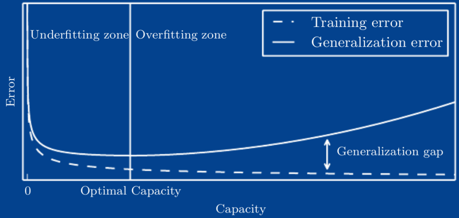
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

LESSON 9: Model-capacity, Under/over-fitting, Generalization

CARSTEN EIE FRIGAARD

AUTUMN 2021



"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E." — Mitchell (1997).

L09: Model-capacity, Under/over-fitting, Generalization

Agenda

- Resumé af GD og NN's.
- Model Capacity,
- Under/over-fitting,
  Exercise: `L09/capacity_under_overfitting.ipynb`
  [OPTIONAL]
- Generalization Error,
  Exercise: `L09/generalization_error.ipynb`
- Searching
  Exercise: `L09/gridsearch.ipynb`

RESUMÉ: GD

The numerically Gradient decent [GD] method is based on the gradient vector

$$\nabla_{\mathbf{w}} J(\mathbf{w})$$

for the gradient oprator

$$\nabla_{\mathbf{w}} = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \ldots, \frac{\partial}{\partial w_m}\right]^\top$$

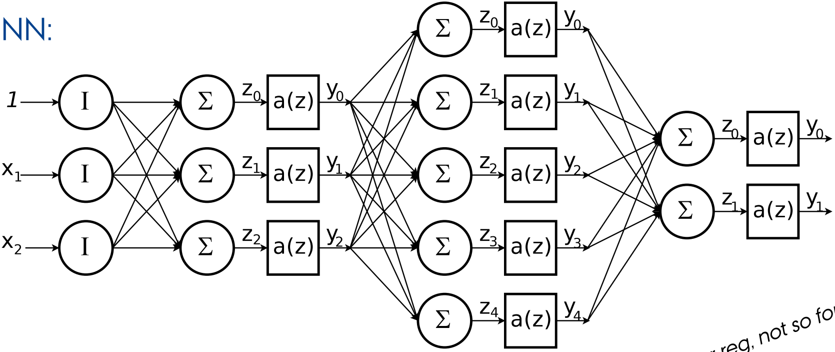
The algoritmn for updating via steps reads

$$\mathbf{w}^{(\text{next step})} = \mathbf{w} - \eta \nabla_{\mathbf{w}} J(\mathbf{w})$$

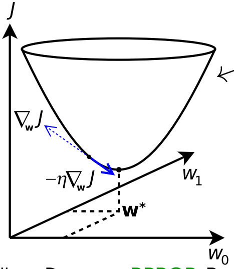
with η being the step size.

RESUMÉ: Training Deep Neural Networks

NN:



GD (via BPROP):



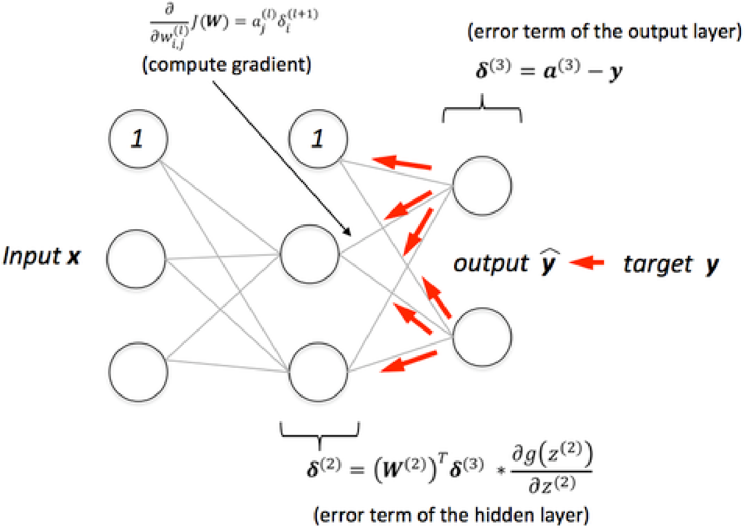
CONVEX ~ linear reg, not so for NNs

$$\mathbf{w}^{(\text{next})} = \mathbf{w} - \eta \nabla_{\mathbf{w}} J(\mathbf{w})$$

