Overfitting

Alexandros Iosifidis

Assistant Professor (tenure track)

Aarhus University

alexandros.iosifidis@eng.au.dk

Prior knowledge

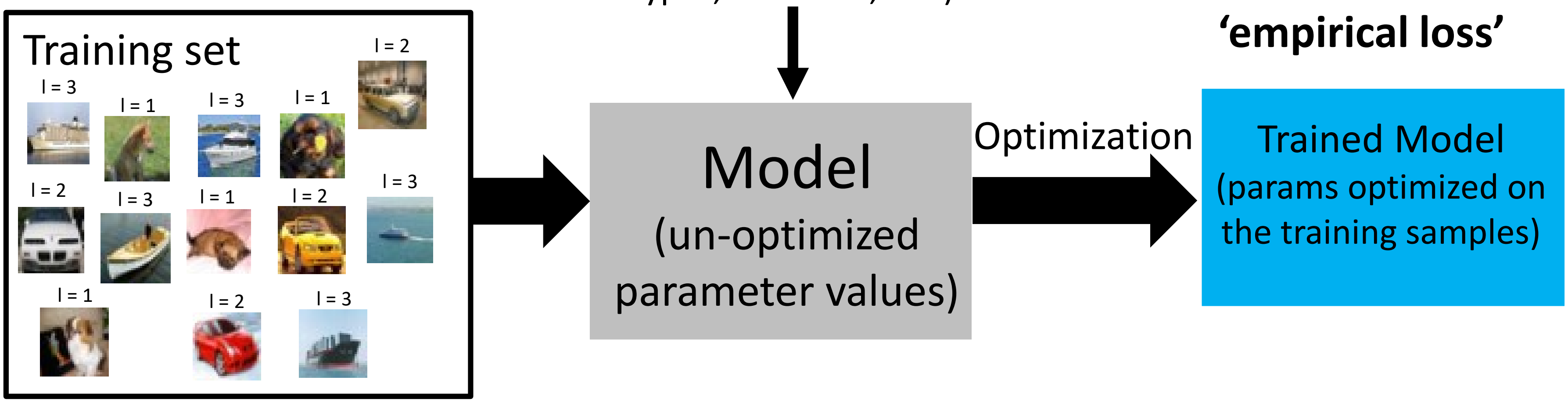
- Supervised/Unsupervised learning
- Supervised learning models
    - Linear regression
    - Basics of artificial neural networks (e.g. single-hidden layer NNs)
- Gradient-based optimization
- Linear Algebra basics (singular value decomposition)

Generalization in Machine Learning

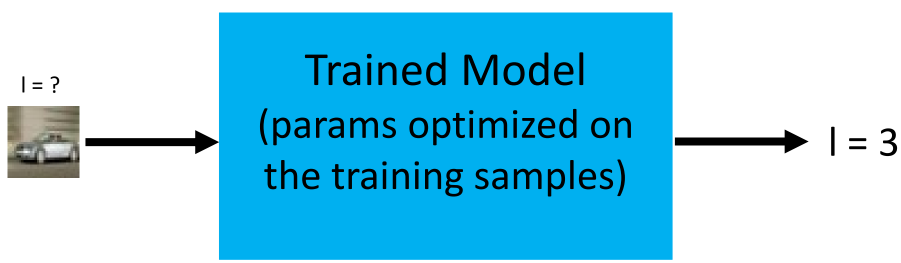
Training phase

Hyper-parameters
(user-given values, function types, structure, etc.)

The error on the training samples is usually called **'empirical loss'**



Test phase or Evaluation/Online process



Generalization ability is measured by using a set of samples that were not used for training.

The error on the test set is usually called **'expected loss'**

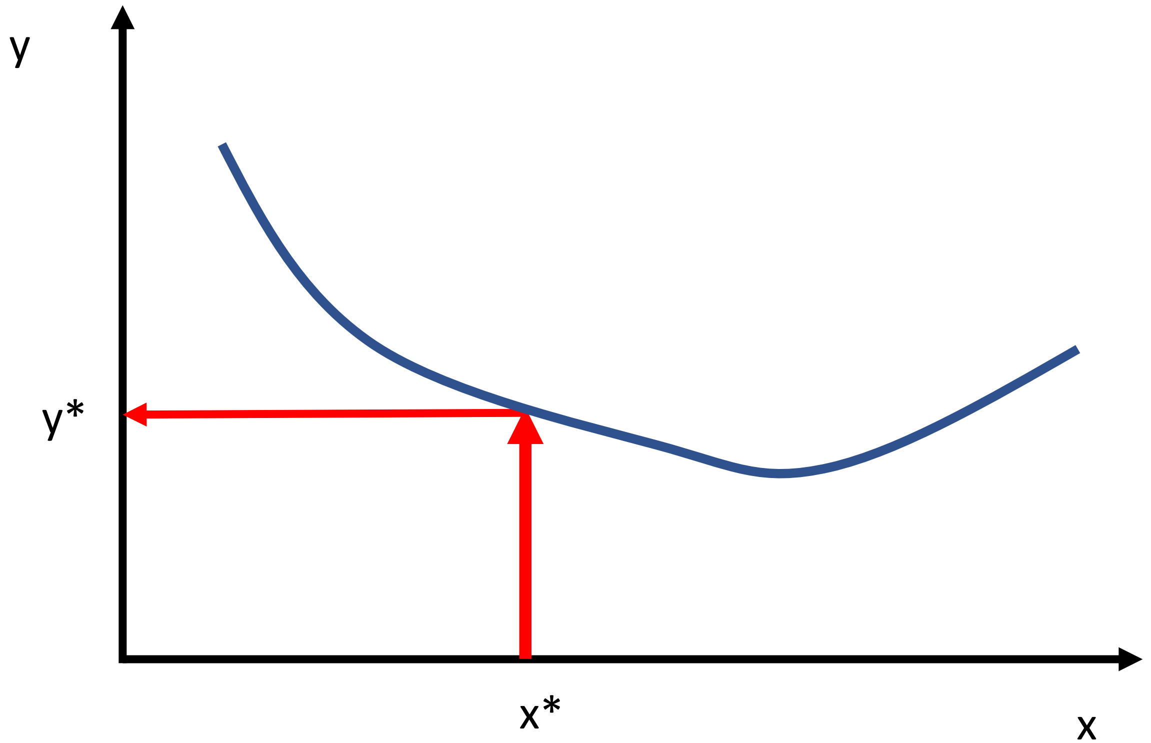
Generalization in Machine Learning

- Generalization ability of a ML model refers to the model's ability to give accurate prediction to new (previously unseen) data

- Assumptions:
  - Unseen (test) data is drawn from the same distributions (have the same properties) as the training data
  - ML models having high accuracy in the training data (low empirical loss) are expected to be accurate on the test data (low expected loss)

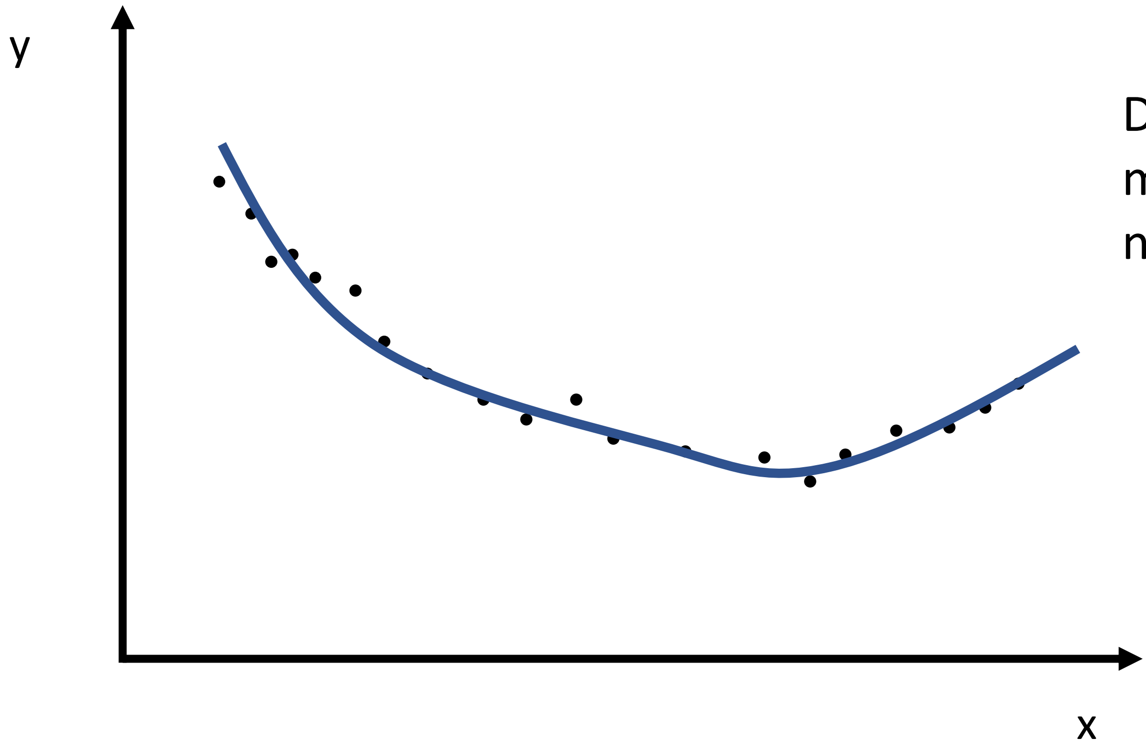
Generalization in Machine Learning

- Let us assume that there is a process (function) connecting x to y



Generalization in Machine Learning

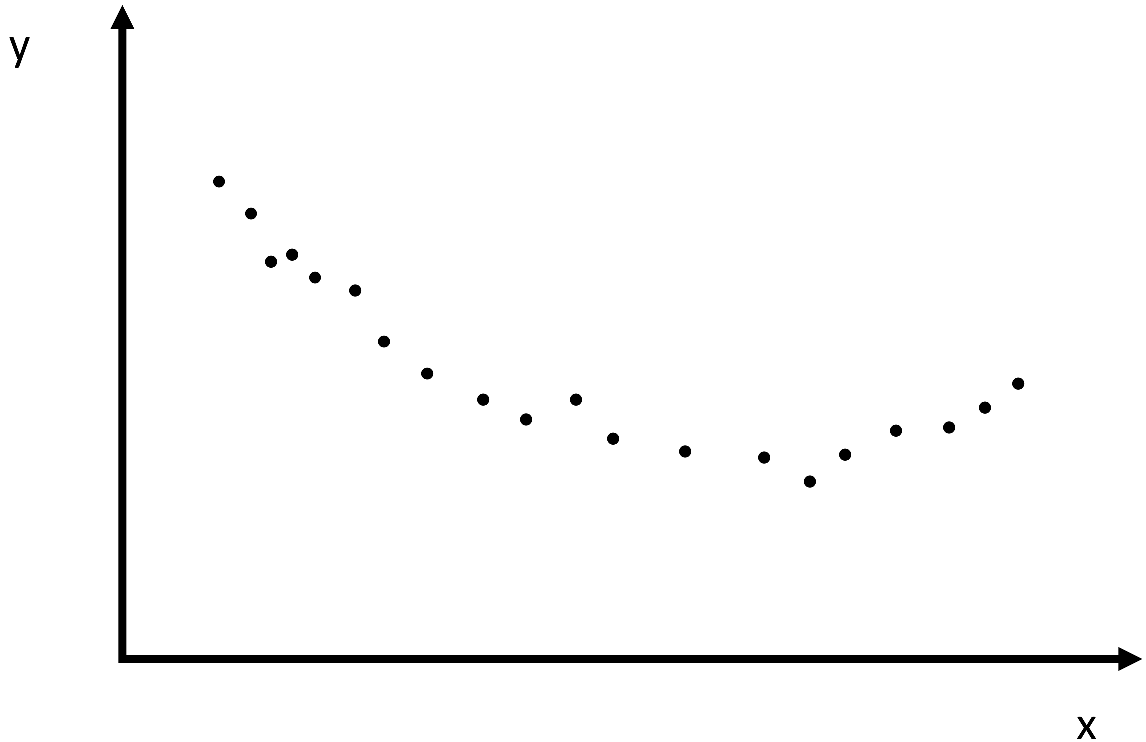
- In practice, we do not have an expression for this process. Instead we get a set of data $\{x_i, y_i\}$, i=1,…,N



Data can be noisy or measurements may not be accurate!

