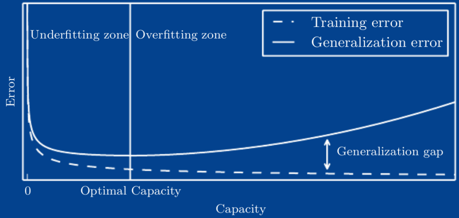
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

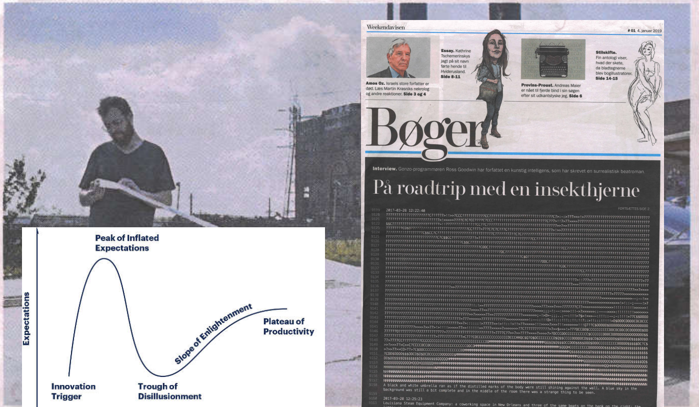
LESSON 8: Model-capacity, Under- and Overfitting, Generalization

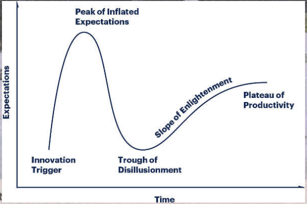
CARSTEN EIE FRIGAARD

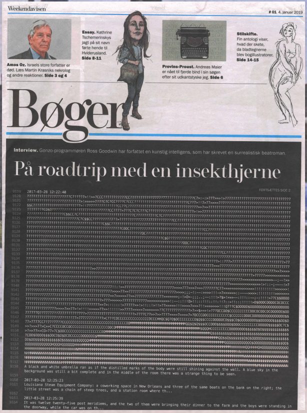
AUTUMN 2020



SmartBus
Hop ombord på Danmarks første førerløse bus.
Powered by holo
Aalborg Kommune
CL 52 074







Weekendavisen # 01 4. januar 2019

Amos Oz. Israels store forfatter er død. Læs Martin Krasniks nekrolog og andre reaktioner. **Side 3 og 4**

Essay. Kathrine Tschemerinskys jagt på sit navn førte hende til Hviderusland. **Side 8-11**

Provins-Proust. Andreas Maier er nået til fjerde bind i sin søgen efter sit udkantstyske jeg. **Side 6**

Stilskifte. Fin antologi viser, hvad der skete, da bladtegnerne blev bogillustratorer. **Side 14-15**

Bøger

Interview. Gonzo-programmøren Ross Goodwin har forfattet en kunstig intelligens, som har skrevet en surrealistisk beatroman.

På roadtrip med en insekthjerne

FORTSÆTTES SIDE 2

```
2017-03-28 12:22:40
????????????????????C??????>!!>>?CCC???????????CC?????????????????????????????????C?>:-:>???>>>!>??????????????????????????????????
??????????????????????????>!>>>>>????C?C?CC????C?CCC?????????????????????????????C???>!!?>??>>>>????????????????????????????????
$$C???????????????????????>!!???????????????CC??C??????????????????????????????????????>>?>>????????????????????????????????????
???????COSO?????????????????????????????CC????>???C?C?C???C???????????????????????>>?>>????????????????????????????????????????
??????????????????C$$CC?C?????????????????????????????C??????????C?C?????????????????????????????????????????????????????????????
[illegible]
A black and white umbrella ran as if the distilled marks of the body were still shining against the wall. A blue sky in the background was still a bit complete and in the middle of the room there was a strange thing to be seen.

2017-03-28 12:25:23
Louisiana Steam Equipment Company: a coworking space in New Orleans and three of the same boats on the bank on the right; the little street was a chain of steep trees, and a station room where th...

2017-03-28 12:25:39
It was twelve twenty-five post meridiems, and the two of them were bringing their dinner to the farm and the boys were standing in the doorway, while the car was on th...
```

FOR

På roadtrip med en insekthjerne

Af MIKKEL BORIS

I 2016 slog computeren AlphaGo den 18-dobbelte verdensmester i brætspillet Go, Lee Sedol. Go er et kompliceret og abstrakt spil, som kræver intuition og kreativitet, men den kunstige intelligens vandt med en række innovative træk overlegent.

