity

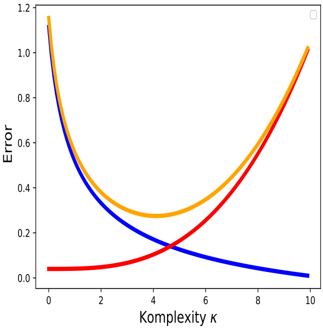
Demonstration later in context of linear models

Typical Bias–Variance Tradeoff

Bias (blue) decreases with k.

Variance (red) increases with k.

Test error (orange) exhibits minimum.



- The complexity κ controls the trade–off.
- *How do we estimate an appropriate value for κ?*
- The training error E_T is a *bad* estimator for the test error e_{test} (typically becomes better with κ due to overfitting).

Bias variance trade–off and model complexity

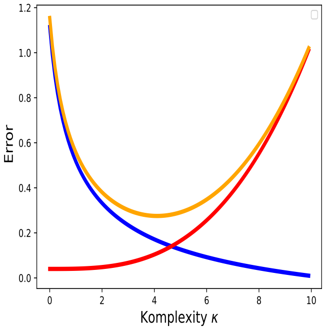
Demonstration later in context of linear models

Typical Bias–Variance Tradeoff

Bias decreases with k.

Variance increases with k.

Test error exhibits minimum.



- The complexity κ controls the trade–off.
- *How do we estimate an appropriate value for κ?*
- The training error E_T is a *bad* estimator for the test error e_{test} (typically becomes better with κ due to overfitting).

Bias variance trade–off and model complexity

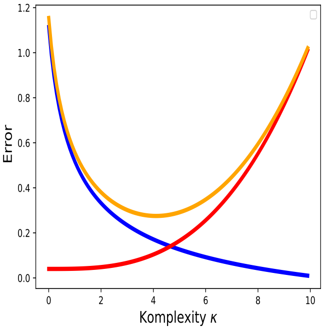
Demonstration later in context of linear models

Typical Bias–Variance Tradeoff

Bias (blue) decreases with k.

Variance (red) increases with k.

Test error (orange) exhibits minimum.



- The complexity κ controls the trade–off.
- *How do we estimate an appropriate value for κ?*
- The training error E_T is a *bad* estimator for the test error $\mathbf{e}_{\text{test}}$ (typically becomes better with κ due to overfitting).

Why are training and test error different?

Demonstration later in context of linear models

The training error E_T is a bad estimator for the test error $\mathbf{e}_{\text{test}}$.

$$\mathbf{e}_{\text{test}} = \mathbb{E}(y - f_T(\mathsf{x}))^2 \qquad (\mathsf{x}, y) \textit{ independent} \text{ from } T,$$

$$E_T = \frac{1}{N}\sum_{n=1}^{N}(y_n - f_T(\mathsf{x}_n))^2$$

$$\cong \mathbb{E}(y - f_T(\mathsf{x}))^2 \qquad (\mathsf{x}, y) \textit{ contained} \text{ in } T.$$

Why are training and test error different?

Demonstration later in context of linear models

The training error E_T is a bad estimator for the test error $\mathbf{e}_{\text{test}}$.

$$\mathbf{e}_{\text{test}} = \mathbb{E}(y - f_T(\mathbf{x}))^2 \qquad (\mathbf{x}, y) \textit{ independent} \text{ from } T,$$

$$\begin{aligned} E_T &= \frac{1}{N}\sum_{n=1}^{N}(y_n - f_T(\mathbf{x}_n))^2 \\ &\cong \mathbb{E}(y - f_T(\mathbf{x}))^2 \qquad (\mathbf{x}, y) \textit{ contained} \text{ in } T. \end{aligned}$$

Estimating the test error

Demonstration later in context of linear models

We find a bias–variance decomposition for the training error. But there will be another term!

Remember: $(\mathsf{x}_n, y_n) \in T$. Then

$$
\begin{aligned}
E_T \cong\ & \mathbb{E}(y_n - f_T(\mathsf{x}_n))^2 \\
=\ & \mathbb{E}(y_n - \bar{f}(\mathsf{x}_n))^2 && \text{bias} \\
& + \mathbb{E}(\bar{f}(\mathsf{x}_n) - f_T(\mathsf{x}_n))^2 && \text{variance} \\
& - 2\mathbb{E}(y_n - \bar{f}(\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n)) \\
=\ & \mathbf{e}_{\text{test}} - 2\,\underbrace{\mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n))}_{\spadesuit}
\end{aligned}
$$

The term $\spadesuit$ is the correlation between y_n and $f_T(\mathsf{x}_n)$ at fixed x_n, averaged over x_n.