Generalization in Machine Learning

- In practice, we do not have an expression for this process. Instead we get a set of data $\{x_i, y_i\}$, $i=1,\ldots,N$

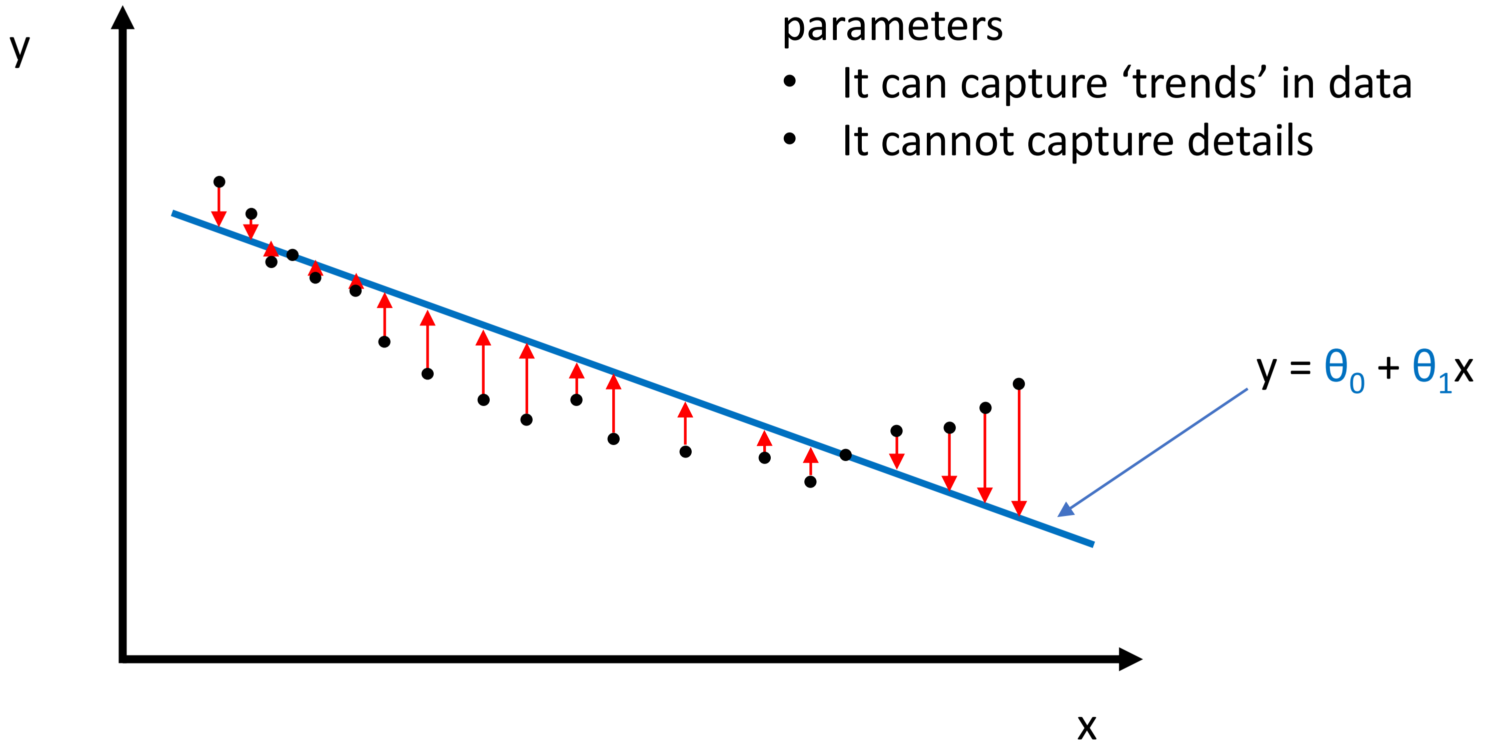


Underfitting vs. Overfitting

- 1D regression example: Given x_i, i=1,…,20, predict y_i

Underfitting: the model has too few parameters

- It can capture 'trends' in data
- It cannot capture details

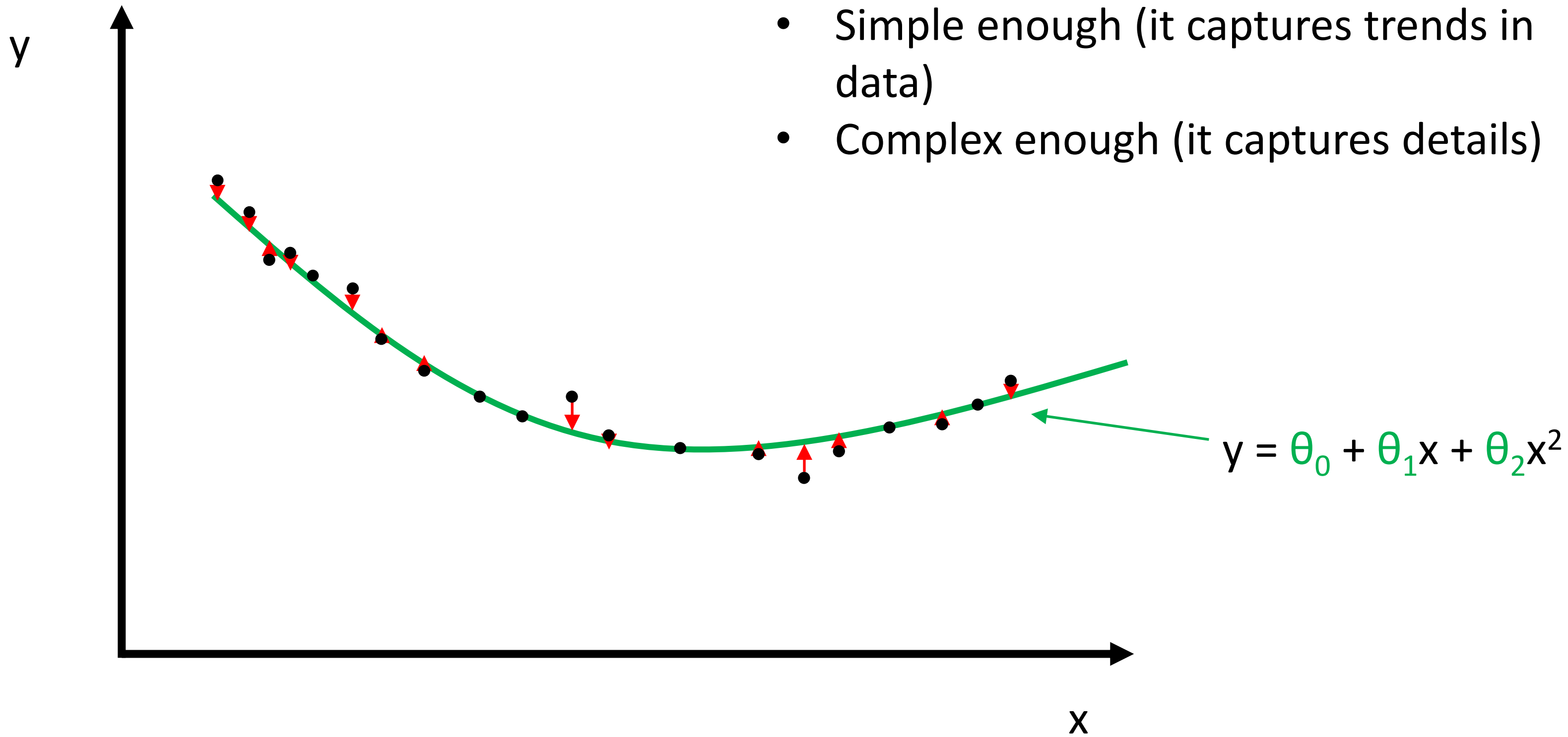


Underfitting vs. Overfitting

- 1D regression example: Given x_i, i=1,…,20, predict y_i

Good fit means the model is:

- Simple enough (it captures trends in data)
- Complex enough (it captures details)

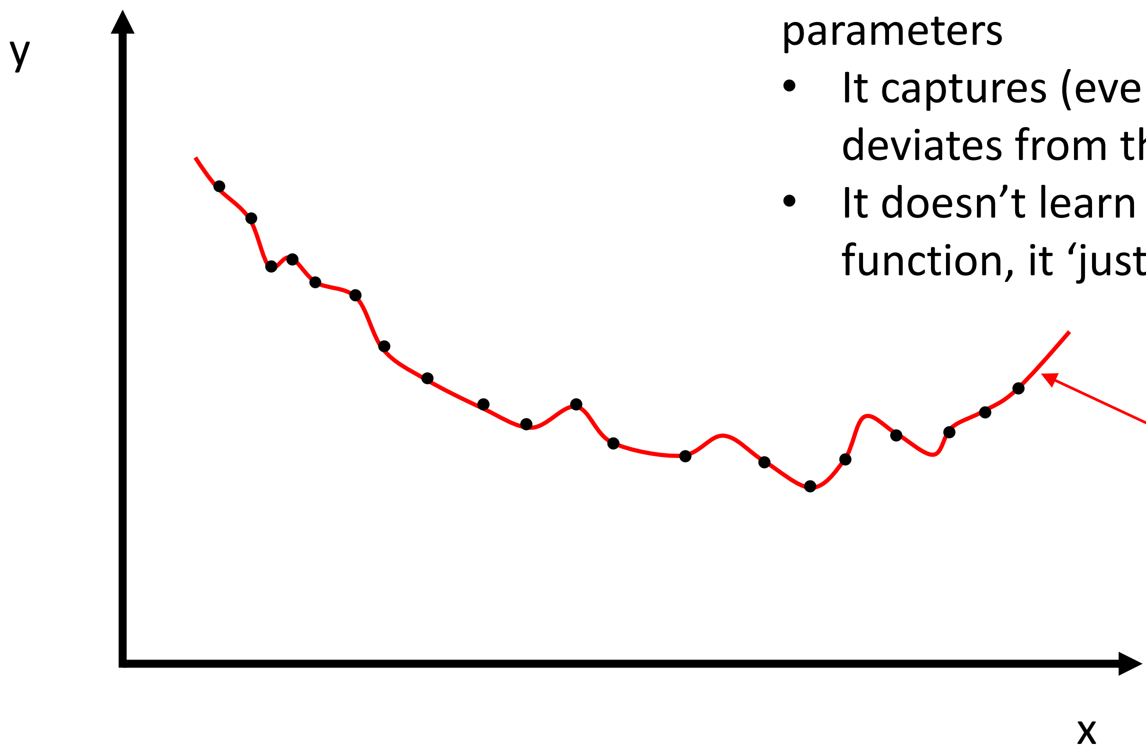


Underfitting vs. Overfitting

- 1D regression example: Given x_i, i=1,…,20, predict y_i

Overfitting: the model has too many parameters

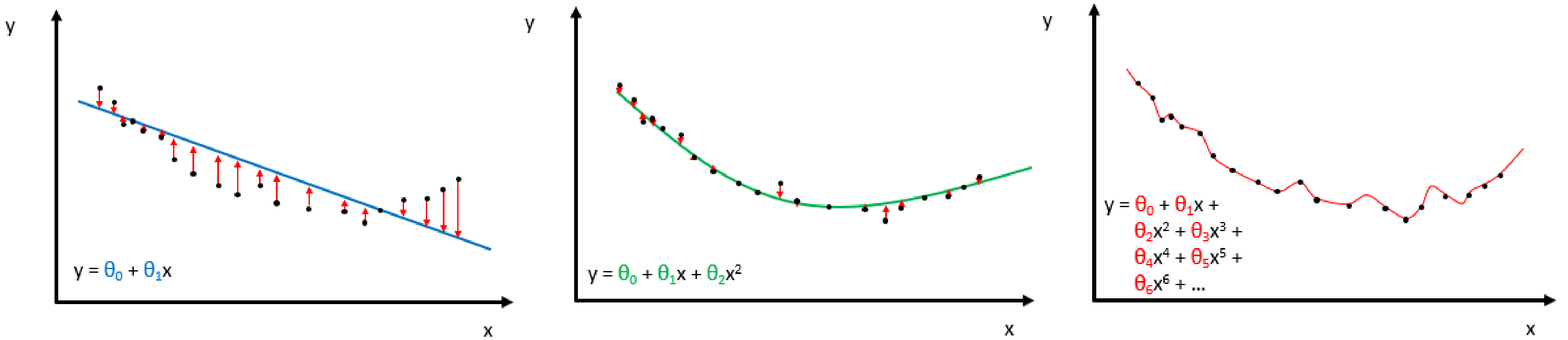
- It captures (even small) details and deviates from the global trends
- It doesn't learn the underlying function, it 'just memorizes' the data