NOTE: NN: Neural net, GD: Gradient Descent, BPROP: Back Propagation

Backpropagation (BProp)

Training MLPs



NOTE: [`https://sebastianraschka.com/images/faq/visual-backpropagation/backpropagation.png`

RESUMÉ: Training Deep Neural Networks

Equation 4-6. Gradient vector of the cost function

$$\nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \mathrm{MSE}(\boldsymbol{\theta}) \\ \frac{\partial}{\partial \theta_1} \mathrm{MSE}(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \mathrm{MSE}(\boldsymbol{\theta}) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Notice that this formula involves calculation
set **X**, at each Gradient Descent step! This i
called *Batch Gradient Descent*: it uses the w
data at every step (actually, *Full Gradient D*
be a better name). As a result it is terribly sl
ing sets (but we will see much faster Gradient Descent algorithms
shortly). However, Gradient Descent scales we
features; training a Linear Regression model
dreds of thousands of features is much fa
Descent than using the Normal Equation or SV

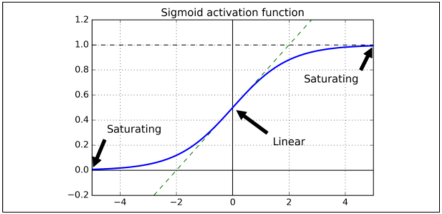


Figure 11-1. Logistic activation function saturation

Once you have the gradient vector, which points uphill, ju
tion to go downhill. This means subtracting $\nabla_{\boldsymbol{\theta}}\mathrm{MSE}(\boldsymbol{\theta})$
learning rate η comes into play:[6] multiply the gradient v
size of the downhill step (Equation 4-7).

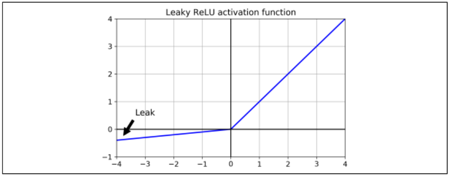


Figure 11-2. Leaky ReLU

Equation 4-7. Gradient Descent step

$$\boldsymbol{\theta}^{(\text{next step})} = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{MSE}(\boldsymbol{\theta})$$

$$\mathbf{w}^{(\text{next})} = \mathbf{w} - \eta \nabla_{\mathbf{w}} J(\mathbf{w})$$

MODEL CAPACITY

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

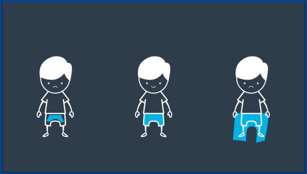
Neural Network model:
capacity $\propto$ number of neurons/layers

Homo sabiens (”modern humans”):
capacity $\propto$ the IQ 'score' function?

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

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

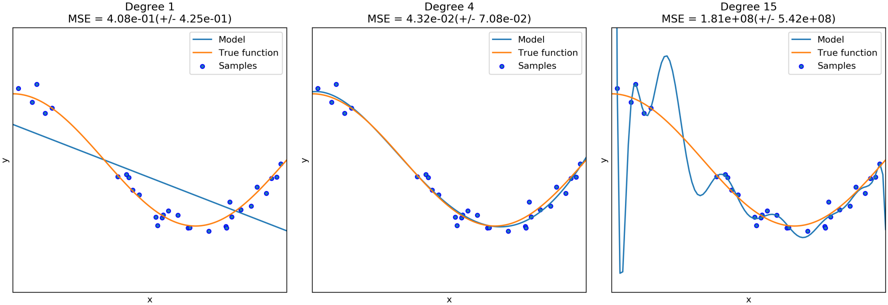
UNDER- AND OVERFITTING



Under- and overfitting

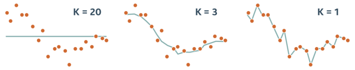
Exercise: `capacity_under_overfitting.ipynb`

