Supervised Machine Learning:
Learning SVMs and Deep Learning

Klaus-Robert Müller

!!et al.!!

Today‘s Tutorial

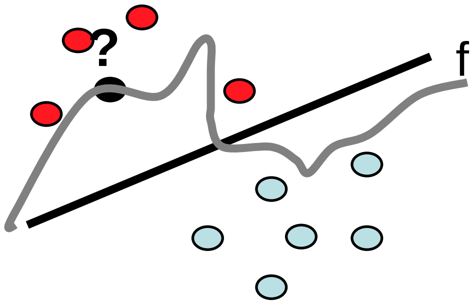
Machine Learning

- introduction: ingredients for ML
- Kernel Methods and Deep networks with explaining & remarks

Applications ML to Physics & Materials

- representation
- models
- remarks

Machine Learning in a nutshell



Typical scenario: learning from data

- given data set **X** and labels **Y** (generated by some joint probabilty distribution p(x,y))

- **LEARN/INFER** underlying **unknown** mapping

$$Y = f(X)$$

Example: left and right imagery...

BUT: how to do this optimally with good performance on **unseen** data?

Kernel-based Learning

Basic ideas in learning theory

Three scenarios: regression, classification & density estimation.
Learn f from examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathbb{R}^n \times \mathbb{R}^m \text{ or } \{\pm 1\}, \quad \text{generated from } P(\mathbf{x}, y),$$

such that expected number of errors on test set (drawn from $P(\mathbf{x}, y)$),

$$R[f] = \int \frac{1}{2} |f(\mathbf{x}) - y|^2 \, dP(\mathbf{x}, y),$$

is minimal *(Risk Minimization (RM))*.

Problem: P is unknown. $\longrightarrow$ need an *induction principle*.

Empirical risk minimization (ERM): replace the average over $P(\mathbf{x}, y)$ by an average over the training sample, i.e. minimize the training error

$$R_{emp}[f] = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} |f(\mathbf{x}_i) - y_i|^2$$

Basic ideas in learning theory II

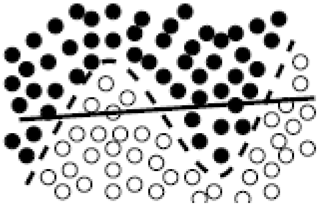
- Law of large numbers: $R_{emp}[f] \to R[f]$ as $N \to \infty$.
  "consistency" of ERM: for $N \to \infty$, ERM should lead to the same result as RM?

- No: *uniform* convergence needed (Vapnik) → VC theory.
  Thm. [classification] (Vapnik 95): with a probability of at least $1 - \eta$,

