Learning and Generalization in linear and non-linear models

Finn Årup Nielsen

Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark

Email: fn@imm.dtu.dk
WWW: http://www.imm.dtu.dk/~fn

OVERVIEW

- Models
  - Linear and Non-linear models
- Learning
  - Optimization, iterative
    - Gradient
    - Hessian
  - Probability based learning
    - Regression
    - Classification
- Generalization
  - Learning curve
  - Bias variance
  - Regularization
  - Pruning
  - Committee of networks
  - Generalization assessment
    - Validation
    - Complexity criteria

MODEL

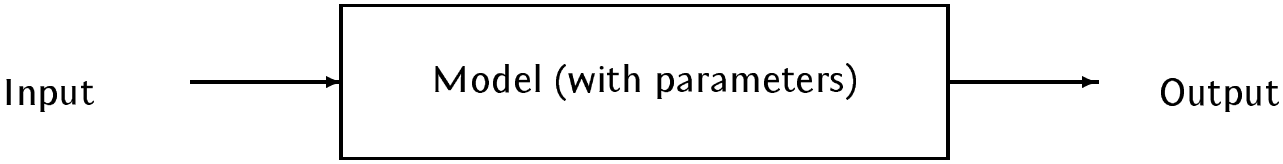

Figure 1: Input/output model.

- A (mathematical) model is relation between a set of observables and parameters, e.g., a simple linear model

$$y = \sum_{i=1}^{d} w_i \, x_i \tag{1}$$

- A statistical model incorporate stochastic elements, which can be characterized by a probability distribution, e.g, a simple linear model with noise

$$t = y + \epsilon = \sum_{i=1}^{d} w_i \, x_i + \epsilon, \tag{2}$$

  where $\epsilon \sim p(\epsilon)$.

- Input, output, target, noise

LINEAR AND NON-LINEAR MODELS

Parameters / Input	Linear	Nonlinear
Linear	GLM with only linear terms	RBF
Nonlinear	Gamma convolution model	Neural network

Table 1: Parameter and input/output linear and nonlinear models. Partly from (Larsen 1996, table 1.1).

- Linear/linear, e.g., the general linear model (GLM)

$$\mathbf{Y} = \mathbf{XB} + \mathbf{U}. \tag{3}$$

- Linear/nonlinear, e.g., radial basis function networks with fixed basis functions (Bishop 1995, eq. 5.14), "generalized additive model"

$$y_k = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}) \tag{4}$$

- Nonlinear/linear

$$y = \sum_{i=1}^{M} \exp(\beta_i) x_i \tag{5}$$

- Nonlinear/nonlinear, e.g., two-layer neural network (Bishop 1995, eq. 4.7), where g is nonlinear

$$y = \tilde{g}\left[\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right] \tag{6}$$

LEARNING

- (Bishop 1995, p. 10) distinguishes between
  - Supervized, involving a target, e.g., regression.
  - Unsupervized, e.g., probability density estimation
  - Reinforcement, target not known, but cost function is
- Define a *cost function* that is large when the discrepancy between the model out and the target is large.
- Batch/online
  - Batch, all examples are used in the optimization.
  - Online, one pattern at a time a "window" (some patterns are used). Stochastic, might escape, learning rate should be decreased as more examples have been presented.

OPTIMIZATION

Continuous valued smooth multidimensional functions with no constraints

- Taylor expansion of cost function around $\hat{\mathbf{w}}$ (Bishop 1995, eq. 7.6)

$$E(\mathbf{w}) = E(\hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})^{\mathsf{T}} \mathbf{b} + \frac{1}{2} (\mathbf{w} - \hat{\mathbf{w}})^{\mathsf{T}} \mathbf{H} (\mathbf{w} - \hat{\mathbf{w}}) + \ldots, \tag{7}$$

  where the gradient and Hessian is defined as

$$(\mathbf{b})_i \equiv \left. \frac{\partial E}{\partial w_i} \right|_{\hat{\mathbf{w}}} \tag{8}$$

$$(\mathbf{H})_{ij} \equiv \left. \frac{\partial E}{\partial w_i \, \partial w_j} \right|_{\hat{\mathbf{w}}} \tag{9}$$

- Appropriate when the cost function is *smooth*.
- Minimum of the cost function is at a stationary point

$$\left. \frac{\partial E}{\partial \mathbf{w}} \right|_{\mathbf{w}^*} = \mathbf{0}. \tag{10}$$

- Optimization by iterations τ

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \Delta \mathbf{w}^{(\tau)} \tag{11}$$

OPTIMIZATION — GRADIENT BASED

- Gradient descent (steepest descent, for neural networks: *backprobagation*) (Bishop 1995, sect. 7.5)

$$\Delta \mathbf{w}^{(\tau)} = \eta \left. \frac{\partial E}{\partial \mathbf{w}} \right|_{\mathbf{w}^{(\tau)}} \tag{12}$$

  Increase step size η if successful descrease of cost function, decrease if not.

- Gradient descent with momentum

$$\Delta \mathbf{w}^{(\tau)} = \eta \left. \frac{\partial E}{\partial \mathbf{w}} \right|_{\mathbf{w}^{(\tau)}} + \mu \Delta \mathbf{w}^{\tau - 1} \tag{13}$$

OPTIMIZATION — HESSIAN BASED

- Quadratic approximation, present point $\hat{\mathbf{w}}$ and optimal point $\mathbf{w}$ (Bishop 1995, eq 7.90)

$$\frac{\partial E}{\partial \mathbf{w}} = \mathbf{0} = \mathbf{b} + \mathbf{H}\left(\mathbf{w} - \hat{\mathbf{w}}\right) \tag{14}$$

$$\mathbf{w} = \hat{\mathbf{w}} - \mathbf{H}^{-1}\mathbf{b} \tag{15}$$

  "Newton method". Hessian not necessarily positive definite: Uphill step to maximum or saddle point.