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



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

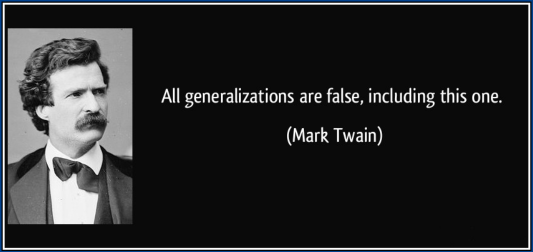
k-NN from L01:



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

NOTE: HOML: *Constraining a model [..] reduce risk of overfitting [via] regularization => L08*

GENERALIZATION ERROR

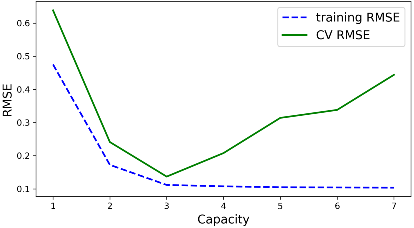


Generalization Error

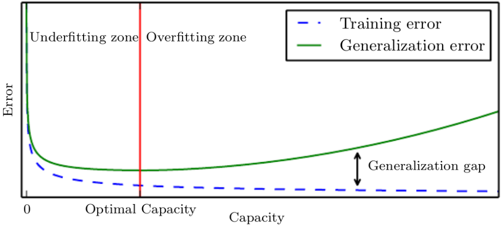
Exercise: `generalization_error.ipynb`

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

(capacity $\sim$ 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

Definition of ML:

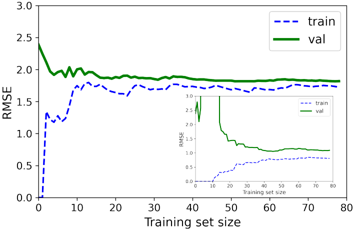
> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T, as measured by P, improves with experience E."
>
> — Mitchell (1997).

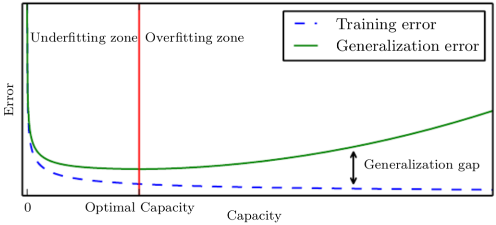
Generalization Error

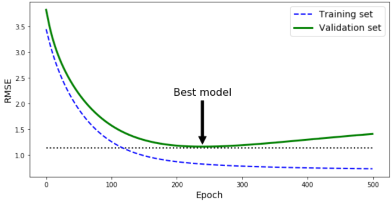
Exercise: `generalization_error.ipynb`

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].







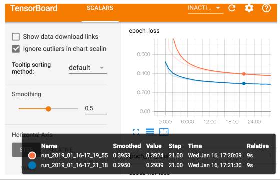


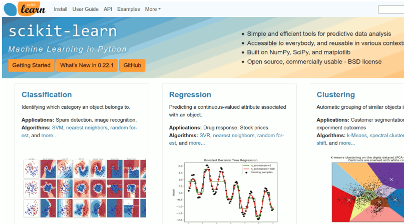
Figure 10-16. Visualizing Learning Curves with TensorBoard

SEARCHING

ML Algorithm +
Model Selection via Searching

ML Models (or ML algorithms)

Models encountered so far



Some classifiers and regressors..

```
sklearn.neighbors.KNeighborsRegressor
sklearn.linear_model.LinearRegression
sklearn.linear_model.SGDClassifier
sklearn.linear_model.SGDRegressor
```

Perhaps..

```
sklearn.naive_bayes.GaussianNB
sklearn.naive_bayes.MultinomialNB
sklearn.svm.SVC
sklearn.svm.SVR
```

and to some degree..

```
sklearn.linear_model.LogisticRegression
sklearn.linear_model.Perceptron
sklearn.neural_network.MLPClassifier
sklearn.neural_network.MLPRegressor
keras.Sequential
```

Or even more exotic models like..