$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6 + \dots$$

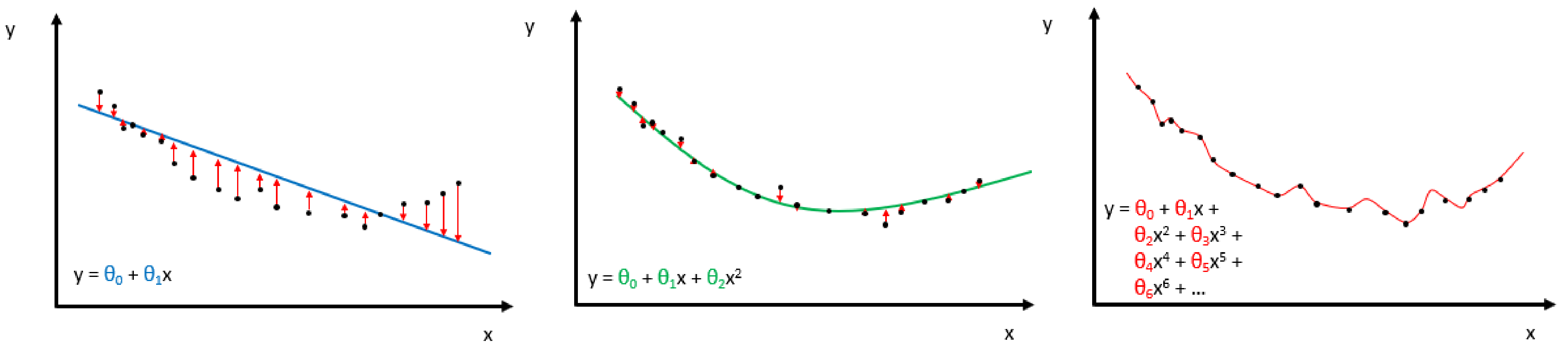
VC dimension

- The ‘capacity’ of a model is the maximum number of samples it can memorize (irrespectively their arrangement). We call this number Vapnik-Chevronenkis (VC) dimension



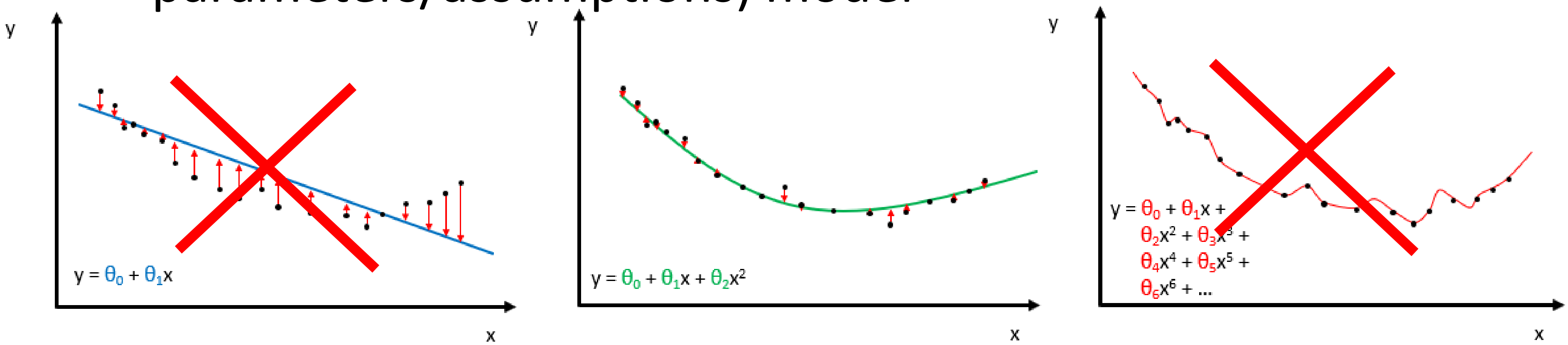
VC dimension

- The ‘capacity’ of a model is the maximum number of samples it can memorize (irrespectively their arrangement). We call this number Vapnik-Chevronenkis (VC) dimension
- When we need to choose between two models having similar empirical losses, we favor the one with lower VC dimension



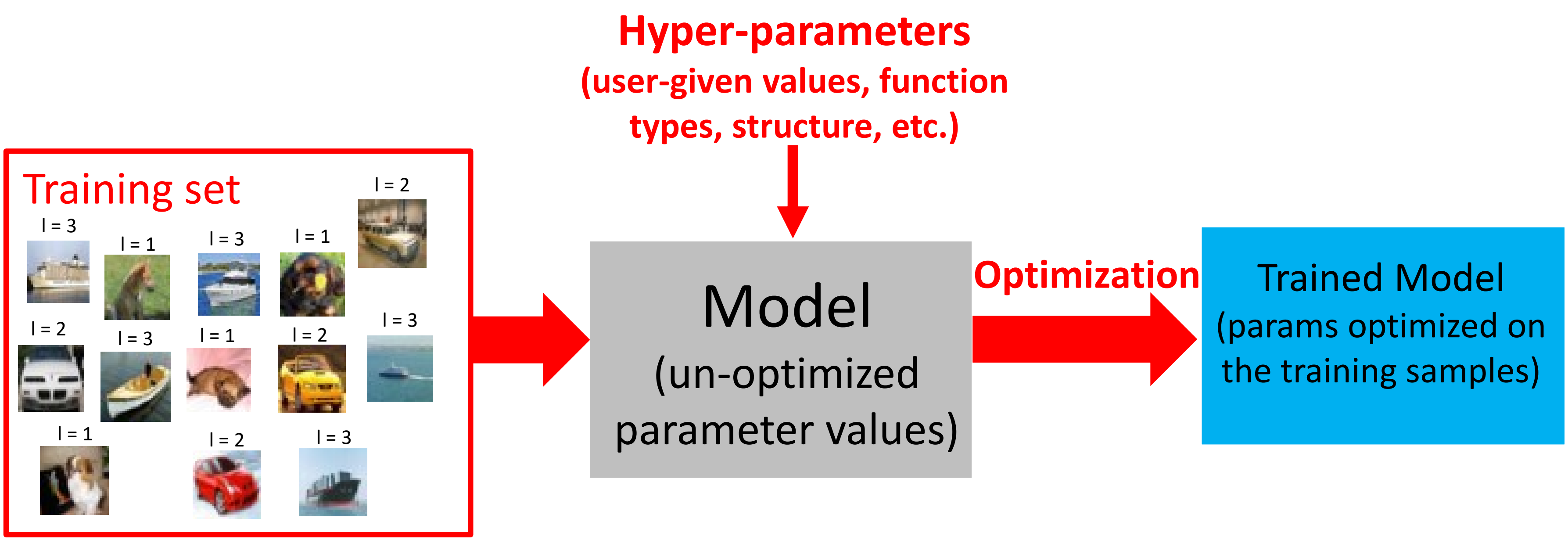
VC dimension

- The ‘capacity’ of a model is the maximum number of samples it can memorize (irrespectively their arrangement). We call this number Vapnik-Chevronenkis (VC) dimension
- When we need to choose between two models having similar empirical losses, we favor the one with lower VC dimension
- Occam’s razor: select the simpler (requiring the fewer parameters/assumptions) model



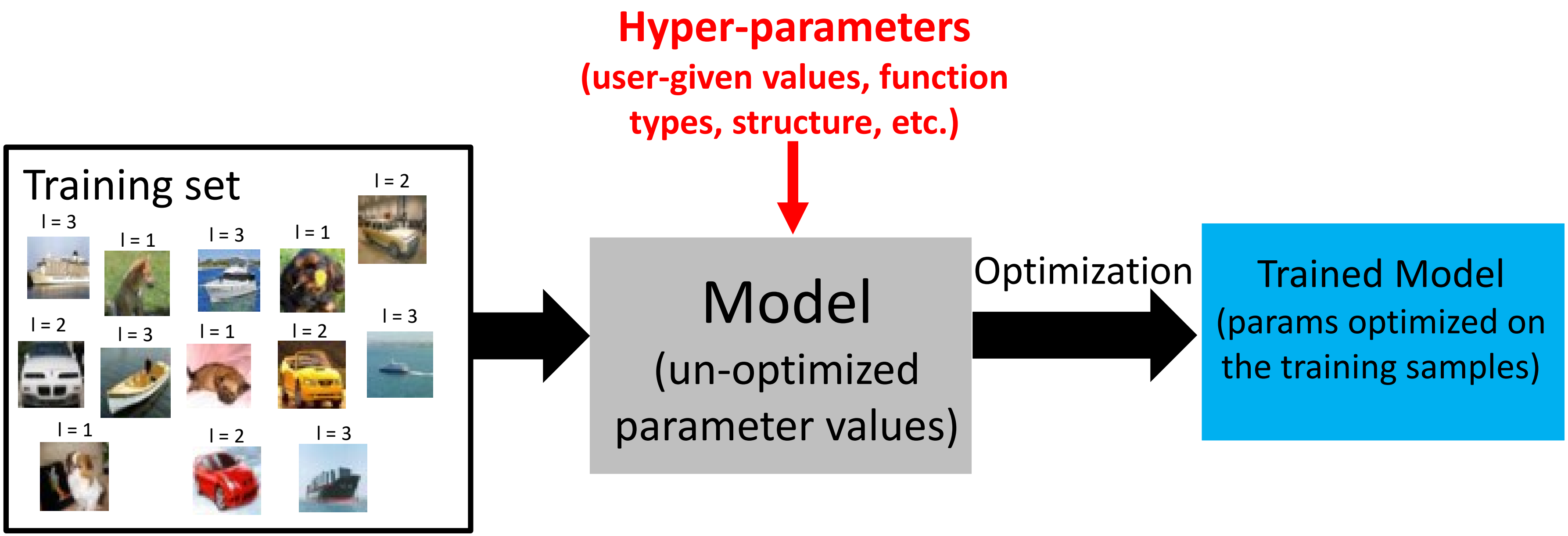
Factors affecting the training process

- Hyper-parameter value selection
- Stopping criteria in iterative optimization
- Regularization
- Training set size



Factors affecting the training process

- Hyper-parameter value selection
- Stopping criteria in iterative optimization
- Regularization
- Training set size



Insert Web Page

This app allows you to insert secure web pages starting with https:// into the slide deck. Non-secure web pages are not supported for security reasons.

Please enter the URL below.

https://	cs.stanford.edu/~karpathy/svmjs/demo/

Note: Many popular websites allow secure access. Please click on the preview button to ensure the web page is accessible.

Preview

Demo from Andrej Karpathy at Stanford

Insert Web Page

This app allows you to insert secure web pages starting with https:// into the slide deck. Non-secure web pages are not supported for security reasons.

Please enter the URL below.