- Make the Hessian positive definite:

$$\tilde{\mathbf{H}} = \mathbf{H} + \lambda \mathbf{I} \tag{16}$$

  with λ with a larger magnitude than the smallest negative eigenvalue of $\mathbf{H}$. This approximates the negative gradient as $\lambda \to \infty$

$$\Delta \mathbf{w} = -\left(\mathbf{H}^{-1} + \lambda \mathbf{I}\right)^{-1} \mathbf{b} = -\frac{1}{\lambda}\mathbf{b} \tag{17}$$

PROBABILISTIC-BASED LEARNING

- Establish the probability density function for the stochastic element(s) in the model: $p(\mathbf{t}|\mathbf{x}, \mathbf{w})$
- Fix the data: The *likelihood*: $\mathcal{L} = p(\mathbf{t}|\mathbf{x}, \mathbf{w})$
- Cost funtion as the negative log-likelihood

$$E = -\ln \mathcal{L} \tag{18}$$

- If the patterns are independent (Bishop 1995, eq. 6.5)

$$E = -\ln \prod_{n=1}^{N} p(\mathbf{t}^n|\mathbf{x}^n) = -\sum_{n=1}^{N} \ln p(\mathbf{t}^n|\mathbf{x}^n) \tag{19}$$

- Maximum a posteriori (MAP). Likelihood augmented with a prior on the weights

$$E_{\text{MAP}} = -\ln \prod_{n=1}^{N} \left[p(\mathbf{t}|\mathbf{x}, \mathbf{w})\ p(\mathbf{w})\right] \tag{20}$$

$$E_{\text{MAP}} = -\sum_{n=1}^{N} \ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}) + \ln p(\mathbf{w}) \tag{21}$$

PROB. LEARNING — REGRESSION

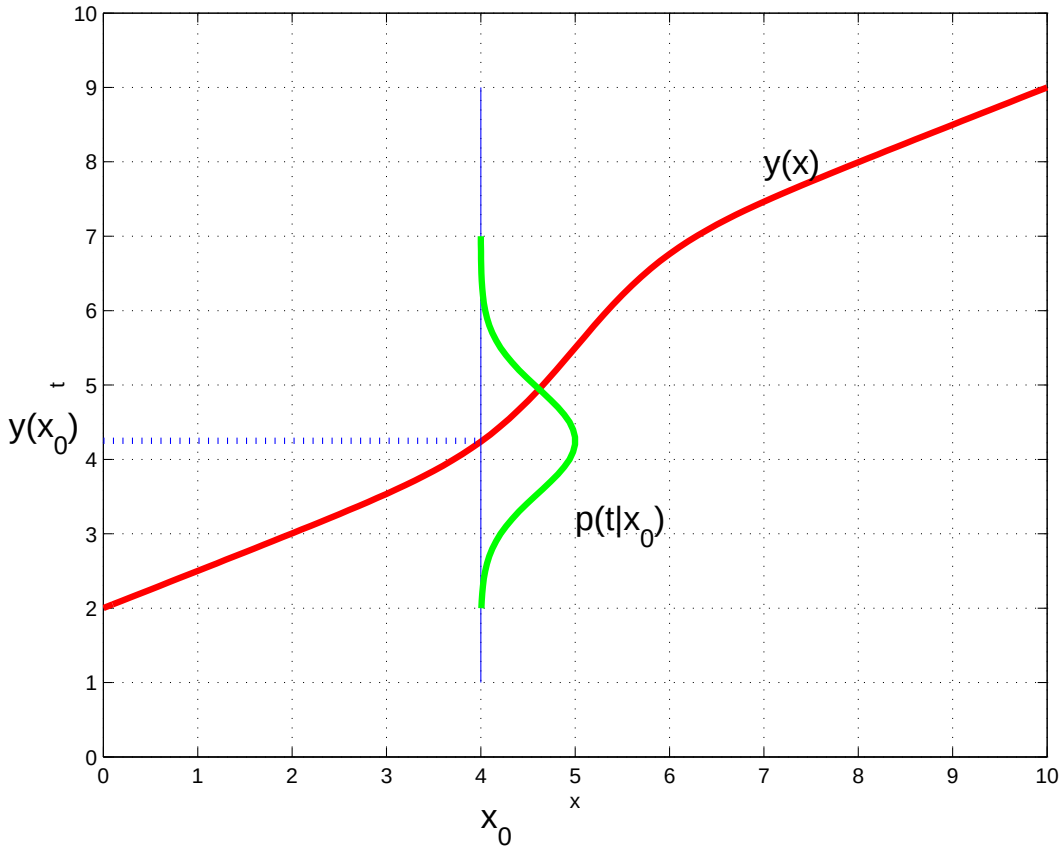

Figure 2: Distribution of $p(t|x)$ (Bishop 1995, figure 6.1): y should be "sufficiently general", optimized completely and N should be large.

- Gaussian error for the noise $\epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$

$$p(t_k|\mathbf{x}) = \left(2\pi\sigma^2\right)^{-\frac{1}{2}} \exp\left[-\frac{\left(y_k(\mathbf{x};\mathbf{t}) - t_k\right)^2}{2\sigma^2}\right] \quad (22)$$

Cost function for all outputs and all patterns

$$E = \frac{1}{2\sigma^2} \underbrace{\sum_n^N \sum_{k=1}^c (y_k^n - t_k^n)^2}_{\text{Dependent on } \mathbf{w}} + Nc \ln \sigma + \frac{Nc}{2} \ln(2\pi) \quad (23)$$

PROB. — CLASSIFICATION

- y_k as the probability for belong to class k.
- Multiple attributes (multivariate Bernoulli) (Bishop 1995, sect. 6.8). With $t_k \in \{0, 1\}$

$$p(\mathbf{t}|\mathbf{x}) = \prod_{k=1}^{c} p(t_k|\mathbf{x}) = \prod_{k=1}^{c} y_k^{t_k} \left(1 - y_k\right)^{1-t_k} \tag{24}$$