- superviced ensemble: AdaBoost, Bagging, DecisionTree, RandomForest,..
- semi-supervised: ??
- unsupervised: K-means, manifolds, restricted Boltzmann machines,..
- clustering: K-means

ML Algorithm + Model Selection via Searching

What ML algorithm to choose?

- manual:
  algorithm characteristics, $\mathcal{O}$ complexity, etc.
  browsing through Scikit-learn documentation,
  ...and also based on data assumptions.
- semi-automatic:
  brute-force model search, and fun with python!

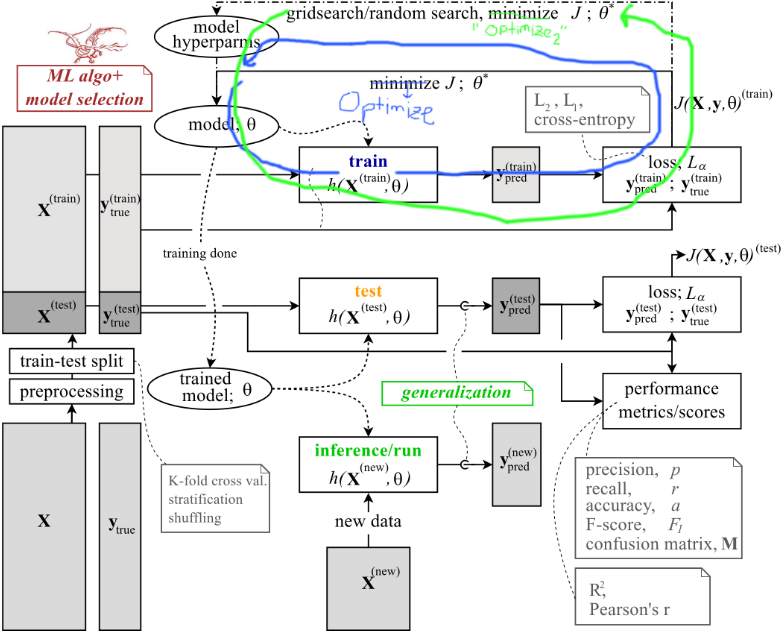
```python
models = {
  SVC(gamma="scale"),
  SGDClassifier(tol=1e-3, eta0=0.1),
  GaussianNB()
}

for i in models:
    i.fit(X_train, y_train)
    y_pred_test = i.predict(X_test)
    p = precision_score(y_test, y_pred_test, average='micro')
    print(f'{type(i).__name__:13s}: precision={p:0.2f}')
```

prints..

```
GaussianNB:    p=1.00
SGDClassifier: p=0.93
SVC:           p=0.98
```

NOTE: Python set = {a, b}
Python dictionary= {a:x, b:y}

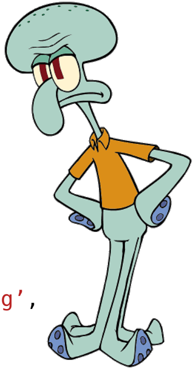
Model Selection via Grid Search



Model Selection via Grid Search

The hyperparamter-set for SGD linear regressor

```
class sklearn.linear_model.SGDRegressor(
  loss    ='squared_loss', penalty      ='l2',
  alpha   =0.0001,         l1_ratio     =0.15,
  tol     =None,           shuffle      =True,
  verbose =0,              epsilon      =0.1,
  eta0    =0.01,           power_t      =0.25,
  n_iter_no_change=5,      warm_start   =False,
  fit_intercept   =True,   max_iter     =None,
  average         =False,  n_iter       =None
  random_state    =None,   learning_rate='invscaling',
  early_stopping  =False,  validation_fraction=0.1
)
```