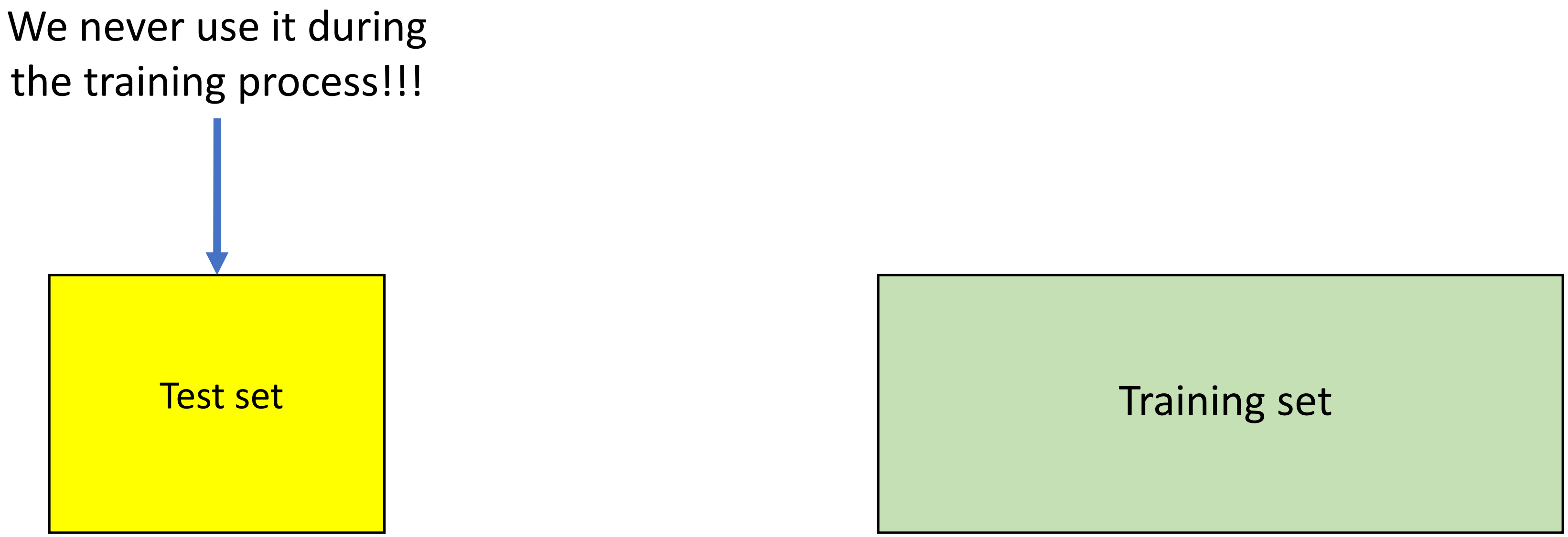
https://	playground.tensorflow.org/#activation=relu&batchSize=10&dataset=xor&regDatas

Note: Many popular websites allow secure access. Please click on the preview button to ensure the web page is accessible.

Preview

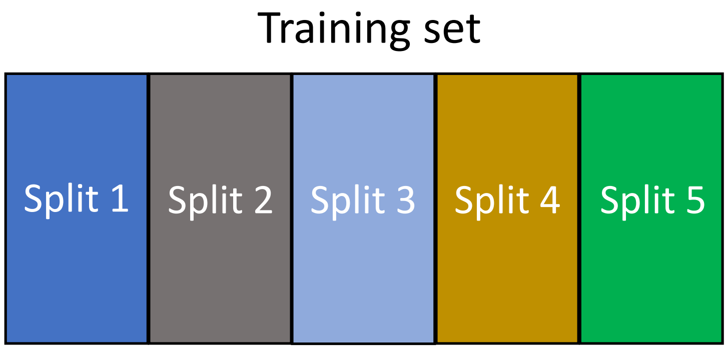
Neural network-based classification (using tensorflow)

Hyper-parameter value selection



Hyper-parameter value selection

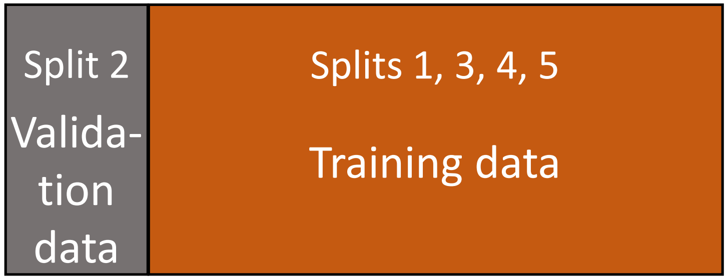
- (Stratified) k-fold cross-validation (e.g. with k=5):
  - Split each class to k sets randomly
  - Use the i^{th} subset of all classes to form the i^{th} dataset split



Hyper-parameter value selection

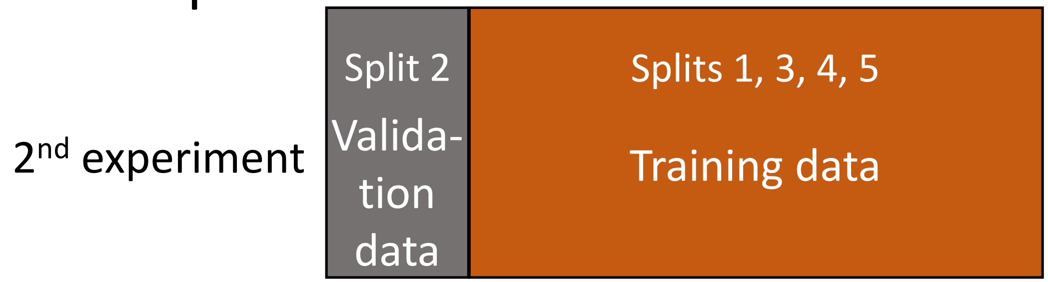
- (Stratified) k-fold cross-validation (e.g. with k=5):
  - Split each class to k sets randomly
  - Use the i^{th} subset of all classes to form the i^{th} dataset split
  - Form k classification problems using the i^{th} dataset split as validation set and the remaining splits as training set

2nd experiment



Hyper-parameter value selection

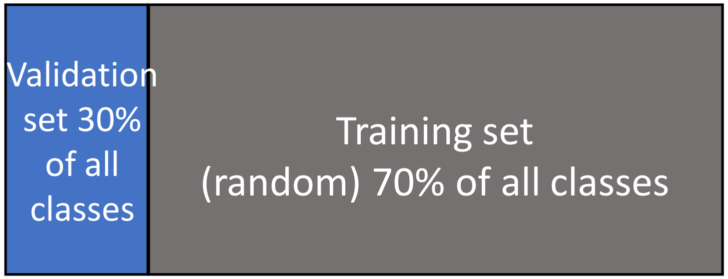
- (Stratified) k-fold cross-validation (e.g. with k=5):
  - Split each class to k sets randomly
  - Use the i^{th} subset of all classes to form the i^{th} dataset split
  - Form k classification problems using the i^{th} dataset split as validation set and the remaining splits as training set
  - Measure the performance of each hyper-parameter value on all k classification problems (if more than one hyper-parameters, use all combinations of them – grid search)
  - Use the hyper-parameter value(s) giving the best average performance over all the k experiments



Hyper-parameter value selection

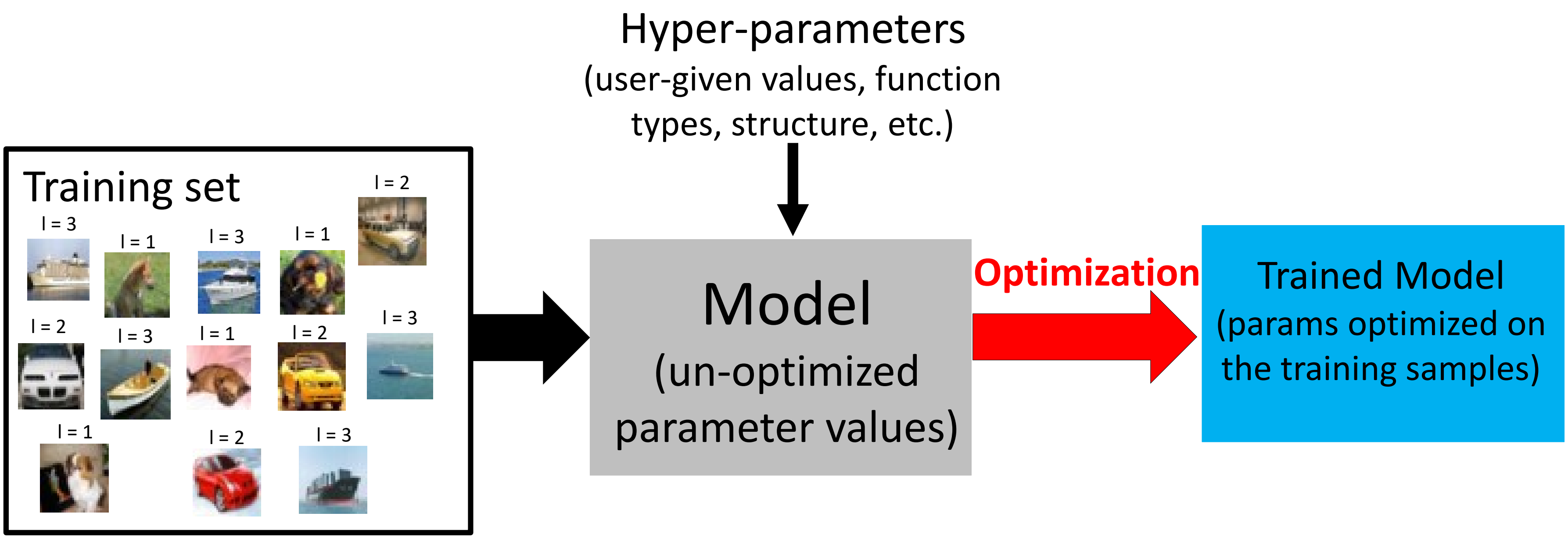
- (Stratified) hold-out set validation:
  - Split each class to, e.g. 70% - 30%, training and validation data randomly and form the (overall) training/validation sets
  - Form k (e.g. with k=5) classification problems using k such random splits
  - Measure the performance for each hyper-parameter value on all k classification problems (if more than one hyper-parameters apply grid search)
  - Use the hyper-parameter value(s) giving the best average performance over all the k experiments

i^{th} experiment

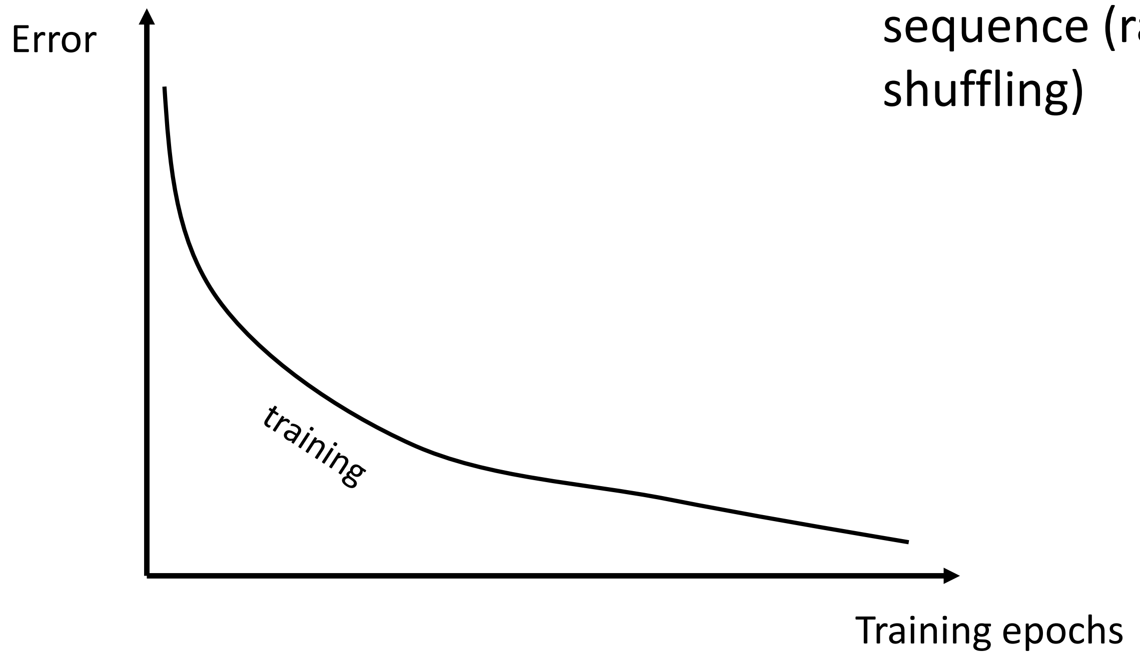


Factors affecting the training process

- Hyper-parameter value selection
- **Stopping criteria in iterative optimization**
- Regularization
- Training set size