  Normalization to range $[0; 1]$ with logistic function

$$y_k = \frac{1}{1 + \exp\left(a_k\right)} \tag{25}$$

- Multiple exclusive classes (multinomial)

$$p(\mathbf{t}|\mathbf{x}) = \prod_{k=1}^{c} y_k^{t_k} \tag{26}$$

  Cross-entropy error function

$$E = -\sum_n \sum_{k=1}^{c} t_k^n \ln y_k^n \tag{27}$$

  Normalization of range to $[0; 1]$ with the softmax function

$$y_k = \frac{\exp\left(a_k\right)}{\sum_{k'} \exp\left(a_{k'}\right)} \tag{28}$$

UNSUPERVISED LEARNING

- No target, density estimation of $p(\mathbf{x})$ with $p(\mathbf{x}|\boldsymbol{\theta})$

UNSUPERVISED/SUPERVISED

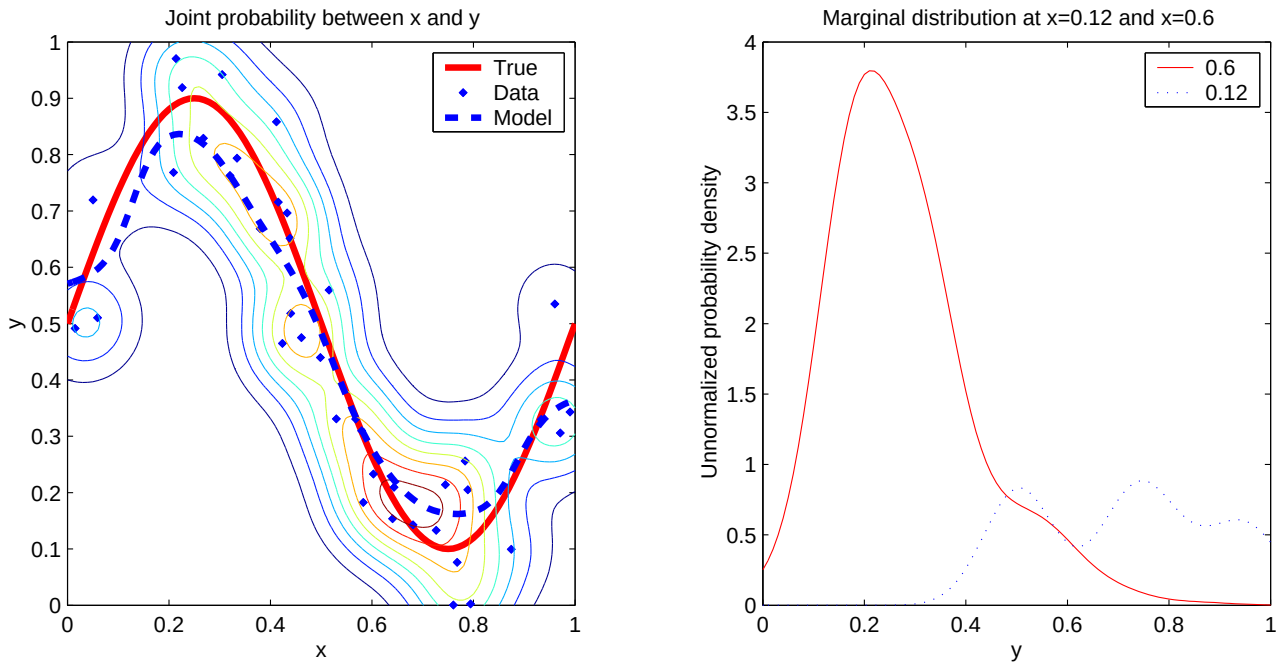

Figure 3: "Joint Modeling": Kernel density modeling with homogenous variance in x and y. Noise is independent between x and y.

- "Joint modeling": Unsupervised/unsupervised distinction might not always be appropriate, e.g., regression can be done with unsupervised methods, where the joint probability for input and output is modeled
- Dependent on noise assumptions (noise on the input).
- A related model in (Bishop 1995, fig 6.7)