Undervejs i spillet troede kommentatorerne,

Goodwin til Weekendavisen fra sin lejlighed i Los Angeles.

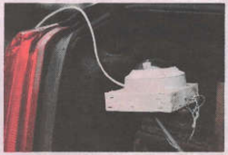
Inden køreturen havde han brugt måneder på at træne maskinen. Han satte den til at læse et stort korpus af moderne litteratur fra hele verden, så den kunne lære at skrive af de store forfattere.

»Det fungerer ligesom autokorrekturen på din telefon, bare klogere og trænet på en mere litterær kilde. Den skriver bogstav for bogstav, så den har lært sig selv at forudsige det næste

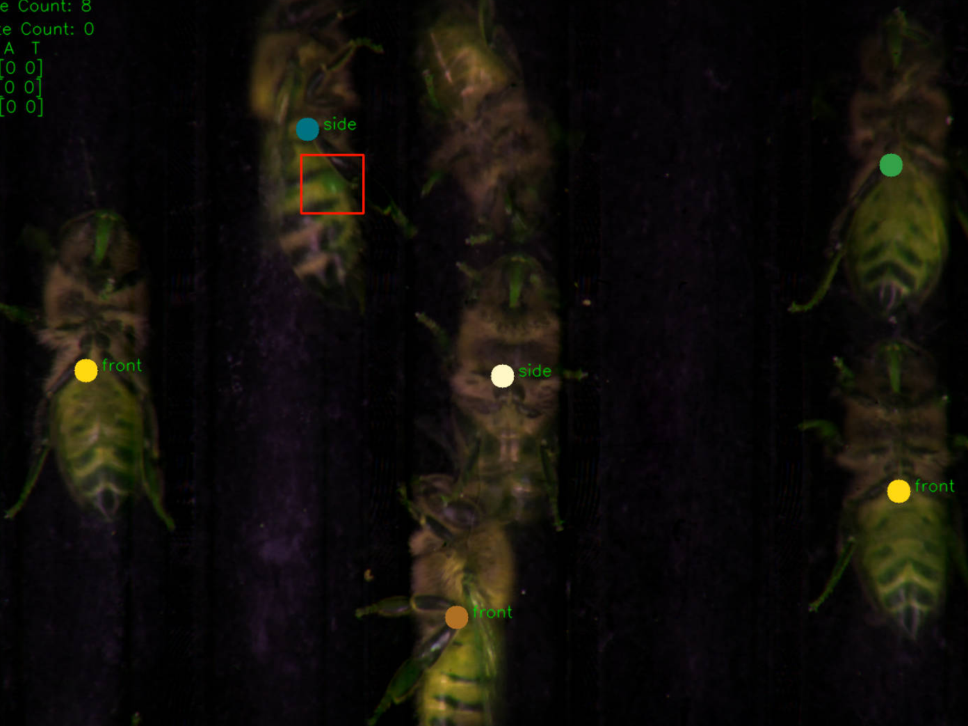
når du dekonstruerer dem. Efter at have læst den i ét stræk og fået turen lidt på afstand har romanen fået en universalitet, så jeg kan projicere mine egne oplevelser ind i teksten,« uddyber Goodwin.

– *Du har beskrevet projektet som at lære en insekthjerne at skrive. Hvad betyder det?*

»Jeg forsøgte at pointere, at maskinen ikke er på niveau med den menneskelige hjerne. Et artificielt neuralt net er en algoritme, der er lavet, som vi tror, hjerner fungerer. Jeg synes



»Det er et forsøg på at skabe en ny brugerflade for at skrive. På en måde har jeg jo skrevet en roman med en bil,« fortæller Ross Goodwin om sin AI-forfattede bog *1 the road*


e Count: 8
te Count: 0
A T
[0 0]
[0 0]
[0 0]
side
front
side
front
front

A computer vision system to monitor the infestation level of Varroa destructor in a honeybee colony

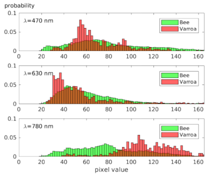


Figure 5: Histograms of bee and varroa pixel intensity values, for the spectral wavelengths 470 nm, 630 nm, and 780 nm respectively, recorded with the JAI camera. The image path is via the mirror-window-mirror, i.e. data were sampled with the setup given in figure 1. The image data for the histogram is the single bee with mite seen in figure 6.