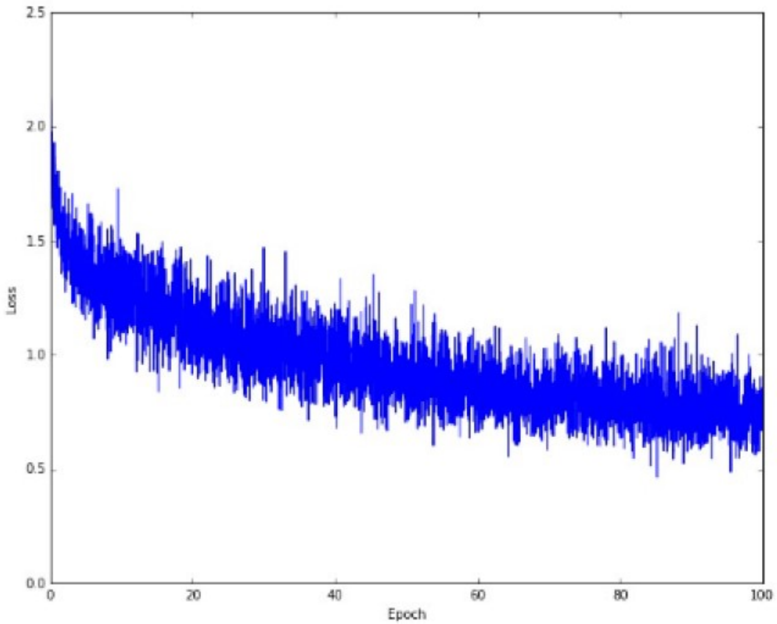
Stopping criteria in iterative optimization



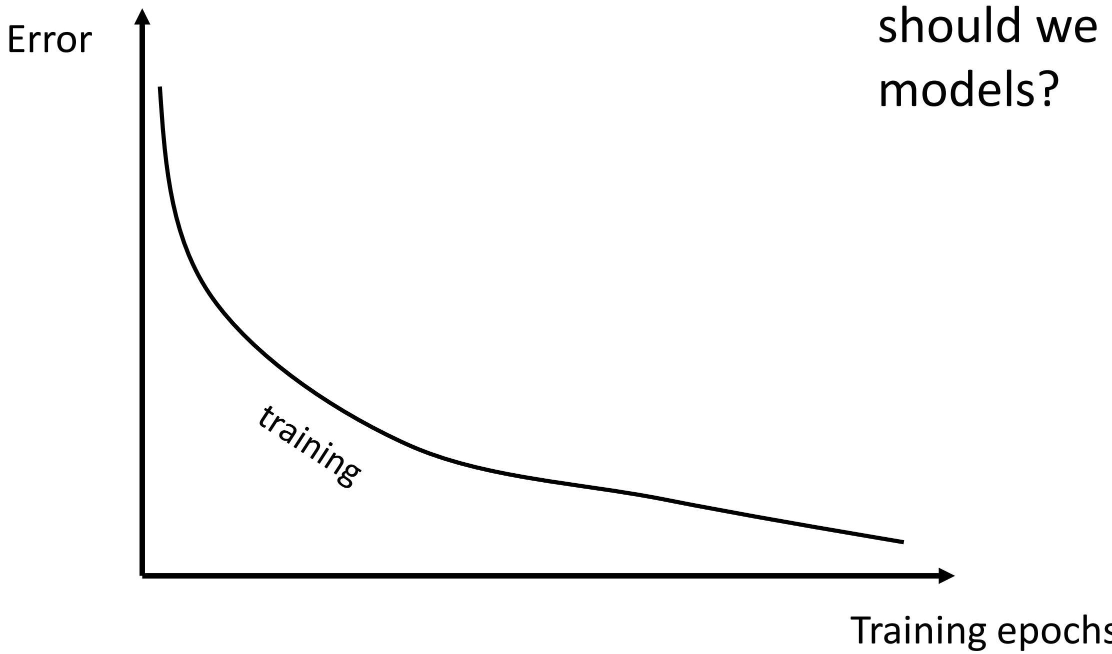
At each epoch samples are introduced in a different sequence (random data shuffling)

Stopping criteria in iterative optimization

In practice, we update the network's parameters on mini-batches of data, and curves are not so good-looking!

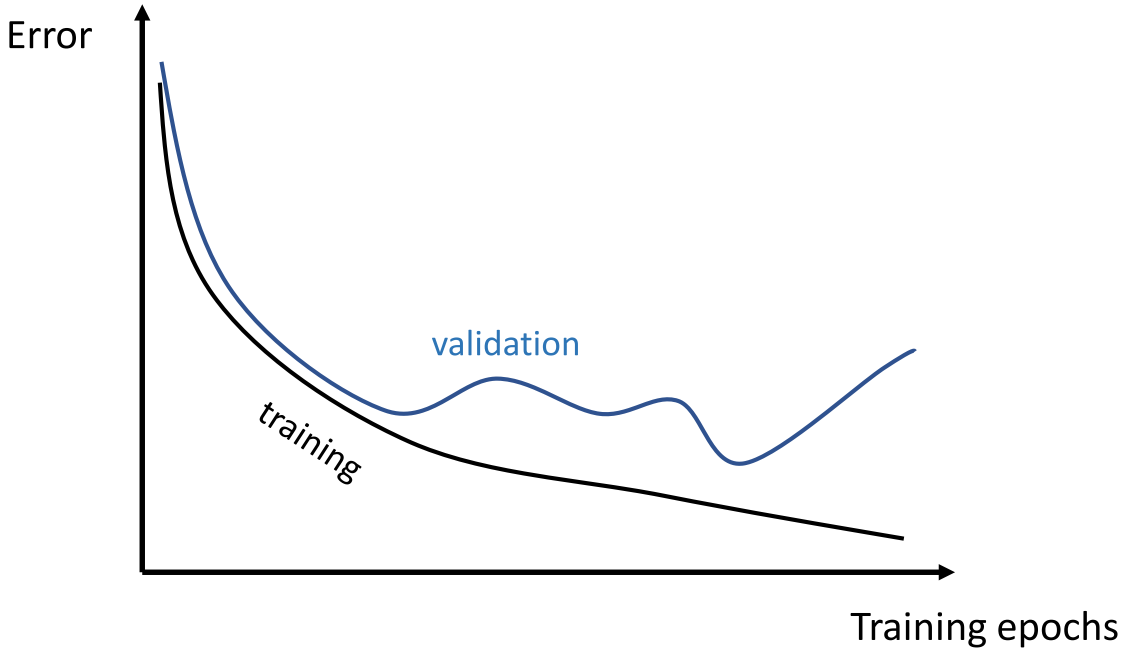


Stopping criteria in iterative optimization



How should we stop the training? Which parameters should we select for the models?

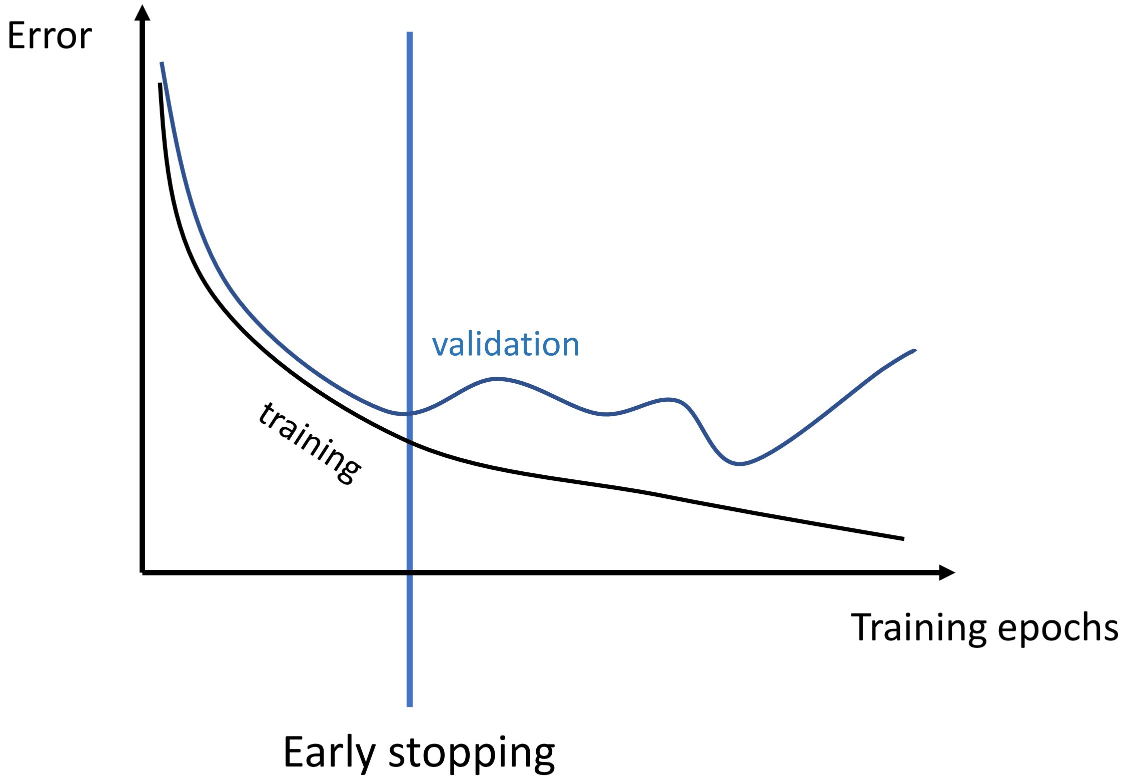
Stopping criteria in iterative optimization



Early stopping

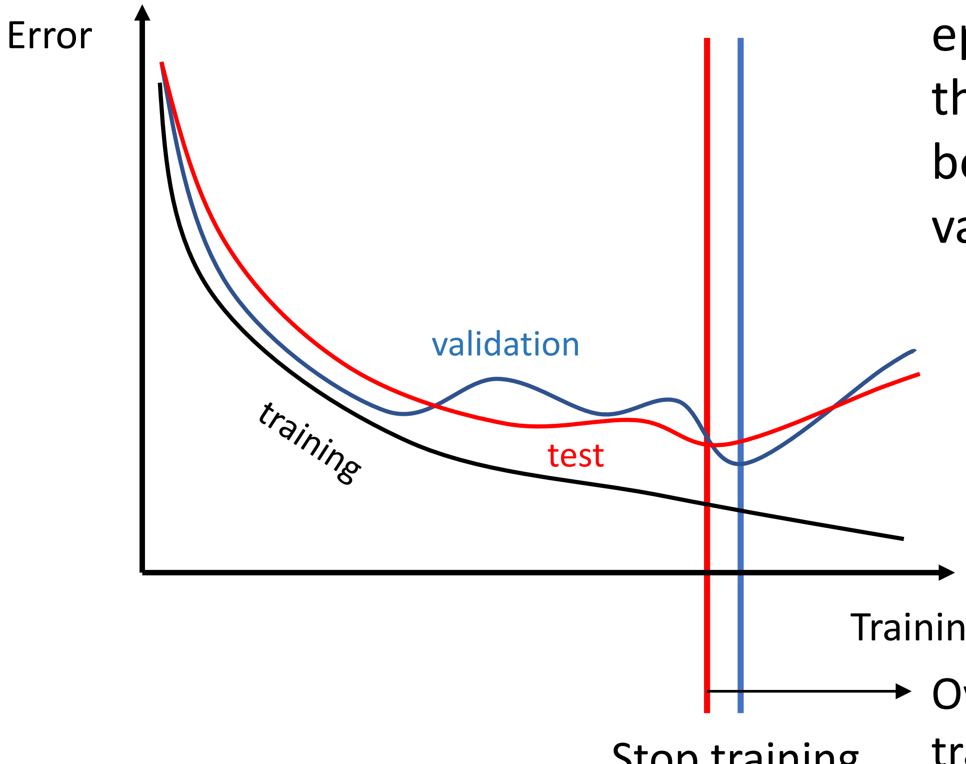
Stopping criteria in iterative optimization