GENERALIZATION

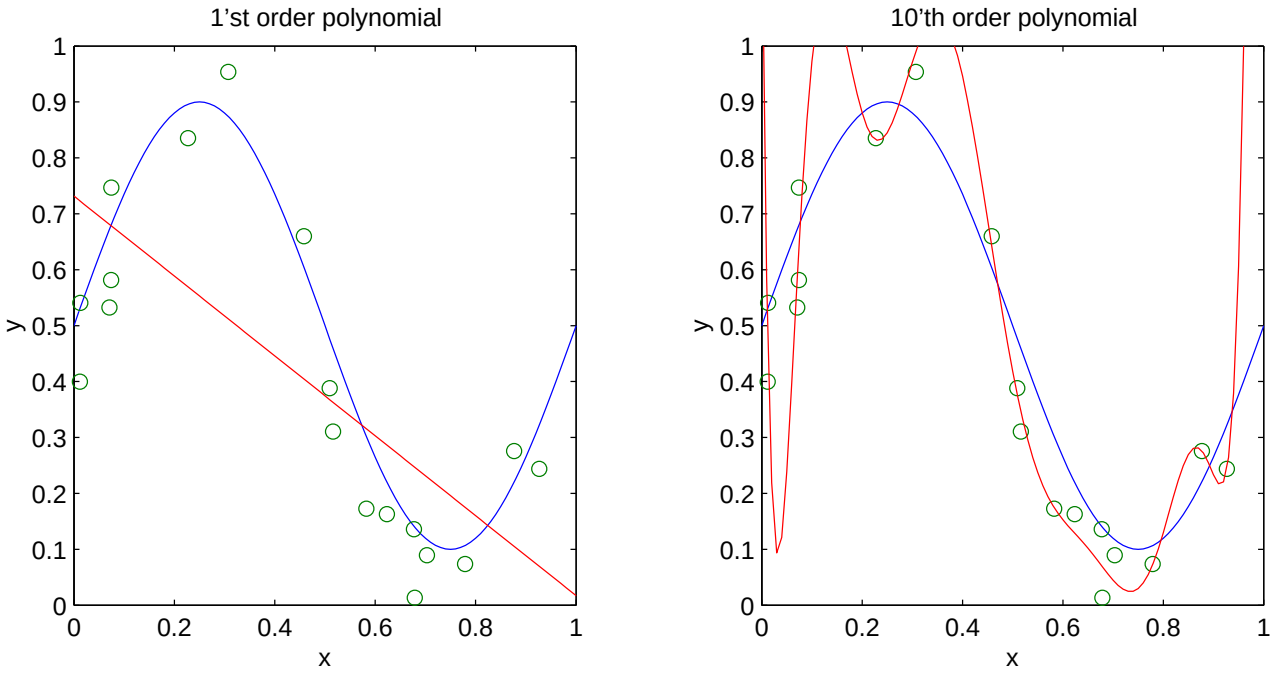

Figure 4: Overfitting and underfitting: Example of overfitting and underfitting for one-dimensional curvefitting (Bishop 1995, eq 1.4). $h(x) = 0.5 + 0.04 * \sin(2\pi x)$. Blue the "true curve". Red is estimated models.

- The variance on the parameters should be small: Not applicable to non-parametric models, such as a neural network because parameter space symmetries, e.g., sign-flip and hidden units permutations, (Bishop 1995, sect. 4.4).
- The prediction of y on the training set should be small: Problem with overfitting.
- The prediction of y on a *new* dataset should be small.

LEARNING CURVE

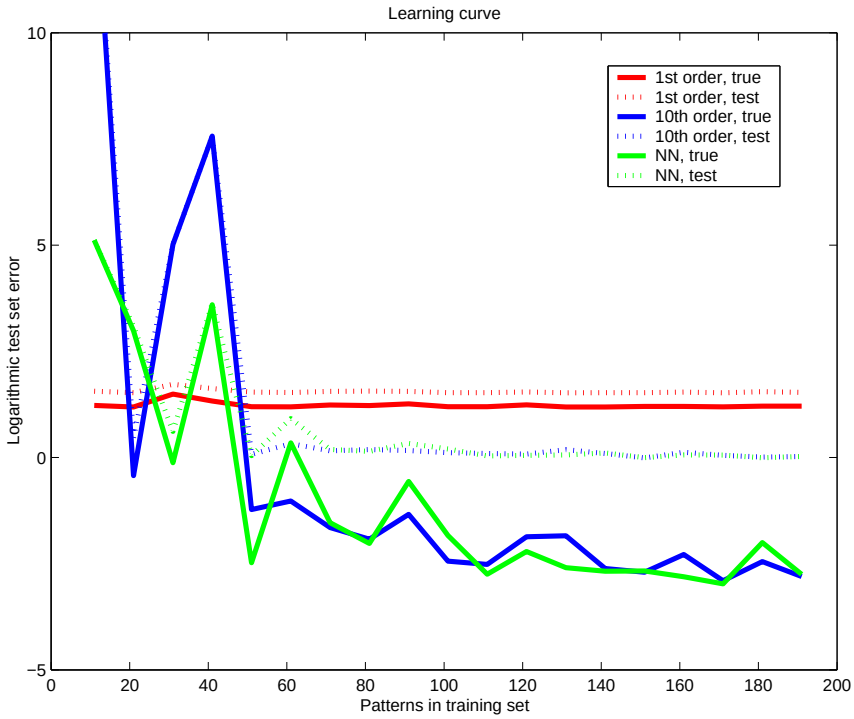

Figure 5: Learning curves for three different models: 1st order polynomial, 10th order polynomial (blue) and a neural network (green) with 10 hidden units ($W = 31$). The target function is (Bishop 1995, eq 1.4).

- Generalization as a function of training set size N
- Complex models should benefit more than simple models:
  - Simple linear model (red): constant error with no benefit of extra training data
  - Complex models (blue/green): Decreasing test set error.
- Select the model according to the number of training examples.