Estimating the test error

Demonstration later in context of linear models

We find a bias–variance decomposition for the training error. But there will be another term!
Remember: $(\mathbf{x}_n, y_n) \in T$. Then

$$
\begin{aligned}
E_T &\cong \mathbb{E}(y_n - f_T(\mathbf{x}_n))^2 \\
&= \mathbb{E}(y_n - \bar{f}(\mathbf{x}_n))^2 \qquad \text{bias} \\
&\quad + \mathbb{E}(\bar{f}(\mathbf{x}_n) - f_T(\mathbf{x}_n))^2 \qquad \text{variance} \\
&\quad - 2\mathbb{E}(y_n - \bar{f}(\mathbf{x}_n))(f_T(\mathbf{x}_n) - \bar{f}(\mathbf{x}_n)) \\
&= \mathbf{e}_{\text{test}} - 2\,\underbrace{\mathbb{E}(y_n - \mathbb{E}(y_n|\mathbf{x}_n))(f_T(\mathbf{x}_n) - \bar{f}(\mathbf{x}_n))}_{\spadesuit}
\end{aligned}
$$

The term $\spadesuit$ is the correlation between y_n and $f_T(\mathbf{x}_n)$ at fixed x_n, averaged over x_n.

Estimating the test error

Demonstration later in context of linear models

We find a bias–variance decomposition for the training error. But there will be another term!
Remember: $(\mathsf{x}_n, y_n) \in T$. Then

$$
\begin{aligned}
E_T &\cong \mathbb{E}(y_n - f_T(\mathsf{x}_n))^2 \\
&= \mathbb{E}(y_n - \bar{f}(\mathsf{x}_n))^2 \qquad \text{bias} \\
&\quad + \mathbb{E}(\bar{f}(\mathsf{x}_n) - f_T(\mathsf{x}_n))^2 \qquad \text{variance} \\
&\quad - 2\mathbb{E}(y_n - \bar{f}(\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n)) \\
&= \mathbf{e}_{\text{test}} - 2\,\underbrace{\mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n))}_{\spadesuit}
\end{aligned}
$$

The term $\spadesuit$ is the correlation between y_n and $f_T(\mathsf{x}_n)$ at fixed x_n, averaged over x_n.

Estimating the test error

Demonstration later in context of linear models

We find a bias–variance decomposition for the training error. But there will be another term!
Remember: $(\mathbf{x}_n, y_n) \in T$. Then

$$
\begin{aligned}
E_T &\cong \mathbb{E}(y_n - f_T(\mathbf{x}_n))^2 \\
&= \mathbb{E}(y_n - \bar{f}(\mathbf{x}_n))^2 \qquad \text{bias} \\
&\quad + \mathbb{E}(\bar{f}(\mathbf{x}_n) - f_T(\mathbf{x}_n))^2 \qquad \text{variance} \\
&\quad - 2\mathbb{E}(y_n - \bar{f}(\mathbf{x}_n))(f_T(\mathbf{x}_n) - \bar{f}(\mathbf{x}_n)) \\
&= \mathbf{e}_{\text{test}} - 2\underbrace{\mathbb{E}(y_n - \mathbb{E}(y_n|\mathbf{x}_n))(f_T(\mathbf{x}_n) - \bar{f}(\mathbf{x}_n))}_{\spadesuit}
\end{aligned}
$$

The term ♠ is the correlation between y_n and $f_T(\mathbf{x}_n)$ at fixed x_n, averaged over x_n.

The linear model

- $T = \{(\mathsf{x}_n, y_n), n = 1, \ldots, N\}$ with $\mathsf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$ (d potentially very large);
- model class $\mathcal{F} = \{f(\mathsf{x}) = \beta^t \mathsf{x}, \beta \in \mathbb{R}^d\}$
- loss function $L(y, \hat{y}) = (y - \hat{y})^2$
- measure of complexity $\kappa(\beta) = |\beta|^2$.