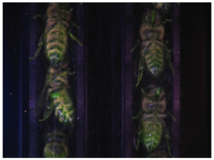


Figure 6: The actual unprocessed camera view of the bees

spectively. The CM analysis was able to rank all wavelengths combinations, using one, two, three or four district wavelengths to give a ranking list of 'best' combination also taking the JAI camera spectrum into account.

The CM value of the actual choose wavelengths combination (470-630-780 nm) gave a rank just below the CM average score. This CM analysis was conducted after picking the actual used wavelengths, so later versions of the VMU might want to investigate a CM combination with a higher rank.

A specially designed diffuser and a number of narrow spectral LE
were mounted i
the camera focal
diffuse illuminat
flections.

Figure 6 disp
along the passa
era, with the gre
with the NIR m

*2.3.3. Real-time
processin*

A color and
tion of 1296×96
from the camer
over two separat
sary sustained b
ing frames real-

These data w
line post-proces
first matching t
rally coalescing
producing a 24
the later image

Lossless real-t
can be applied
bandwidth than

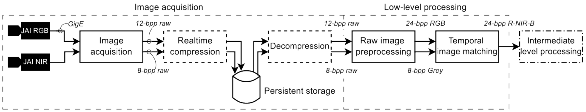


Figure 7: The low-level image processing pipeline. Raw camera images are stored on disk for later retrieval and post-processing. 12- and 8 bits-per-pixel are used as the raw JAI/Bayer packed pixel format for the RGB and IR images respectively. Lossless, real-time compression can be introduced if persistent storage bandwidth is less than the raw-stream image rate of 93 MiB/sec. The 12- and 8-bpp raw images from the network arrives out-of-order with respect to each other, hence the need for the temporal image matching.

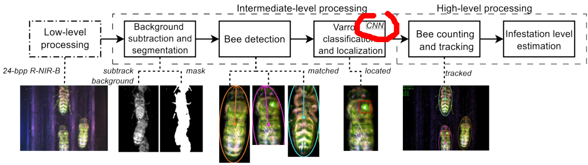
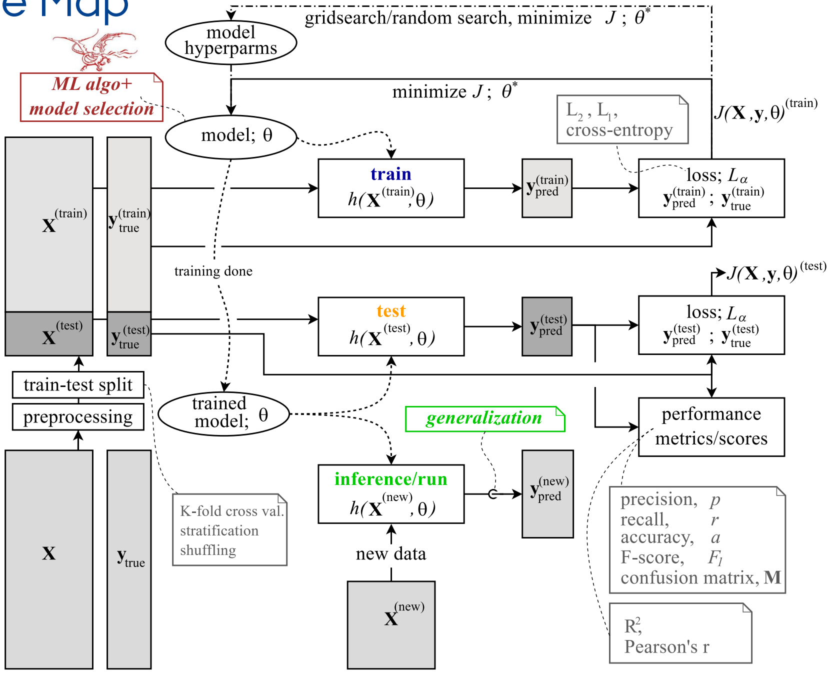

Intermediate-level processing
High-level processing
Low-level processing
Background subtraction and segmentation
Bee detection
Varroa classification and localization
CNN
Bee counting and tracking
Infestation level estimation
24-bpp R-NIR-B
subtrack background
mask
matched
located
tracked

Figure 8: The processing pipeline of the intermediate- to high-level image processing algorithms to analyze and count the number of bees with *Varroa destructor*. A trained convolutional neural network (CNN) was used for the Varroa classification and localization stage.