BIAS VARIANCE DECOMPOSITION

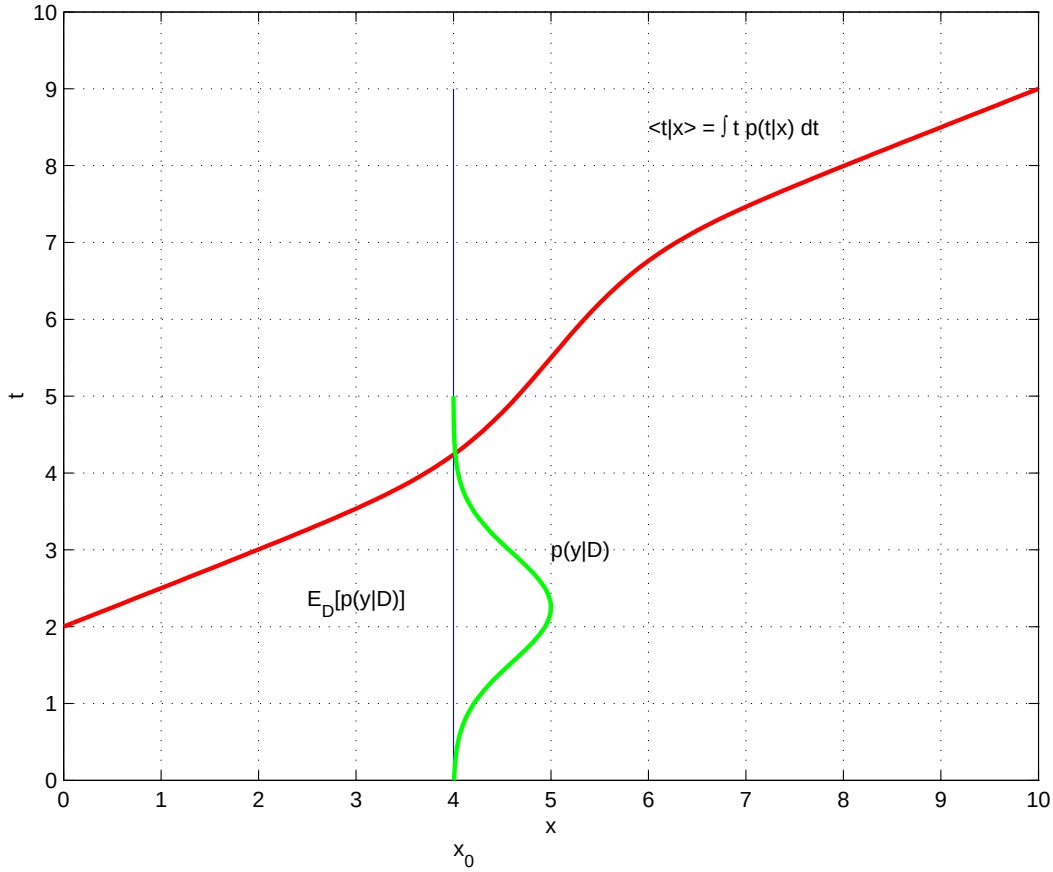

Figure 6: Bias variance decomposition.

- Ensemble of finite training sets, y a stochastic variable dependent on the training set D: $p(y|D)$
- Bias variance decomposition, (Bishop 1995, eq. 9.7)

$$\mathcal{E}_D\left[\{y(\mathbf{x}) - \langle t|\mathbf{x}\rangle\}^2\right] = \tag{29}$$

$$\underbrace{\{\mathcal{E}_D\left[y(\mathbf{x})\right] - \langle t|\mathbf{x}\rangle\}^2}_{(\text{bias})^2} + \underbrace{\mathcal{E}_D\left[\{y(\mathbf{x}) - \mathcal{E}_D\left[y(\mathbf{x})\right]\}^2\right]}_{\text{variance}} \tag{30}$$

CONTROLLING THE EFFECTIVE COMPLEXITY

Bishop (1995, p. 332) distinguishes between:

- Structural stabilization, changing the number of parameters
  - *Optimal Brain Damage* (OBD) pruning
  - *Optimal Brain Surgeon* (OBS) pruning
  - Node pruning
- Regularization, "Adding a penalty term Ω to the cost function"
  - Weight decay, (Bishop 1995, sect. 9.2.1) also called ridge regression

$$\Omega = 1/2 \sum_i w_i^2. \tag{31}$$

  - Soft weight sharing, Weights generated from a mixture of Gaussians.
  - Early stopping. Stop the optimization when the validation set is lowest.
  - Training with noise, perturbing the data points in the training set with noise.

EARLY STOP

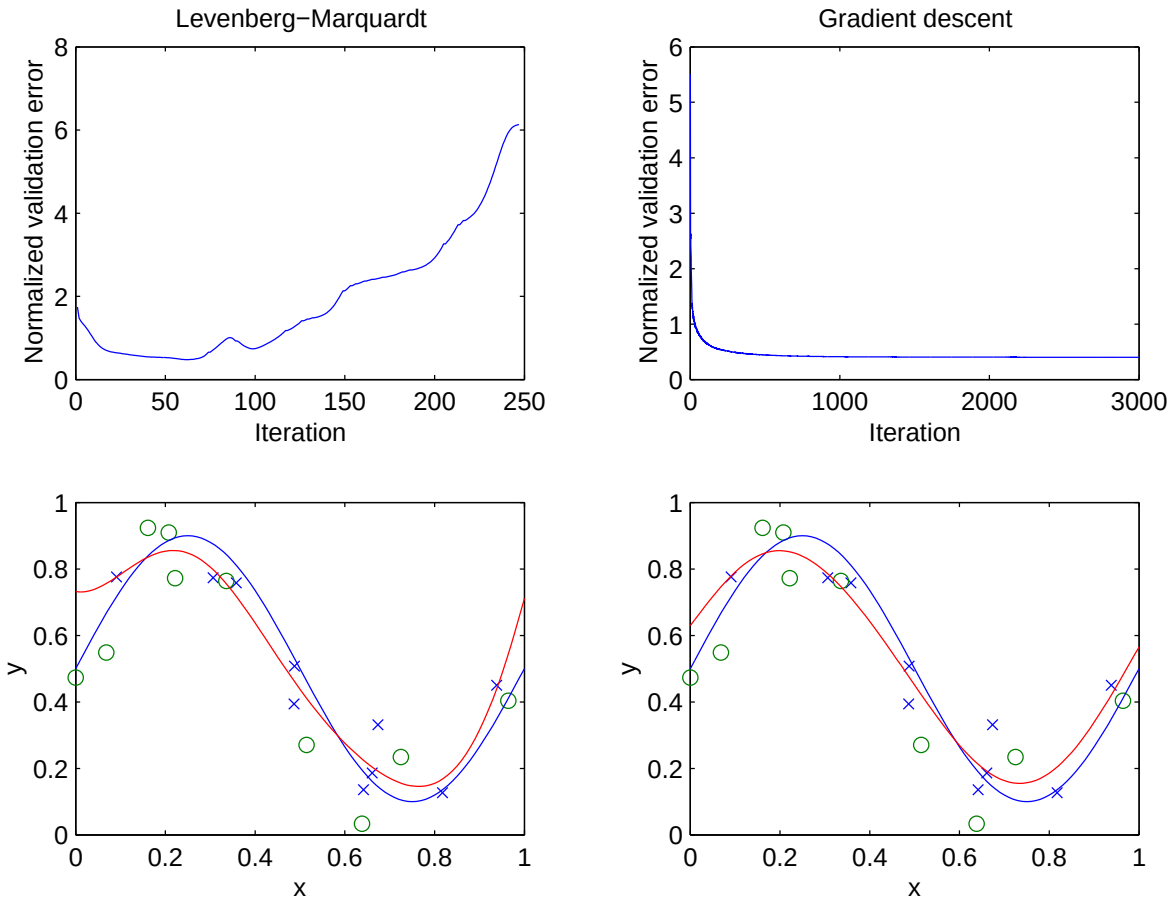

Figure 7: Two-layer neural network curvefitting with least squares an with 40 hidden units and $N = 10$ training examples. "x" is training set and "o" is validation set.