- Early stopping based on hold-out validation set



Stopping criteria in iterative optimization

- Stopping based on hold-out validation set

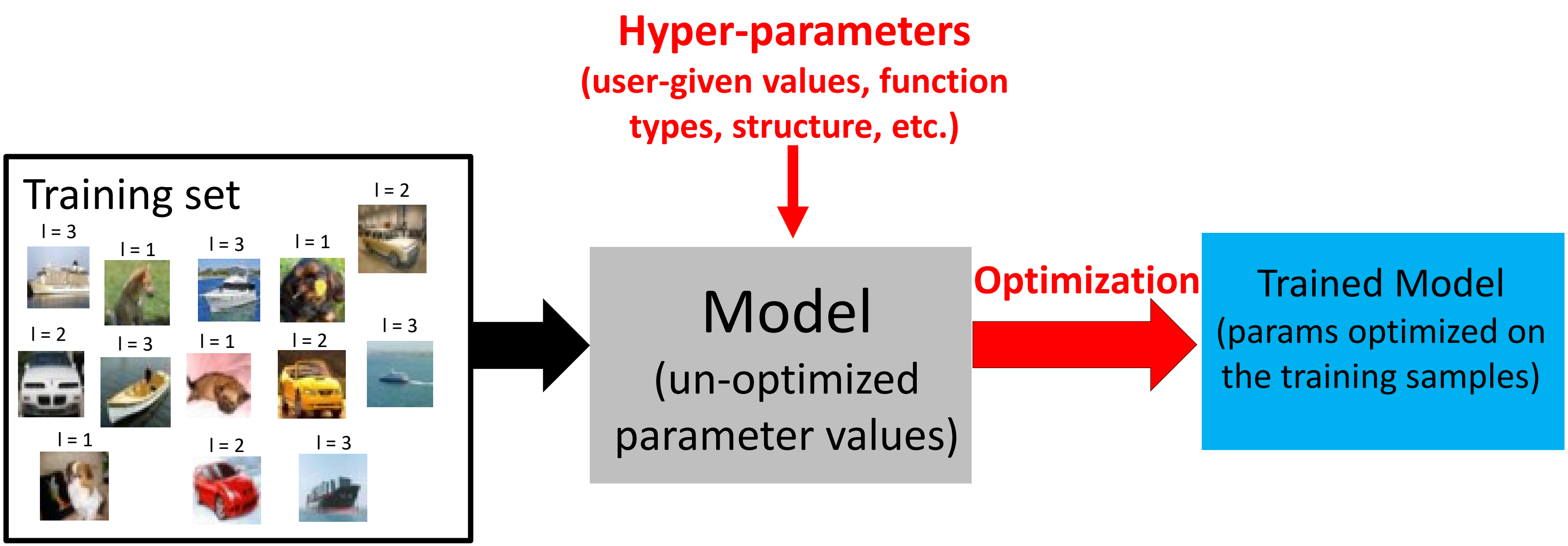


<u>In practice</u>: training is applied for a number of epochs (e.g. 1000) and the model providing the best performance on the validation set is selected

→ Overfitting to the training data

Factors affecting the training process

- Hyper-parameter value selection
- Stopping criteria in iterative optimization
- **Regularization**
- Training set size



Regularization

- Regularization can be seen as a process to enforce properties in the solution of an optimization problem (e.g. low capacity, smoothness)

- When optimizing a function $f(\cdot)$ using training samples x_i (followed by labels y_i)

$$\min_f \underbrace{\sum_{i=1}^{N} loss\left(f(x_i), y_i\right)}_{\text{Average error over the N training samples}} + \lambda R(f)$$

Regularization term

Regularization parameter: relative importance of the two terms ($\lambda > 0$)

Regularization

- When optimizing a linear function $f(x) = \mathbf{w}^T\mathbf{x}$ the regularized optimization problem takes the form:

$$\min_{w} \sum_{i=1}^{N} loss\left(\mathbf{w}^T\mathbf{x}_i, y_i\right) + \lambda \|\mathbf{w}\|_A^B$$

- Widely used regularization terms are:
  - L_2-(Tikhonov) regularization (A = 2, B = 2) forces the values of **w** to be small (leads to smoother decision functions)
  - L_1-regularization (A = 1, B = 1) forces **w** to be sparse, i.e. to have most values equal to zero (leads to simpler and more interpretable solutions)

Regularization

- Regularization is used in models for both:
  - Iterative optimization (e.g. artificial neural networks). In practice, for these models the regularization is imposed explicitly, e.g. by normalizing the network's weight to unit l_2-norm after each update
  - Convex optimization (e.g. regression). It is added as a constraint or as an additional term in the optimization problem

Regularization

- Example: linear regression of $\mathbf{x}_i$ to y_i, i=1,…,N

$$loss\left(\mathbf{w}^T\mathbf{x}_i, y_i\right) = \left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2$$

  Using l_2-regularization:

$$\min_{w} \sum_{i=1}^{N} \left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \lambda\|\mathbf{w}\|_2^2$$

- The solution $\mathbf{w}^*$ of the above problem is:

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

  where $\mathbf{X} = [\mathbf{x}_1,\ldots,\mathbf{x}_N]$ and $\mathbf{y} = [y_1,\ldots,y_N]^T$

Regularization

- Example: linear regression of $\mathbf{x}_i$ to y_i, i=1,…,N

$$loss\left(\mathbf{w}^T\mathbf{x}_i, y_i\right) = \left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2$$

Using l_2-regularization:

$$\min_{w} \sum_{i=1}^{N} \left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \lambda\|\mathbf{w}\|_2^2$$

- The solution $\mathbf{w}^*$ of the above problem is:

$$\mathbf{w}^* = \left(\mathbf{X}\mathbf{X}^T + \underline{\lambda\mathbf{I}}\right)^{-1}\mathbf{X}\mathbf{y}$$

where $\mathbf{X} = [\mathbf{x}_1,\ldots,\mathbf{x}_N]$ and $\mathbf{y} = [y_1,\ldots,y_N]^T$

Remember that in linear regression: $\mathbf{w}^* = \left(\mathbf{X}\mathbf{X}^T\right)^{-1}\mathbf{X}\mathbf{y}$

Thus, the d^{th} dimension of **w** is scaled based on λ

Regularization

- Example: linear regression of $\mathbf{x}_i$ to y_i, i=1,…,N

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

SVD of **X**: $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ (we can also express $\mathbf{I} = \mathbf{U}\mathbf{U}^T$)

then $\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}\mathbf{U}^T = \mathbf{U}\mathbf{S}^2\mathbf{U}^T$

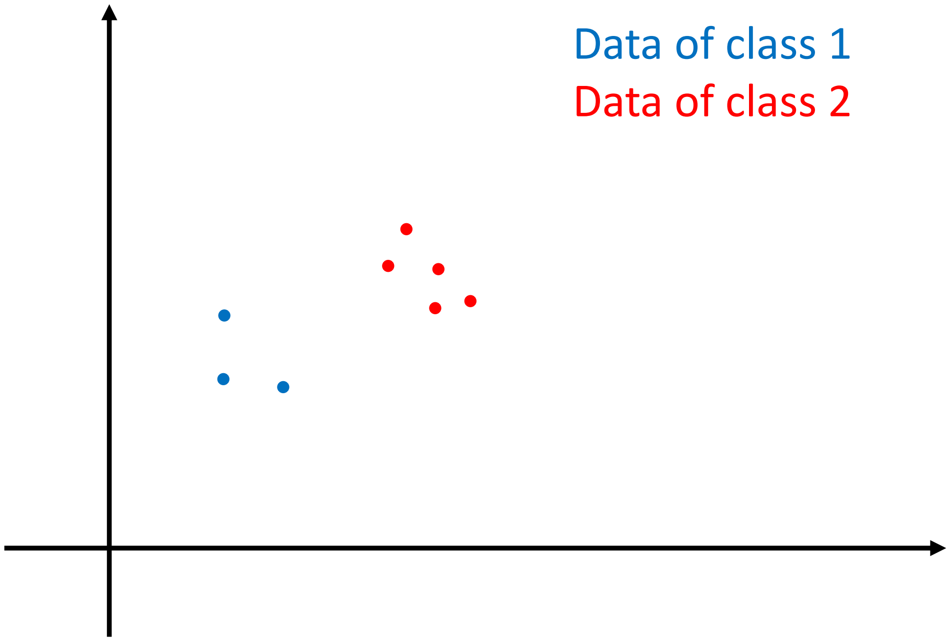
and $\left(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}\right)^{-1} = \mathbf{U}(\mathbf{S}^{-2} + \frac{1}{\lambda}\mathbf{I})\mathbf{U}^T$

$$\mathbf{w}^* = \mathbf{U}(\mathbf{S}^{-2} + \frac{1}{\lambda}\mathbf{I})\mathbf{S}\mathbf{V}^T\mathbf{y}$$

The d^{th} dimension is scaled with $\frac{\sigma_d}{\sigma_d^2 + \lambda}$ instead of σ_d^{-1}

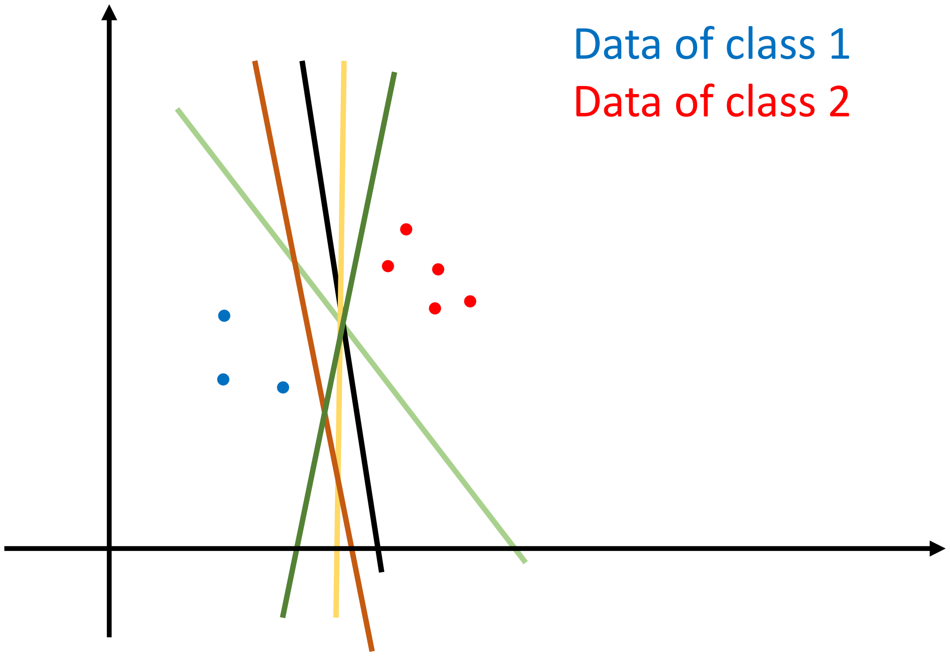
Regularization

- Smoothness and semi-supervised learning:



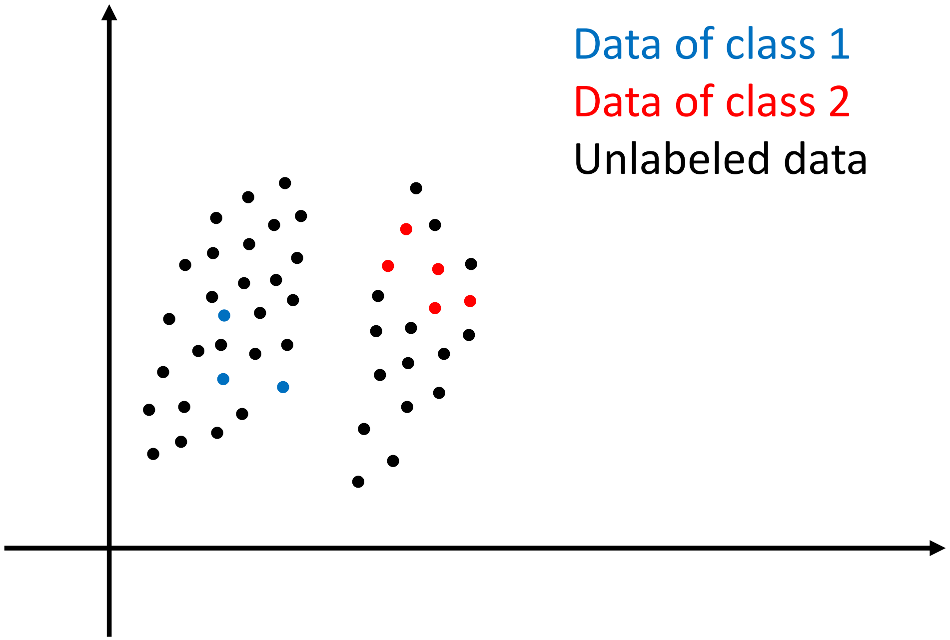
Regularization

- Smoothness and semi-supervised learning:



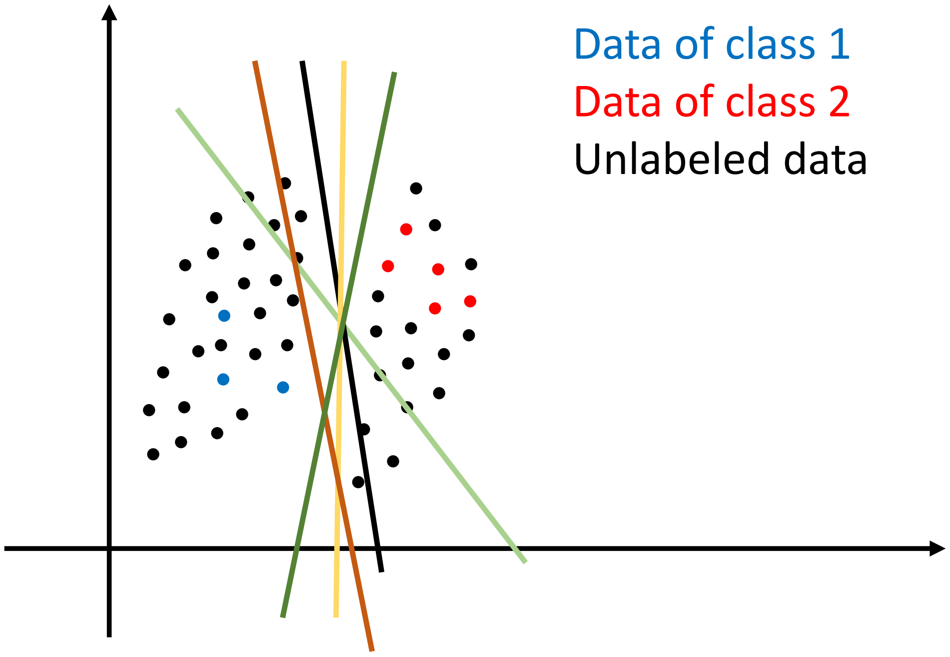
Regularization

- Smoothness and semi-supervised learning:



Regularization

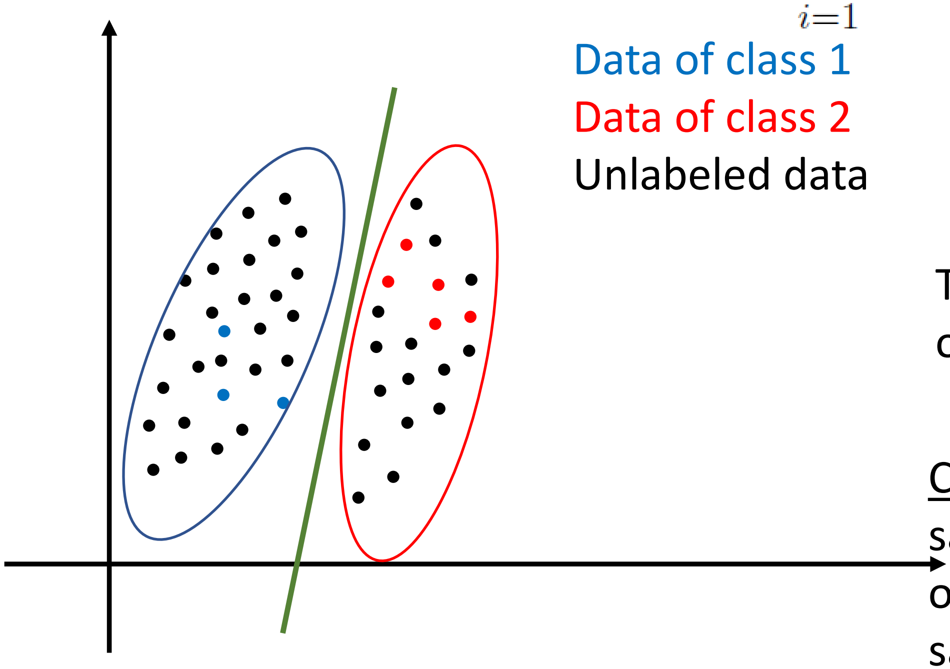
- Smoothness and semi-supervised learning:



Regularization

- Smoothness and semi-supervised learning (SSL):

$$\min_{w} \sum_{i=1}^{N_l} loss\left(\mathbf{w}^T\mathbf{x}_i, y_i\right) + \lambda \mathbf{w}^T\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{w}$$



Data of class 1

Data of class 2

Unlabeled data

The loss is measured only on the labeled data

<u>Cluster assumption in SSL</u>: samples which are close to each other are likely to belong to the same class

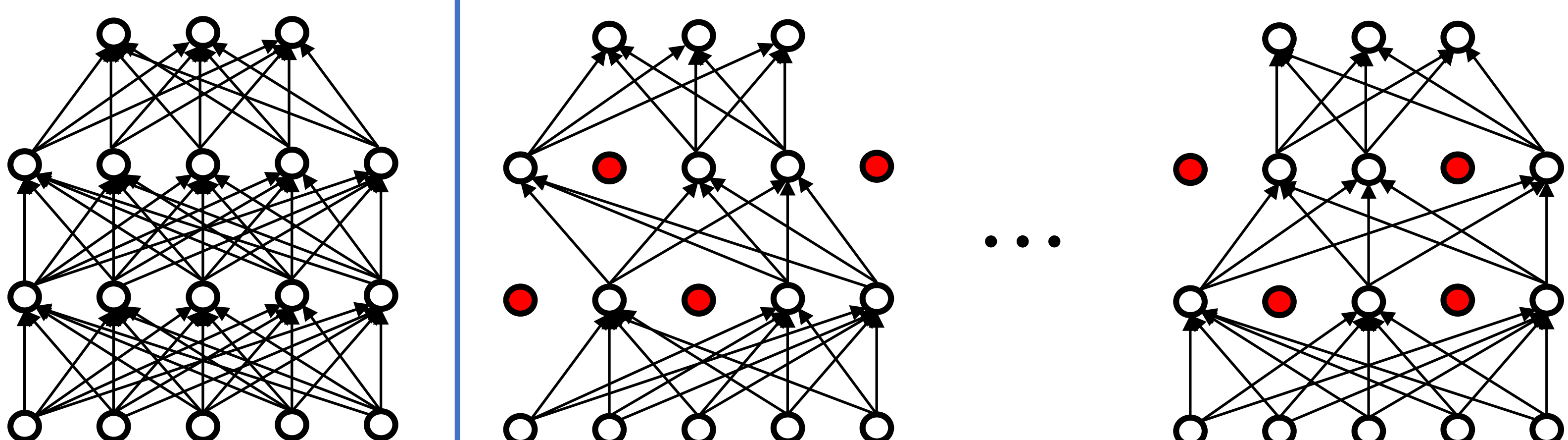
L expresses proximity of data pairs

Regularization

- Dropout in iterative optimization: A probabilistic process to 'augment' the training set during iterative training and increase invariance

Standard neural network training

Dropout-based training
At each iteration, each neuron is active with probability p (using Bernoulli distribution and cut-off value of e.g. p = 0.5)



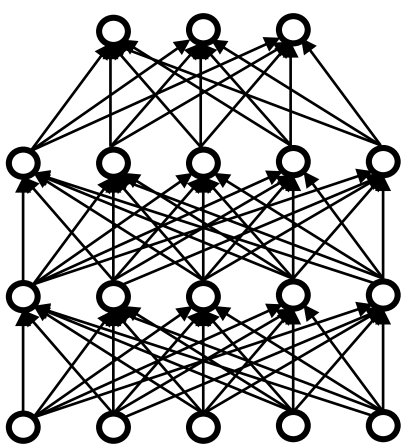
All iterations

Training iteration 1 • • • Training iteration t

Regularization

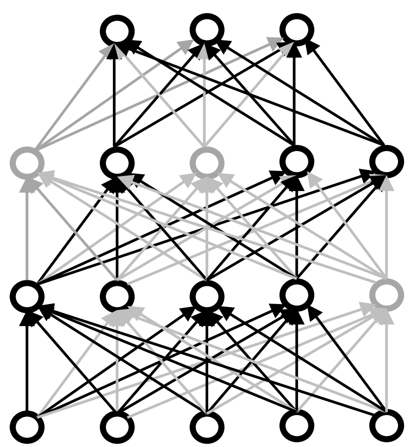
- Continuous Dropout-based iterative optimization

Standard neural network training



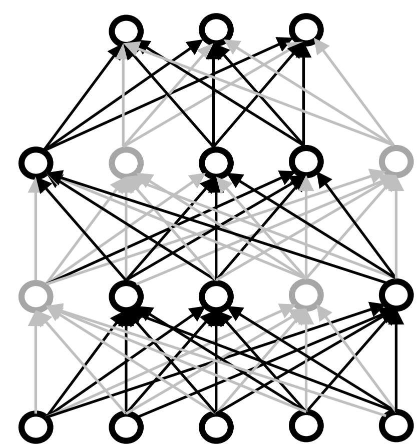
All iterations

Continuous dropout-based training
At each iteration, each neuron is ‘suppressed’ (multiplied) with masks sampled from $\mu \sim U(0, 1)$ or $g \sim N(0.5, \sigma^2)$



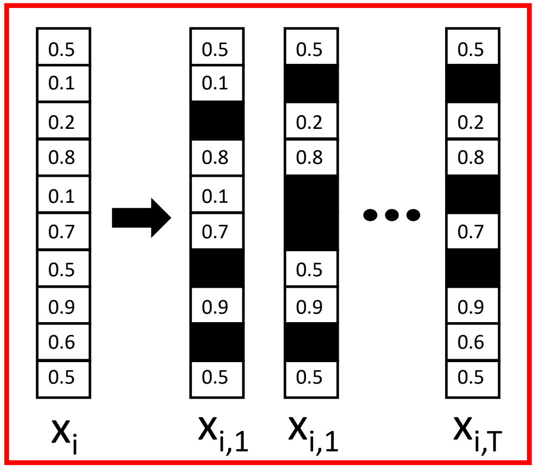
Training iteration 1

• • •



Training iteration t

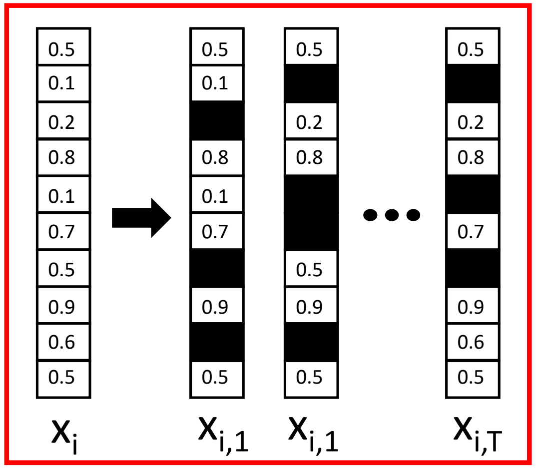
Regularization



- Dropout in convex optimization:
  - We want the response of the model for different (masked) versions of the training data to be very close to that of the original samples: $\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_{i,t}) \rightarrow 0$
  - This means that, if the problem is solved in using an iterative process with T iterations, the regularization term is:

$$R(\mathbf{w}) = \sum_{i=1}^{N} \sum_{t=1}^{T} \|\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_{i,t}\|_2^2$$