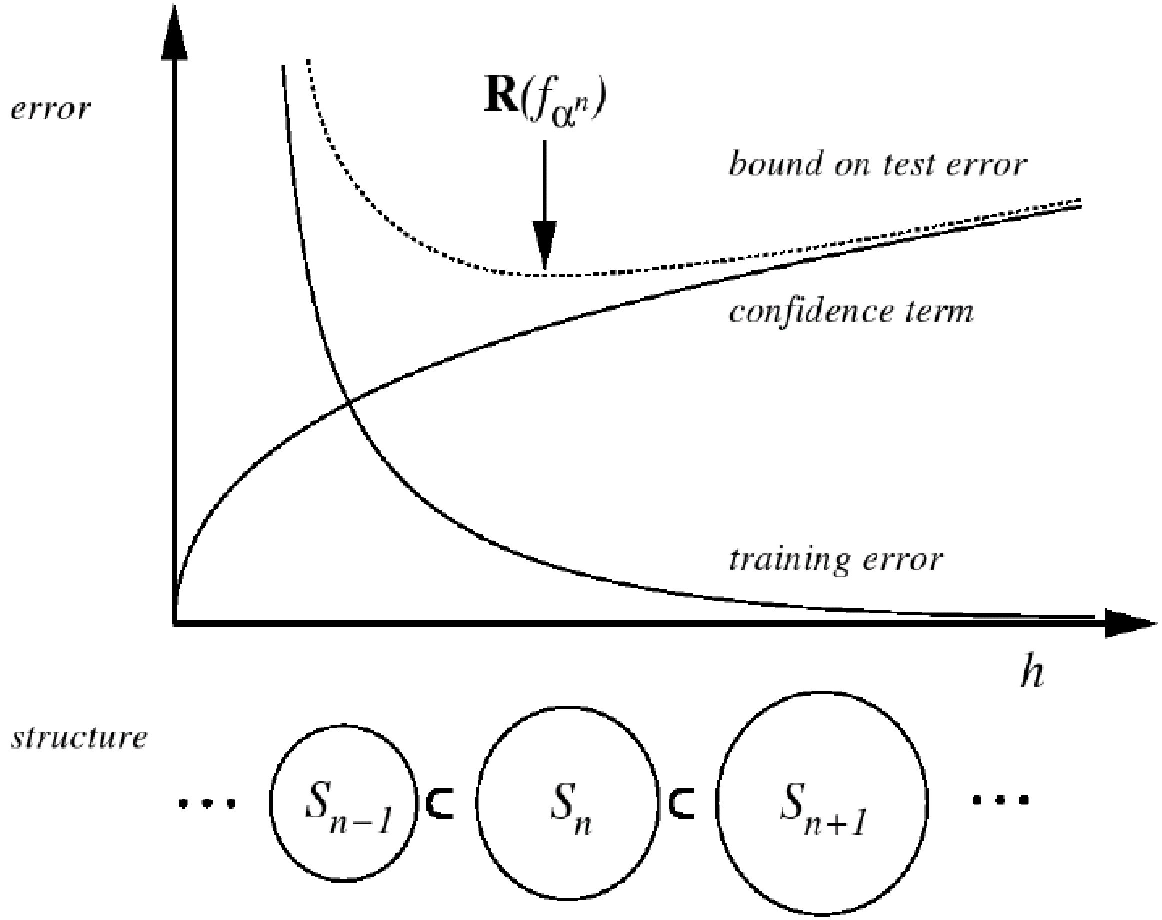
$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log \frac{2N}{d} + 1\right) - \log(\eta/4)}{N}}.$$

- Structural risk minimization (SRM): introduce structure on set of functions $\{f_\alpha\}$ & minimize RHS to get low risk! (Vapnik 95)

- d is VC dimension, measuring complexity of function class



Structural Risk Minimization: the picture



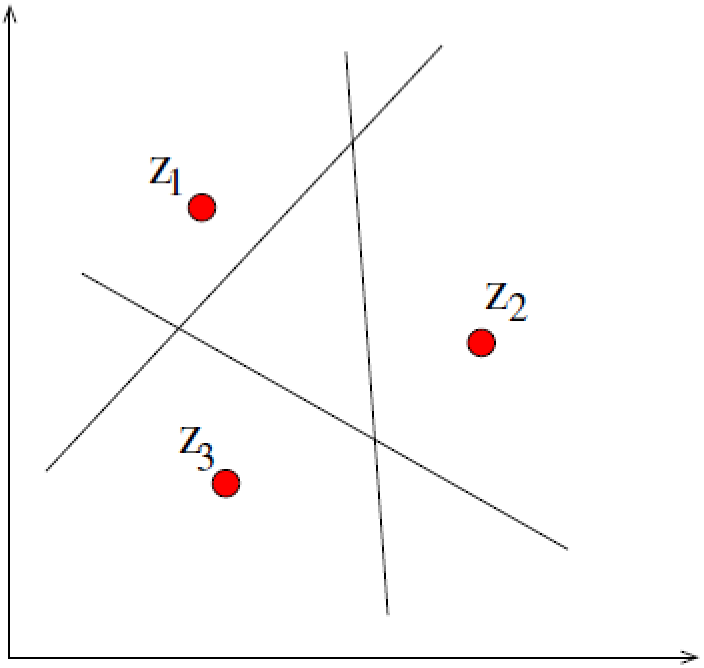
Learning f requires small training error *and* small complexity of the set $\{f_\alpha\}$.