- Effective optimization (Levenberg-Marquardt): fast learning and fast overfitting.
- Slow optimization (gradient descent with adaptive step size): Slow convergence, but no overfitting (yet!).

REGULARIZATION FROM PROBABILISTIC ASSUMPTIONS

- Bayes formula, (Bishop 1995, eq. 10.3)

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})\, p(\mathbf{w})}{p(\mathcal{D})}, \tag{32}$$

  where $\mathbf{w}$ is the parameters and $\mathcal{D} \equiv (t^1, \ldots, t^N)$ is the training set of the target.

- $p(\mathbf{D})$ is constant for a fixed data set

$$p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})\, p(\mathbf{w}) \tag{33}$$

- Cost function

$$E = -\ln p(\mathcal{D}|\mathbf{w}) - \ln p(\mathbf{w}) \tag{34}$$

- If Gaussian prior (indepedent) on the weights $p(\mathbf{w}) \propto \exp(-\lambda \sum_i w_i^2)$

$$E = -\ln p(\mathcal{D}|\mathbf{w}) + \lambda \sum_i w_i^2 \tag{35}$$

  which is *weight decay*.

PRUNING BY OBD

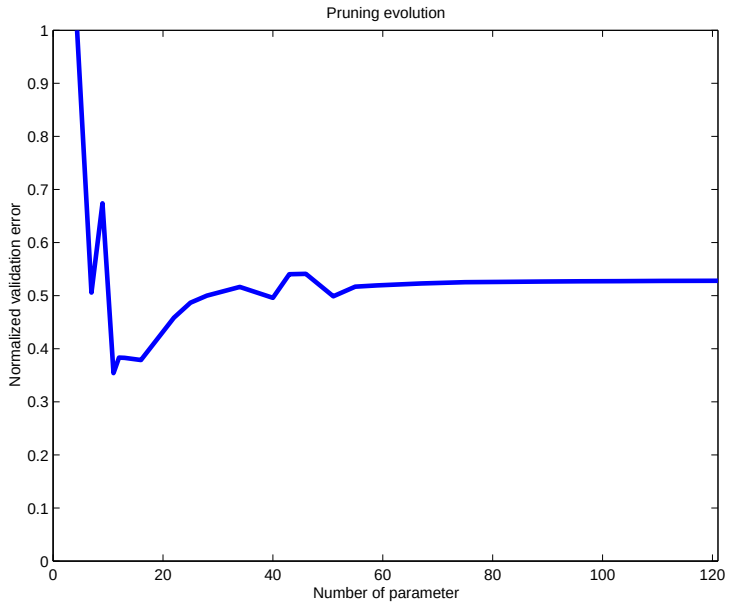

Figure 8: Pruning by OBD in a two-layer neural network curvefitting with least squares and with 40 hidden units and $N = 50$ training examples and $N_{\text{val}} = 50$ validation examples.

- Optimal Brain Damage (OBD) considers the *saliency* of weights: The change in the cost function when a small perturbation is made on a weight (Bishop 1995, eq. 9.66)

$$\delta E = \underbrace{\sum_i \frac{\partial E}{\partial w_i}\,\delta w_i}_{\text{Ignore if optimized}} + \frac{1}{2}\sum_i \sum_j H_{ij}\,\delta w_i\,\delta w_j + \ldots \tag{36}$$

  and diagonal approximation to the Hessian. Erase (set to zero) the weigths associated with low effect (saliency).

COMMITTEE OF NETWORKS

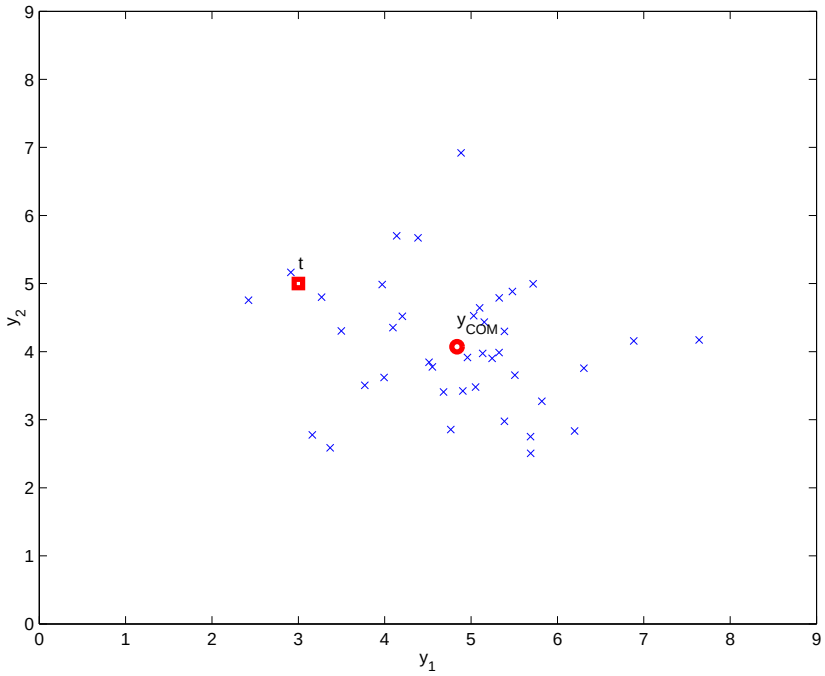

Figure 9: Committee network for model with two outputs.