A few remarks

- the models are linear *in the parameters*, but can be nonlinear in the features; to treat models of the form $f(\mathsf{x}) = \beta^t \phi(\mathsf{x})$ just introduce new features $\mathsf{z} = \phi(\mathsf{x})$;
- Rather than minimising training error under constraint, we may minimise

$$R_T := \frac{1}{N} \sum_{n=1}^{N} (y_n - \beta^t \mathsf{x}_n)^2 + \lambda |\beta|^2$$

The linear model

- $T = \{(\mathbf{x}_n, y_n), n = 1, \ldots, N\}$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$ (d potentially very large);
- model class $\mathcal{F} = \{f(\mathbf{x}) = \beta^t \mathbf{x}, \beta \in \mathbb{R}^d\}$
- loss function $L(y, \hat{y}) = (y - \hat{y})^2$
- measure of complexity $\kappa(\beta) = |\beta|^2$.

A few remarks

- the models are linear *in the parameters*, but can be nonlinear in the features; to treat models of the form $f(\mathbf{x}) = \beta^t \phi(\mathbf{x})$ just introduce new features $\mathbf{z} = \phi(\mathbf{x})$;
- Rather than minimising training error under constraint, we may minimise

$$R_T := \frac{1}{N} \sum_{n=1}^{N} (y_n - \beta^t \mathbf{x}_n)^2 + \lambda |\beta|^2$$

The linear model

- $T = \{(\mathbf{x}_n, y_n), n = 1, \dots, N\}$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$ (d potentially very large);
- model class $\mathcal{F} = \{f(\mathbf{x}) = \beta^t \mathbf{x}, \beta \in \mathbb{R}^d\}$
- loss function $L(y, \hat{y}) = (y - \hat{y})^2$
- measure of complexity $\kappa(\beta) = |\beta|^2$.

A few remarks

- the models are linear *in the parameters*, but can be nonlinear in the features; to treat models of the form $f(\mathbf{x}) = \beta^t \phi(\mathbf{x})$ just introduce new features $\mathbf{z} = \phi(\mathbf{x})$;
- Rather than minimising training error under constraint, we may minimise

$$R_T := \frac{1}{N} \sum_{n=1}^{N} (y_n - \beta^t \mathbf{x}_n)^2 + \lambda |\beta|^2$$

The linear model

- $T = \{(\mathbf{x}_n, y_n), n = 1, \dots, N\}$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$ (d potentially very large);
- model class $\mathcal{F} = \{f(\mathbf{x}) = \beta^t \mathbf{x}, \beta \in \mathbb{R}^d\}$
- loss function $L(y, \hat{y}) = (y - \hat{y})^2$
- measure of complexity $\kappa(\beta) = |\beta|^2$.

A few remarks

- the models are linear *in the parameters*, but can be nonlinear in the features; to treat models of the form $f(\mathbf{x}) = \beta^t \phi(\mathbf{x})$ just introduce new features $\mathbf{z} = \phi(\mathbf{x})$;
- Rather than minimising training error under constraint, we may minimise

$$R_T := \frac{1}{N} \sum_{n=1}^{N} (y_n - \beta^t \mathbf{x}_n)^2 + \lambda |\beta|^2$$

The linear model

- $T = \{(\mathbf{x}_n, y_n), n = 1, \ldots, N\}$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$ (d potentially very large);
- model class $\mathcal{F} = \{f(\mathbf{x}) = \beta^t \mathbf{x}, \beta \in \mathbb{R}^d\}$
- loss function $L(y, \hat{y}) = (y - \hat{y})^2$
- measure of complexity $\kappa(\beta) = |\beta|^2$.

A few remarks

- the models are linear *in the parameters*, but can be nonlinear in the features; to treat models of the form $f(\mathbf{x}) = \beta^t \phi(\mathbf{x})$ just introduce new features $\mathbf{z} = \phi(\mathbf{x})$;
- Rather than minimising training error under constraint, we may minimise

$$R_T := \frac{1}{N} \sum_{n=1}^{N} (y_n - \beta^t \mathbf{x}_n)^2 + \lambda |\beta|^2$$

The linear model

- $T = \{(\mathbf{x}_n, y_n), n = 1, \ldots, N\}$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \mathbb{R}$ (d potentially very large);
- model class $\mathcal{F} = \{f(\mathbf{x}) = \beta^t \mathbf{x}, \beta \in \mathbb{R}^d\}$
- loss function $L(y, \hat{y}) = (y - \hat{y})^2$
- measure of complexity $\kappa(\beta) = |\beta|^2$.