BA Project: generic tagging/labling tool

The Map



ML algo+ model selection
model hyperparms
gridsearch/random search, minimize J ; θ^*
minimize J ; θ^*
model; θ
$J(\mathbf{X},\mathbf{y},\theta)^{(train)}$
L_2, L_1, cross-entropy
train $h(\mathbf{X}^{(train)},\theta)$
$\mathbf{y}^{(train)}_{pred}$
loss; L_α $\mathbf{y}^{(train)}_{pred}$; $\mathbf{y}^{(train)}_{true}$
$\mathbf{X}^{(train)}$
$\mathbf{y}^{(train)}_{true}$
training done
$J(\mathbf{X},\mathbf{y},\theta)^{(test)}$
$\mathbf{X}^{(test)}$
$\mathbf{y}^{(test)}_{true}$
test $h(\mathbf{X}^{(test)},\theta)$
$\mathbf{y}^{(test)}_{pred}$
loss; L_α $\mathbf{y}^{(test)}_{pred}$; $\mathbf{y}^{(test)}_{true}$
train-test split
preprocessing
trained model; θ
generalization
performance metrics/scores
$\mathbf{X}$
$\mathbf{y}_{true}$
K-fold cross val. stratification shuffling
inference/run $h(\mathbf{X}^{(new)},\theta)$
$\mathbf{y}^{(new)}_{pred}$
new data
$\mathbf{X}^{(new)}$
precision, p
recall, r
accuracy, a
F-score, F_1
confusion matrix, $\mathbf{M}$
R^2,
Pearson's r

Preprocessing of Data

Scaling, Standardization, Normalization...

Why the need for preprocessing?

> *Standardization of datasets is a* **common requirement for many machine learning estimators** *[..]; they might* **behave badly** *if the individual features do not more or less look like standard normally distributed data. [..]*
>
> [`https://scikit-learn.org/stable/modules/preprocessing.html`]

Standardization of a feature vector **x**, giving **x**′ mean zero, and standard deviation one

$$\mathbf{x}' = \frac{\mathbf{x} - \mu\_\mathbf{x}}{\sigma\_\mathbf{x}}$$

What kind of estimators needs standardization?

$\rightarrow$ **Neural networks (NNs) in particular!**

Regularization and optimization:

$\rightarrow$ WARM-UP for the comming use of NNs!

Pipelines

Revisit the Housing data in 01s for MLPs

Feature: GDP per capita feature in range 10K to 50K $.
But MLP expects input in the range [0;1] or perhaps [-1;1].

```python
# Manual scaling..
X_min = np.min(X)
X_max = np.max(X)
s = X_max - X_min

print(f"X_min={X_min:.0f},  X_max={X_max:.0f},  s={s:.0f}")

X_scaled = (X-X_min)/s
print(f"X_scaled.shape={X_scaled.shape}")
print(f"np.min(X_scaled)={np.min(X_scaled)}")
print(f"np.max(X_scaled)={np.max(X_scaled)}")

mlp.fit(X_scaled ,y.ravel())
y_pred_mlp = mlp.predict((M-X_min)/s)

plt.plot(m, y_pred_lin, "r")
#plt.plot(m, y_pred_knn, "b")
plt.plot(m, y_pred_mlp, "k")

print(f"mpl.score={mlp.score(X_scaled, y.ravel()):0.2f}")
```

```
Prints:
X_min=9055,
X_max=55805, s=46750
X_scaled.shape=(29, 1)
np.min(X_scaled)=0.0
np.max(X_scaled)=1.0
mpl.score=0.70
```

Pipelines

Housing Data and MLPs: introducing a `MinMaxScaler`

```python
# Now, do the same but via a pipeline..
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(X)
X_scaled2 = scaler.transform(X)
M_scaled = scaler.transform(M)

mlp.fit(X_scaled2, y.ravel())
y_pred_mlp = mlp.predict(M_scaled)

print(f"mpl.score={mlp.score(X_scaled, y.ravel()):0.2f}")

# PRINTS: mpl.score=0.71
```

Pipelines

Housing Data and MLPs: putting everything in a Full Pipeline

```
# Or even better, in a full pipeline..
from sklearn.pipeline import Pipeline

pipe = Pipeline(
  [
    ('scaler', MinMaxScaler()),
    ('mlp', mlp)
  ])
pipe.fit(X, y.ravel())

print(f"pipe.score(..) = {pipe.score(X, y.ravel()):0.2f}"

# PRINTS: mpl.score=0.68
```