- Consensus model, e.g., average output of L models, (Bishop 1995, eq. 9.83)

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{L}\sum_{i=1}^{L} y_i(\mathbf{x}) \tag{37}$$

- This prediction is better than the average error of the individual models ($E_{\mathrm{COM}} < E_{\mathrm{AV}}$), if
  - Errors are uncorrelated. Fully correlated (the same model) $E_{\mathrm{COM}} = E_{\mathrm{AV}}$.
  - Error function is convex, e.g., Gaussian

COMMITTEE OF NETWORKS

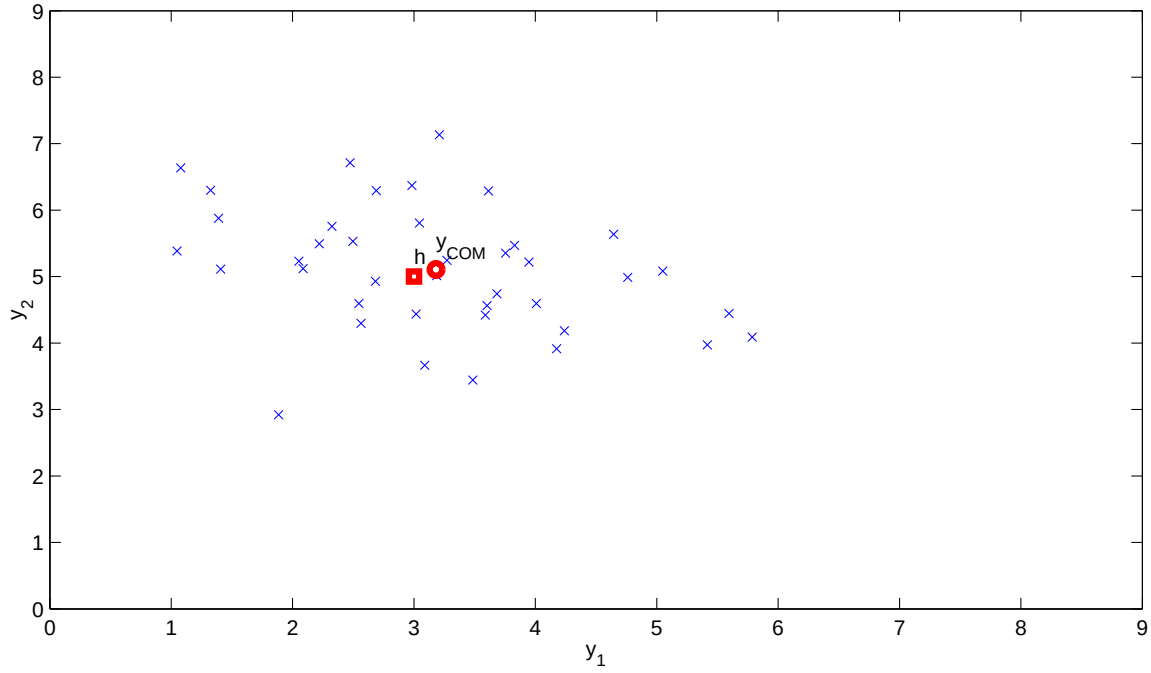

Figure 10: Committee network for model with two outputs.

- With no model bias against the true output

$$E_{\text{COM}} = \frac{1}{L} E_{\text{AV}} \tag{38}$$

$$E_{\text{AV}} = \frac{1}{L} \sum_{i=1}^{L} E_i = \frac{1}{L} \sum_{i=1}^{L} \mathcal{E}\left[\epsilon_i^2\right] \tag{39}$$

$$E_{\text{COM}} = \mathcal{E}\left[\left(\frac{1}{L} \sum_{i=1}^{L} y_i(\mathbf{x} - h(\mathbf{x})\right)^2\right] = \mathcal{E}\left[\left(\frac{1}{L} \sum_{i=1}^{L} \epsilon_i\right)^2\right] \tag{40}$$

COMMITTEE OF NETWORKS

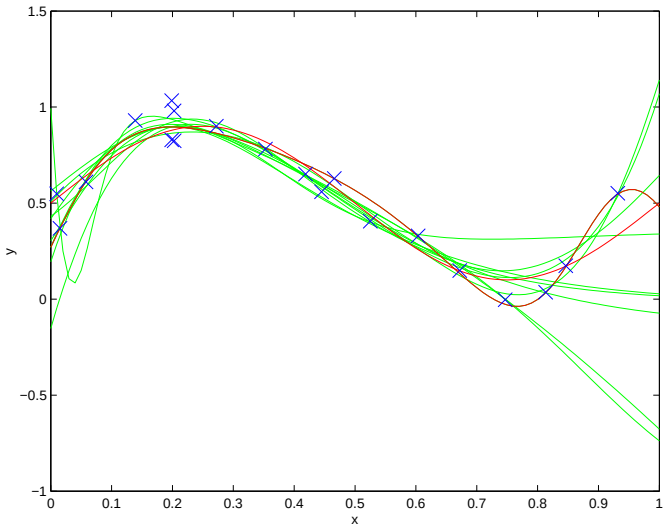

Figure 11: Committee neural network prediction.

- Models should be heterogenous, e.g., a linear model will fit the same curve.
- Averaging over models mostly reduces the *variance* rather than the *bias*.
- Neural network committee example (permuting training and test set, different seed, very little regularization) with validation set and early stop. 10 networks in committee.

$$E_{\text{AV, Test set}} = 0.0394 \tag{41}$$

$$E_{\text{COM, Test set}} = 0.0084 \tag{42}$$

2 individual models were better, 8 worse. Empirical observation: Errors are not necessarily Gaussian

MODEL ORDER SELECTION, VALIDATION

- Validation-based (Bishop 1995, sect. 9.8.1), test-set should be independent "Hold out method") and from the same distribution
  - Single-set validation. A finite size validation set will be "noisy".
  - Cross-validation, partition the data set in S distinct segments. S times larger computation