Regularization



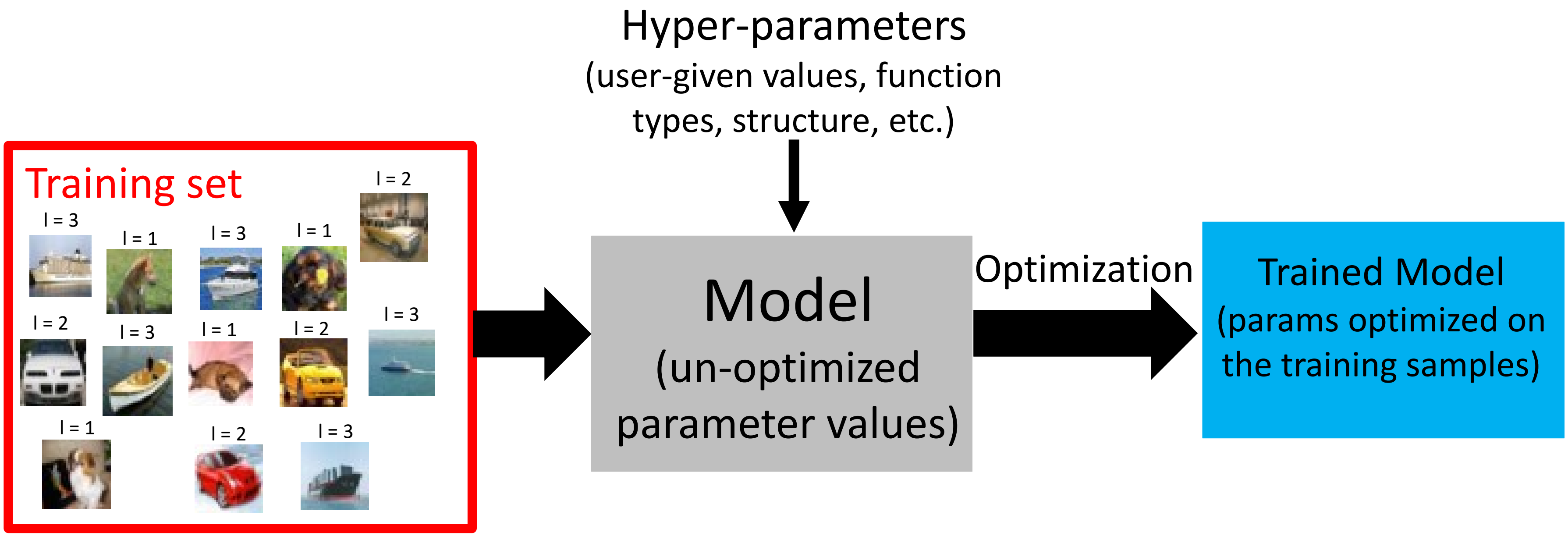
- Dropout in convex optimization:
  - We want the response of the model for different (masked) versions of the training data to be very close to that of the original samples: $\mathbf{w}^T(\mathbf{x}_i - \mathbf{x}_{i,t}) \rightarrow 0$
  - This means that, if the problem is solved in using an iterative process with T iterations, the regularization term is:

$$R(\mathbf{w}) = \sum_{i=1}^{N} \sum_{t=1}^{T} \|\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_{i,t}\|_2^2$$

  - At the limit of T → ∞, R(**w**) becomes $R(\mathbf{w}) = \mathbf{w}^T\left(\mathbf{X}\mathbf{X}^T \circ \mathbf{P}\right)\mathbf{w}$, where $\mathbf{P} = \left[(\mathbf{p}\mathbf{p}^T)\circ(\mathbf{1}\mathbf{1}^T - \mathbf{I})\right] + \left[(\mathbf{p}\mathbf{1}^T)\circ\mathbf{I}\right]$ and p is a vector with values (1-p) and 1 is a vector of ones

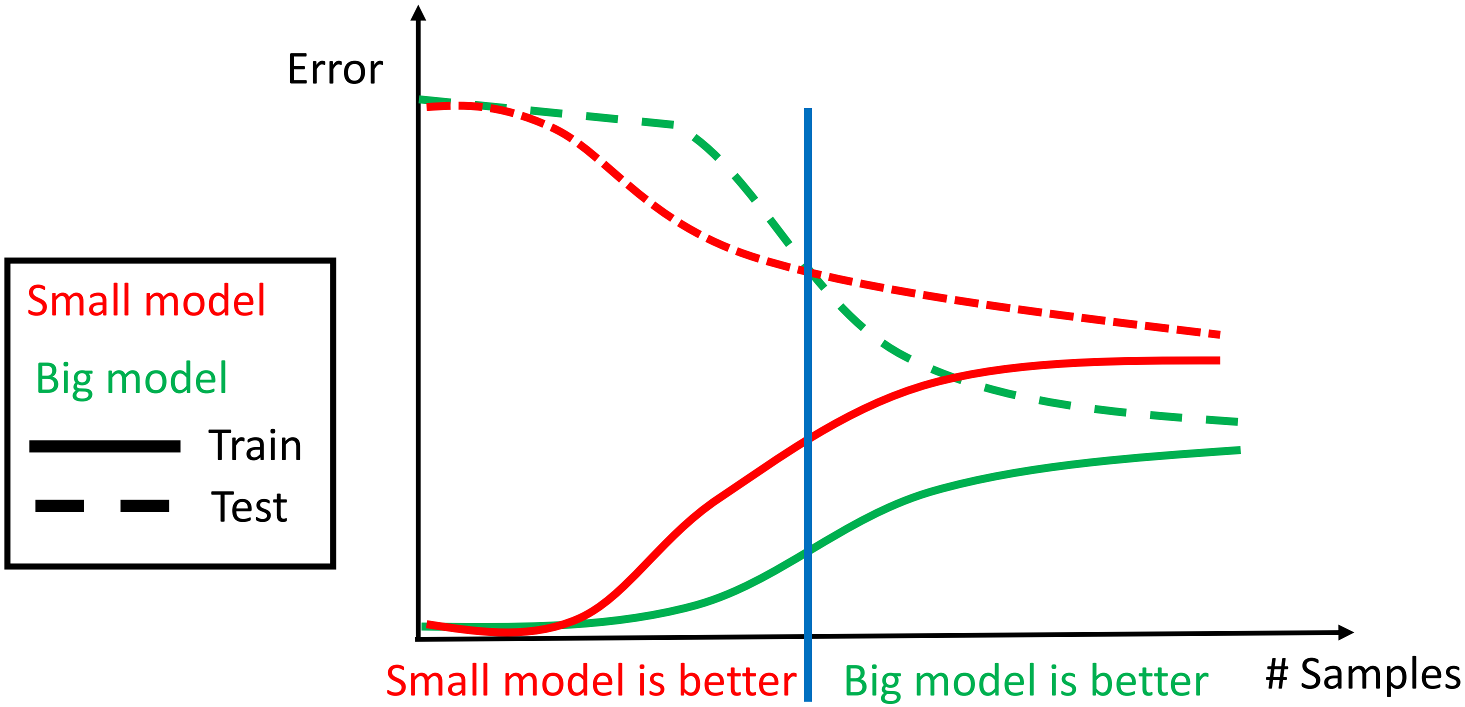
Factors affecting the training process

- Hyper-parameter value selection
- Stopping criteria in iterative optimization
- Regularization
- **Training set size**



Training set size

- When the number of training samples is small (smaller than the number of the model's parameters) the model tends to overfit (under-determined problem)



Training set size

- In neural networks, this problem is usually addressed by using
  - Data augmentation: create new samples by applying small variations on the training data. For example, for images: geometric variations (shift, rotations, scaling), crops, noise
  - Transfer learning: (a) Initialize the model one pre-trained on a big data set (having similar properties to the problem we want to solve) and (b) fine-tuning of the model using the small data set

Training set size

- Pre-trained NN models in Keras

Insert Web Page

This app allows you to insert secure web pages starting with https:// into the slide deck. Non-secure web pages are not supported for security reasons.

Please enter the URL below.

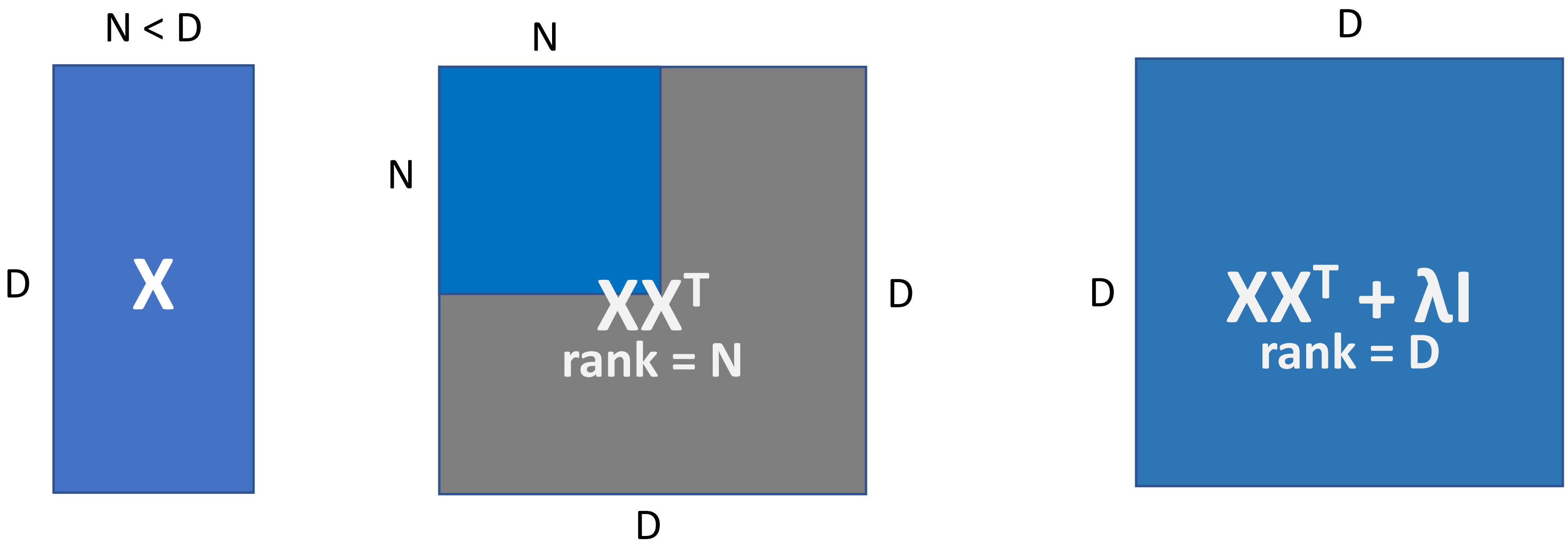
https://	keras.io/applications/

Note: Many popular websites allow secure access. Please click on the preview button to ensure the web page is accessible.

Web Viewer Terms | Privacy & Cookies   Preview

Training set size

- In statistical ML, this is the so-called 'small sample size' problem and it is addressed using regularization



Remember: $\mathbf{w}^* = \left(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{y}$

References

- V. N. Vapnik and A. Chervonenkis, “On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities”, Theory of Probability & Its Applications, vol. 16, no. 2, 1971
- C. M. Bishop, “Pattern recognition and machine learning”, New York: Springer, ISBN 978-0387310732, 2007
- Y. M. Zhang, “Comparative analysis of different cross-validation bandwidth selectors in kernel regression estimators”, International Conference on Machine Learning and Cybernetics, 2013
- C. M. Bishop, “Curvature-driven smoothing: a learning algorithm for feedforward networks”, IEEE Transactions Neural Networks, vol. 4, no. 5, pp. 882–884, 1993
- F. Girosi, M. Jones and T. Poggio, “Regularization theory and neural networks architectures”, Neural Computation, vol. 7, no. 2, pp. 219–269, 1995
- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929-1958, 2014

References

- G. Cao, M. A. Waris, A. Iosifidis and M. Gabbouj, “Multi-modal subspace learning with dropout regularization for cross-modal recognition and retrieval”, International Conference on Image Processing, Tools and Applications, 2016
- A. Iosifidis, A. Tefas and I. Pitas, “DropELM: Fast neural network regularization with Dropout and DropConnect”, Neurocomputing, vol. 162, pp. 57-66, 2015
- X. Shen, X. Tian, T. Liu, F. Xun and D. Tao, “Continuous Dropout”, IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 9, pp. 3926-3937, 2018
- L. Perez and J. Wang, “The effectiveness of data augmentation in image classification using deep learning”, arXiv:1712.04621, 2017
- J. Yosinski, J. Clune, Y. Bengio and H. Lipson, “How transferable are features in deep neural networks”, Neural Information Processing Systems, 2014
- R. Huang, Q. Liu, H. Lu and S. Ma, “Solving the small sample size problem of LDA”, Object recognition supported by user interaction for service robots, 2002