Search best hyperparameters in a (smaller) set, say

```
model = SGDRegressor()
tuning_parameters = {
  'alpha': [ 0.001, 0.01, 0.1],
  'max_iter': [1, 10, 100, 1000],
  'learning_rate':('constant','optimal','invscaling','adaptive')
}
..
grid_tuned = GridSearchCV(model, tuning_parameters, ..
```

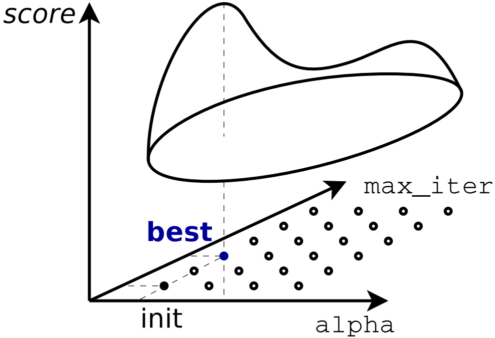
Model Selection via Grid Search

How to select 'best' set of hyperparameter—using bute force?

Gridsearch seen in 3D for the two hyperspace dimensions:

- `alpha` $\in [1, 2, 3, ..]$
- `max_iter` $\in [1, 2, 3, ..]$

(NOTE: linear range for this plot only, should be 1, 10, 100 or similar.)



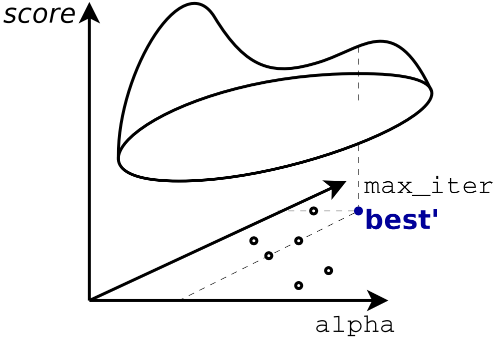
- why `score` and not J on $z-$axis?
- and what if there are many hyperparameters and many combinations? $\rightarrow$ Zzzzzzz!

Model Selection via Randomized Search

How to select 'best' set of hyperparameters—faster than brute force?

Replace `GridSearchCV()` with

`RandomizedSearchCV(n_iter=100,..)`



- faster, but will not yield the (sub) optimal score maximum,
- ...but does it matter in a huge hyperparameter search-space?

Exercise: L09/gridsearch.ipynb

Qd MNIST Search Quest II: Husk at publicer på Brightspace

<i>L08:</i> Regulari ×

https://blackboard.au.dk/webapps/blackboard/content/listContentEditable.jsp?content_id=_2931033_1&course_id=_145075 120%

Qd MNIST Search Quest II

Publicer (helt nederst i denne item) løbende dit bedste resultat fra opgaven

`gridsearch.ipynb` (Qd)

Indtast din model contructor string og score og en kommentar vdr. dit model ala

```
Grp09: best: dat=mnist, score=0.90780, model=SGDClassifier(alpha=1.0,eta0=0.0001,learning_rate='invscaling')

Grp09: CTOR for best model: SGDClassifier(alpha=1.0, average=False, class_weight=None, early_stopping=False,
           epsilon=0.1, eta0=0.0001, fit_intercept=True, l1_ratio=0.15,
           learning_rate='invscaling', loss='hinge', max_iter=1000,
           n_iter_no_change=5, n_jobs=None, penalty='l2', power_t=0.5,
           random_state=None, shuffle=True, tol=0.001,
           validation_fraction=0.1, verbose=0, warm_start=False)
```

NB: brug af **neurale netværks** modeller (Perceptrons, MLP's, Keras etc.) samt **`KNeighborsClassifier`** ikke tilladt i denne quest!

Highscore fra sidste semeter = 0.97369

```
model=KNeighborsClassifier(algorithm='ball_tree', n_neighbors=3, p=4,weights='distance')
```

ITMAL Search Quest

Send message...