VC Dimensions: an examples

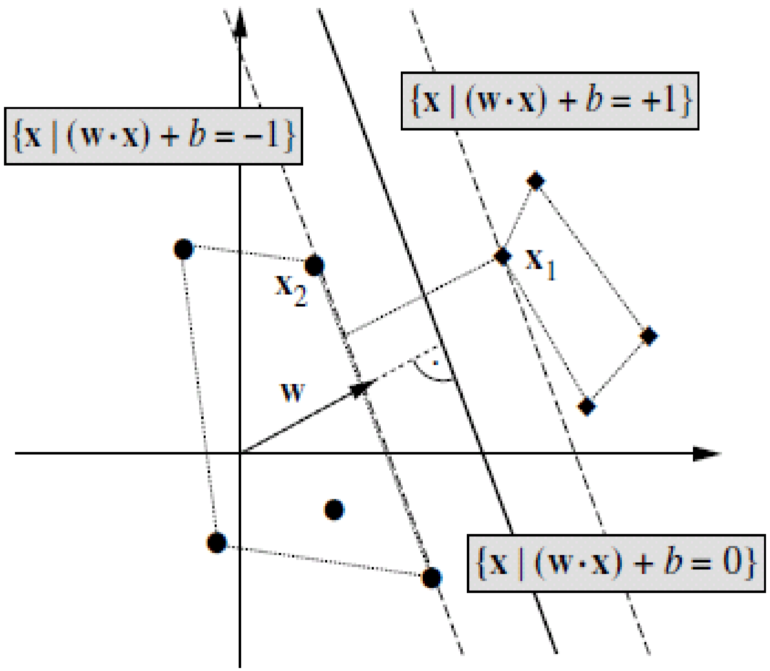
Half-spaces in $\mathbf{R}^2$:

$$f(x, y) = \text{sgn}(a + bx + cy), \quad \text{with parameters } a, b, c \in \mathbf{R}$$

- Clearly, we can shatter three non-collinear points.
- But we can never shatter four points.
- Hence the VC dimension is $d = 3$
- in n dimensions: VC dimension is $d = n + 1$



Linear Hyperplane Classifier



Note:

$$(\mathbf{w} \cdot \mathbf{x}_1) + b = +1$$
$$(\mathbf{w} \cdot \mathbf{x}_2) + b = -1$$

=> $(\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2)) = 2$

=> $\left(\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}_1 - \mathbf{x}_2)\right) = \frac{2}{\|\mathbf{w}\|}$

- hyperplane $y = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$ in canonical form if $\min_{\mathbf{x}_i \in X} |(\mathbf{w} \cdot \mathbf{x}_i) + b| = 1.$, i.e. scaling freedom removed.
- larger margin $\sim 1/\|\mathbf{w}\|$ is giving better generalization $\rightarrow$ LMC!

VC Theory applied to hyperplane classifiers

- Theorem (Vapnik 95): For hyperplanes in canonical form VC–dimension satisfying

$$d \leq \min\{[R^2 \|\mathbf{w}\|^2] + 1, n + 1\}.$$

Here, R is the radius of the smallest sphere containing data.
Use d in SRM bound

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log\frac{2N}{d} + 1\right) - \log(\eta/4)}{N}}.$$

- maximal margin = minimum $\|\mathbf{w}\|^2$ $\rightarrow$ good generalization, i.e. low risk, i.e. optimize

$$\min \|\mathbf{w}\|^2$$

- independent of the dimensionality of the space!

Feature Spaces & curse of dimensionality

The Support Vector (SV) approach: *preprocess* the data with

$$\begin{aligned} \Phi : \mathbf{R}^N &\rightarrow F \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \\ \text{where } N &\ll \dim(F). \end{aligned}$$

to get data $(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_N), y_N) \in F \times \mathbf{R}^M$ or $\{\pm 1\}$.