A few remarks

- the models are linear *in the parameters*, but can be nonlinear in the features; to treat models of the form $f(\mathbf{x}) = \beta^t \phi(\mathbf{x})$ just introduce new features $\mathbf{z} = \phi(\mathbf{x})$;
- Rather than minimising training error under constraint, we may minimise

$$R_T := \frac{1}{N} \sum_{n=1}^{N} (y_n - \beta^t \mathbf{x}_n)^2 + \lambda |\beta|^2$$

The linear model

continued

Convenient to introduce notation

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}_1^t \\ \vdots \\ \mathbf{x}_N^t \end{bmatrix} \qquad Y := \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Then fitted parameters can be written as

$$\beta = (\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y.$$

We define the fitted outputs $\hat{y}_n = \beta^t \mathbf{x}_n$ and

$$\hat{Y} := \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \mathbf{X}\beta = \mathbf{X}(\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y = HY$$

with *hat matrix* H (it puts the hat on the y's).

The linear model

continued

Convenient to introduce notation

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}_1^t \\ \vdots \\ \mathbf{x}_N^t \end{bmatrix} \qquad Y := \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Then fitted parameters can be written as

$$\beta = (\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y.$$

We define the fitted outputs $\hat{y}_n = \beta^t \mathbf{x}_n$ and

$$\hat{Y} := \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \mathbf{X}\beta = \mathbf{X}(\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y = HY$$

with *hat matrix* H (it puts the hat on the y's).

The linear model

continued

Convenient to introduce notation

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}_1^t \\ \vdots \\ \mathbf{x}_N^t \end{bmatrix} \qquad Y := \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Then fitted parameters can be written as

$$\beta = (\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y.$$

We define the fitted outputs $\hat{y}_n = \beta^t \mathbf{x}_n$ and

$$\hat{Y} := \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \mathbf{X}\beta = \mathbf{X}(\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y = HY$$

with *hat matrix* H (it puts the hat on the y's).

Estimating the test error for the linear model

Assumption for estimating test error:
$\beta = (\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y.$

$$E_T \cong \mathbf{e}_{\text{test}} - 2\underbrace{\mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n))}_{\spadesuit}$$

with

$$\spadesuit = \mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n)) = \frac{1}{N}\mathbb{E}r_n^2\,\mathbb{E}\,\mathrm{tr}(H)$$

Estimating the test error for the linear model

Assumption for estimating test error:

$\beta = (\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y.$

$$E_T \cong \mathbf{e}_{\text{test}} - 2\underbrace{\mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n))}_{\spadesuit}$$

with

$$\spadesuit = \mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n)) = \frac{1}{N}\mathbb{E}r_n^2\,\mathbb{E}\,\mathrm{tr}(H)$$

Estimating the test error for the linear model

Assumption for estimating test error:
$\beta = (\mathbf{X}^t\mathbf{X} + N\lambda)^{-1}\mathbf{X}^t Y$.

$$E_T \cong \mathbf{e}_{\text{test}} - 2\underbrace{\mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n))}_{\spadesuit}$$

with

$$\spadesuit = \mathbb{E}(y_n - \mathbb{E}(y_n|\mathsf{x}_n))(f_T(\mathsf{x}_n) - \bar{f}(\mathsf{x}_n)) = \frac{1}{N}\mathbb{E}r_n^2\,\mathbb{E}\,\mathrm{tr}(H)$$

Setup of Data Assimilation

Consider *signal process* $\{Z_0, Z_1, Z_2, \dots\}$ satisfying

$$Z_{n+1} = \mathcal{M}(Z_n, \theta) + R_{n+1}, \qquad n = 0, 1, \dots,$$

on some *state space* E and with model $\mathcal{M}$ and *unknown parameter*. The *observation process* $\{Y_1, Y_2, \dots\}$ is given by

$$Y_n = \mathcal{H}(X) + S_n, \qquad n = 1, 2, \dots.$$

Problem statement

Estimate θ (along with Z_n) from observations $Y_1, Y_2, \dots$.

Cannot be mapped 100% to ML framework as presented so far.

Setup of Data Assimilation

Consider *signal process* $\{Z_0, Z_1, Z_2, \dots\}$ satisfying

$$Z_{n+1} = \mathcal{M}(Z_n, \theta) + R_{n+1}, \qquad n = 0, 1, \dots,$$

on some *state space* E and with model $\mathcal{M}$ and *unknown parameter*. The *observation process* $\{Y_1, Y_2, \dots\}$ is given by

$$Y_n = \mathcal{H}(X) + S_n, \qquad n = 1, 2, \dots.$$

Problem statement

Estimate θ (along with Z_n) from observations $Y_1, Y_2, \dots$.