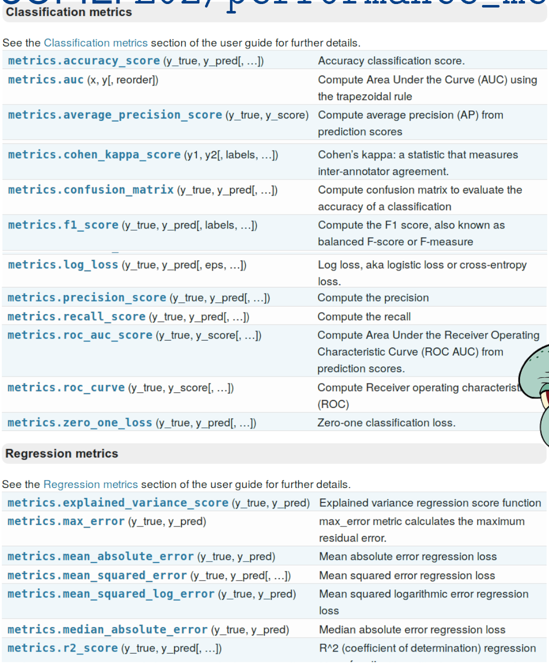
Pipelines

Brief intro to Scikit-learn pipelines..

Python code from `capacity_under_overfitting.ipynb`

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score

..

polynomial_features = PolynomialFeatures(degree=degrees[i], ..
linear_regression = LinearRegression()

pipeline = Pipeline([
        ("polynomial_features", polynomial_features),
        ("linear_regression", linear_regression)
    ])
pipeline.fit(X[:, np.newaxis], y)