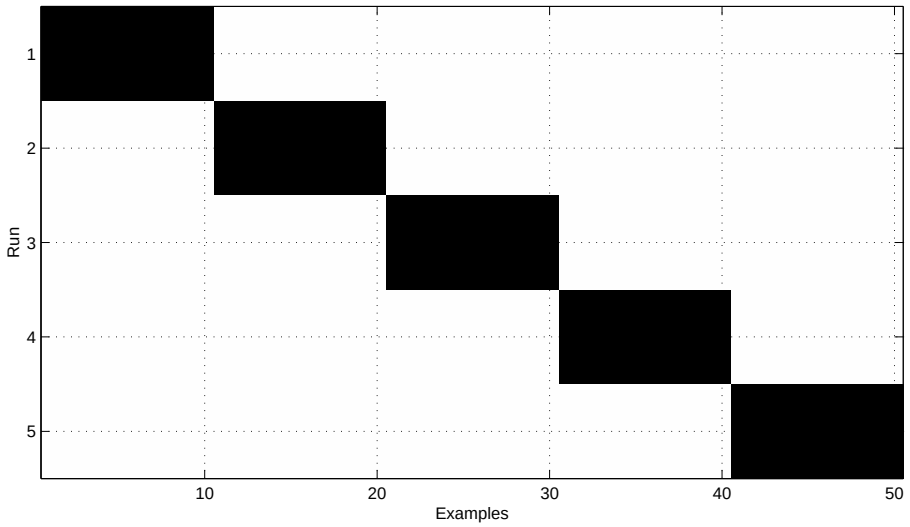

Figure 12: Cross-validation partioning (Bishop 1995, figure 9.17). With $N = 50$ examples and $S = 5$ distinct partitions of the data.

  - Leave-one-out. Only one example in the validation set.
  - "Overvalidation": If the validation set is applied too much the model might not generalize (Bishop 1995, p. 364–365), e.g., consider random models picked by the validation set.

BIAS VARIANCE TRADE-OFF

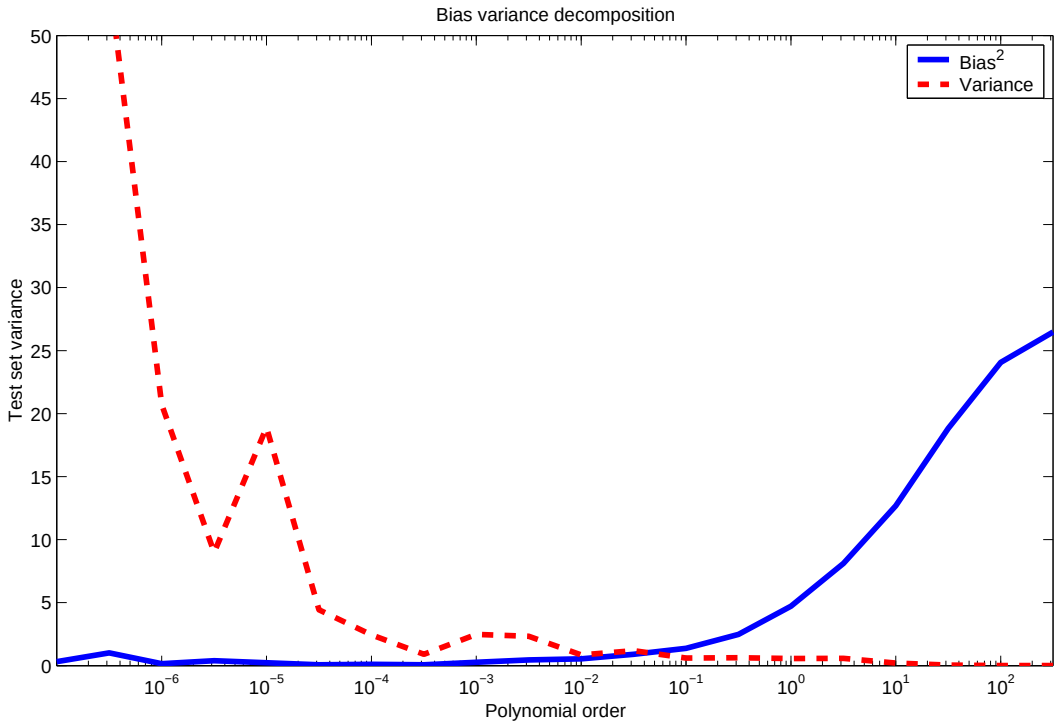

Figure 13: Bias variance decomposition on a 20th order polynomial with a weight decay hyperparameter varied with $N = 10$ examples and 100 runs (Bishop 1995, fig. 9.16, eqs. 9.109 and 9.110).

- Bias variance as a function of model complexity (Bishop 1995, figure 9.16): $\mathcal{E}_D[y] \approx \bar{y} = \sum_{i=1}^{100} y_i$ and $\langle t|x \rangle$ assessed by large validation set
- Simple models: High bias, low variance, e.g., a constant model $y = 0$ have no variance and bias as $\langle t|\mathbf{x} \rangle^2$.
- Complex model: Might have low bias and high variance, e.g., a model that fits the data points perfectly.

MODEL ORDER SELECTION — COMPLEXITY CRITERIA

- Complexity criteria, (Bishop 1995, sect. 9.8.3). One of the forms (Bishop 1995, eq. 9.111)

$$\text{PE} = \text{training error} + \text{complexity term} \tag{43}$$

  For sum-of-squares error, $E = \frac{1}{2}\sum_{n=1}^{N}\left[y(\mathbf{x},\mathbf{w}) - t\right]^2$

  - Final prediction error (FPE), (Bishop 1995, eq. 9.112)

$$\text{FPE} = \frac{2E}{N} + \frac{W}{N}\sigma^2 \tag{44}$$

    where W is the number of free parameters.

  - Generalized prediction error (GPE)

$$\text{GPE} = \frac{2E}{N} + \frac{2\gamma}{N}\sigma^2 \tag{45}$$

    where γ is an effective number of parameters.

CONCLUSION

- Learning can be performed in a variaty of ways: gradient, Hessian-based.
- Learning problems can be based on probabilistic models: regression, classification, ...
- Model should generalize well: It should not only consider presented data (training) but fit new data as well.
- Complexity of model can be adjusted by varying the number of free paramters, by regularization, pruning or by not training the model "well".
- Combining models ("consensus models", "committee of network") might improve generalization.
- Generalization can be assessed by validation set or by complexity criteria.

REFERENCES

References

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Larsen, J. (1996, January). *Design of Neural Network Filters*. Ph. D. thesis, Electronics Institute, Technical University of Denmark, Lyngby, Denmark. Second edition.