Cannot be mapped 100% to ML framework as presented so far.

Setup of Data Assimilation

Consider *signal process* $\{Z_0, Z_1, Z_2, \ldots\}$ satisfying

$$Z_{n+1} = \mathcal{M}(Z_n, \theta) + R_{n+1}, \qquad n = 0, 1, \ldots,$$

on some *state space* E and with model $\mathcal{M}$ and *unknown parameter*. The *observation process* $\{Y_1, Y_2, \ldots\}$ is given by

$$Y_n = \mathcal{H}(X) + S_n, \qquad n = 1, 2, \ldots.$$

Problem statement

Estimate θ (along with Z_n) from observations $Y_1, Y_2, \ldots$.

Cannot be mapped 100% to ML framework as presented so far.

Relation with ML I

Idea:

Estimate θ by using Y_n as target and $Y_1, \ldots, Y_{n-1}$ as feature for each $n = 1, 2, \ldots$.

Loss minimisation principle:

Find θ by minimising *prediction error*

$$E(\theta) := \frac{1}{N} \sum_{n=1}^{N} L(Y_n, \hat{Y}_n)$$

where $\hat{Y}_n$ is a predition of Y_n computed through DA. Dependence on θ is implicit in $\hat{Y}_n$.

Relation with ML II

More general method: Maximum likelihood approach

Find θ by minimising *prediction error*

$$\mathcal{L}(\theta) := \log p_\theta(Y_1, \ldots, Y_n)$$

where $p_\theta(\ldots)$ is the probability density of $Y_1, \ldots, Y_n$. Computation of this *very difficult* but comes as a by–product of fully nonlinear data assimilation.

Alternative method: adjoining parameter to state vector

$$\begin{aligned} Z_{n+1} &= A_n Z_n + bf + \rho R_{n+1} \\ Y_n &= Z_n^{(1)} + \sigma S_n \\ A &= \begin{pmatrix} \cos(\omega n) & -\sin(\omega n) \\ \sin(\omega n) & \cos(\omega n) \end{pmatrix}, \qquad f = \begin{pmatrix} 1/2 \\ 1 \end{pmatrix}, \end{aligned}$$

with b unknown parameter.

Estimate b by adjoining another state equation

$$b_{n+1} = b_n,$$

making this a 3–dimensional Data Assimilation problem.

Alternative method: adjoining parameter to state vector

$$
\begin{aligned}
Z_{n+1} &= A_n Z_n + bf + \rho R_{n+1} \\
Y_n &= Z_n^{(1)} + \sigma S_n \\
A &= \begin{pmatrix} \cos(\omega n) & -\sin(\omega n) \\ \sin(\omega n) & \cos(\omega n) \end{pmatrix}, \qquad f = \begin{pmatrix} 1/2 \\ 1 \end{pmatrix},
\end{aligned}
$$

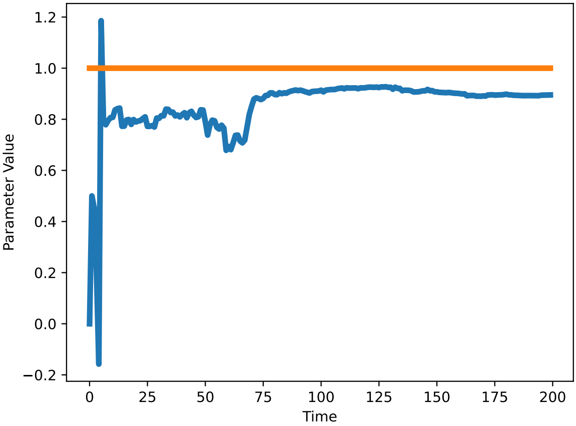
with b unknown parameter.
Estimate b by adjoining another state equation

$$b_{n+1} = b_n,$$

making this a 3–dimensional Data Assimilation problem.

Alternative method: adjoining parameter to state vector

Results



For further reading

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer, New York, second edition, 2009.

Jochen Bröcker, David Engster, and Ulrich Parlitz.
Probabilistic evaluation of time series models; a comparison of several approaches.
Chaos, 19, 2009.

Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino.
Combining data assimilation and machine learning to infer unresolved scale parametrization.
Philosophical Transactions of the Royal Society A, 379(2194): 20200086, 2021.