scores = cross_val_score(pipeline, X[:, np.newaxis], y, scoring=
    "neg_mean_squared_error", cv=10)
..
score_mean = -scores.mean()
```

RESUMÉ: `L02/performance_metrics.ipynb`



Classification metrics

See the Classification metrics section of the user guide for further details.

`metrics.accuracy_score` (y_true, y_pred[, …])	Accuracy classification score.
`metrics.auc` (x, y[, reorder])	Compute Area Under the Curve (AUC) using the trapezoidal rule
`metrics.average_precision_score` (y_true, y_score)	Compute average precision (AP) from prediction scores
`metrics.cohen_kappa_score` (y1, y2[, labels, …])	Cohen's kappa: a statistic that measures inter-annotator agreement.
`metrics.confusion_matrix` (y_true, y_pred[, …])	Compute confusion matrix to evaluate the accuracy of a classification
`metrics.f1_score` (y_true, y_pred[, labels, …])	Compute the F1 score, also known as balanced F-score or F-measure
`metrics.log_loss` (y_true, y_pred[, eps, …])	Log loss, aka logistic loss or cross-entropy loss.
`metrics.precision_score` (y_true, y_pred[, …])	Compute the precision
`metrics.recall_score` (y_true, y_pred[, …])	Compute the recall
`metrics.roc_auc_score` (y_true, y_score[, …])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
`metrics.roc_curve` (y_true, y_score[, …])	Compute Receiver operating characterist (ROC)
`metrics.zero_one_loss` (y_true, y_pred[, …])	Zero-one classification loss.

Regression metrics

See the Regression metrics section of the user guide for further details.

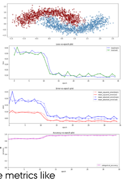
`metrics.explained_variance_score` (y_true, y_pred)	Explained variance regression score function
`metrics.max_error` (y_true, y_pred)	max_error metric calculates the maximum residual error.
`metrics.mean_absolute_error` (y_true, y_pred)	Mean absolute error regression loss
`metrics.mean_squared_error` (y_true, y_pred[, …])	Mean squared error regression loss
`metrics.mean_squared_log_error` (y_true, y_pred)	Mean squared logarithmic error regression loss
`metrics.median_absolute_error` (y_true, y_pred)	Median absolute error regression loss
`metrics.r2_score` (y_true, y_pred[, …])	R^2 (coefficient of determination) regression

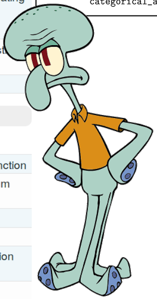
Notes on Keras MLPs

Typical Keras MLP Supervised Classifier setup..

- loss function
  `loss='categorical_crossentropy'`
- metrics collected via history
  ```
  metrics=[
    'categorical_accuracy',
    'mean_squared_error',
    'mean_absolute_error'])
  ```
- input lay.: categorical encoding,
- output lay.: softmax function,

And notice that Keras do *not* provide metrics like
`precision, recall, F1`
but instead
`categorical_accuracy, binary_accuracy`





Model capacity

Exercise: `capacity_under_overfitting.ipynb`

Dummy and Paradox classifier:
capacity fixed ~ 0, cannot generalize at all!

Linear regression for a polynomial model:
capacity $\sim$ degree of the polynomial, x^n

Neural Network model:
capacity $\propto$ number of neurons/layers

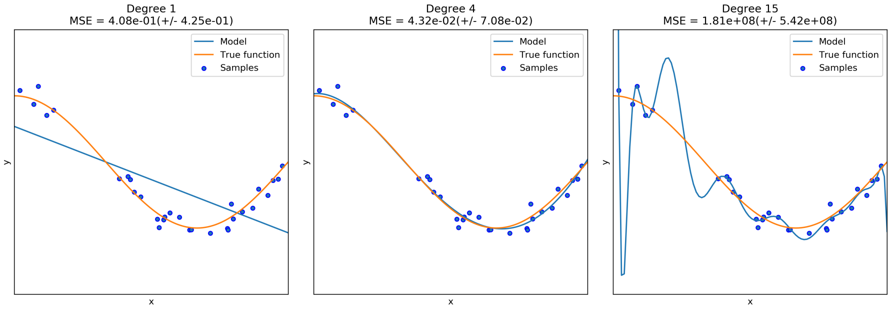
$\Rightarrow$ **Capacity** can be hard to express as a quantity for some models, but you need to choose..

$\Longrightarrow$ how to choose the **optimal** *capacity*?

Under- and overfitting

Exercise: `capacity_under_overfitting.ipynb`

Polynomial linear reg. fit for underlying model: `cos(x)`



- underfitting: capacity of model too low,
- overfitting: capacity to high.

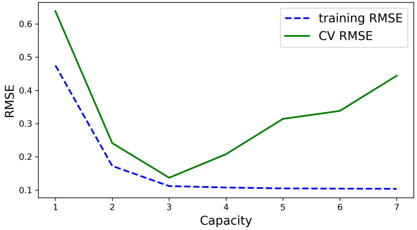
$\Longrightarrow$ how to choose the **optimal** capacity?

Generalization Error

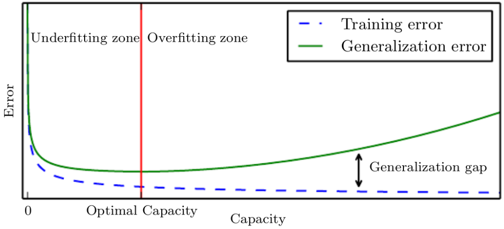
Exercise: `generalization_error.ipynb`

RMSE-capacity plot for lin. reg. with polynomial features

(capacity ∼ degree of poly)



(Figure 5.3 from [DL])



Inspecting the plots from the exercise (`.ipynb`) and [DL], extracting the concepts:

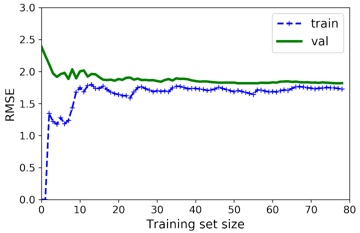
- training/generalization error,
- generalization gab,
- underfit/overfit zone,
- optimal capacity (best-model, early stop),
- (and the two axes: x/capacity, y/error.)

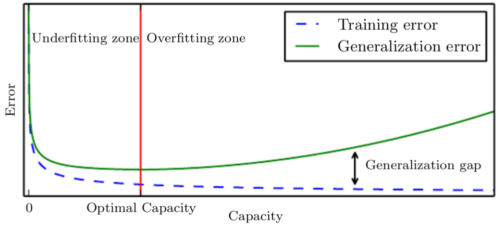
Generalization Error

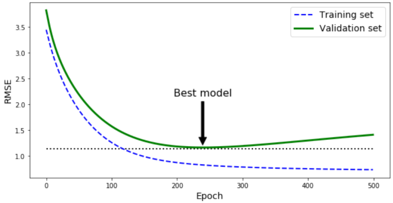
Exercise: `generalization_error.ipynb`

NOTE: three methods/plots:

i) via **learning curves** as in [HOML],
ii) via an **error-capacity** plot as in [GITHOML] and[DL],
iii) via an **error-epoch** plot as in [GITHOML].


train
val
RMSE
Training set size


Underfitting zone
Overfitting zone
Training error
Generalization error
Error
Generalization gap
Optimal Capacity
Capacity


Training set
Validation set
Best model
RMSE
Epoch