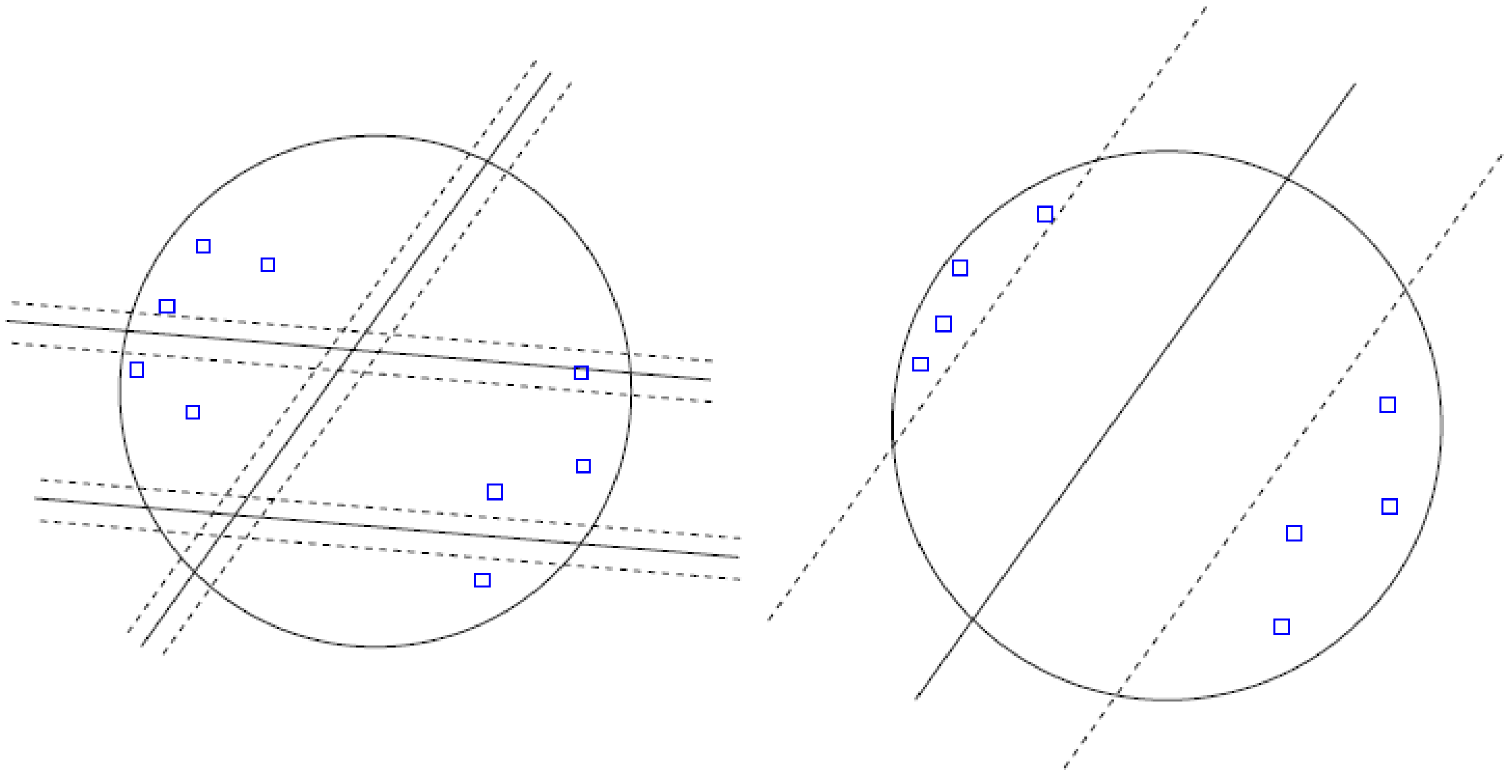
Learn $\tilde{f}$ to construct $f = \tilde{f} \circ \Phi$

- classical statistics: **harder**, as the data are high-dimensional
- SV-Learning: (in some cases) **simpler**:

If Φ is chosen such that $\{\tilde{f}\}$ allows small training error *and* has low complexity, then we can guarantee good generalization.

The *complexity* matters, not the *dimensionality* of the space.

Margin Distributions – large margin hyperplanes



Feature Spaces & curse of dimensionality

The Support Vector (SV) approach: *preprocess* the data with

$$\begin{aligned} \Phi : \mathbf{R}^N &\rightarrow F \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) \\ \text{where } N &\ll \dim(F). \end{aligned}$$

to get data $(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_N), y_N) \in F \times \mathbf{R}^M$ or $\{\pm 1\}$.

Learn $\tilde{f}$ to construct $f = \tilde{f} \circ \Phi$

- classical statistics: **harder**, as the data are high-dimensional
- SV-Learning: (in some cases) **simpler**:

If Φ is chosen such that $\{\tilde{f}\}$ allows small training error *and* has low complexity, then we can guarantee good generalization.

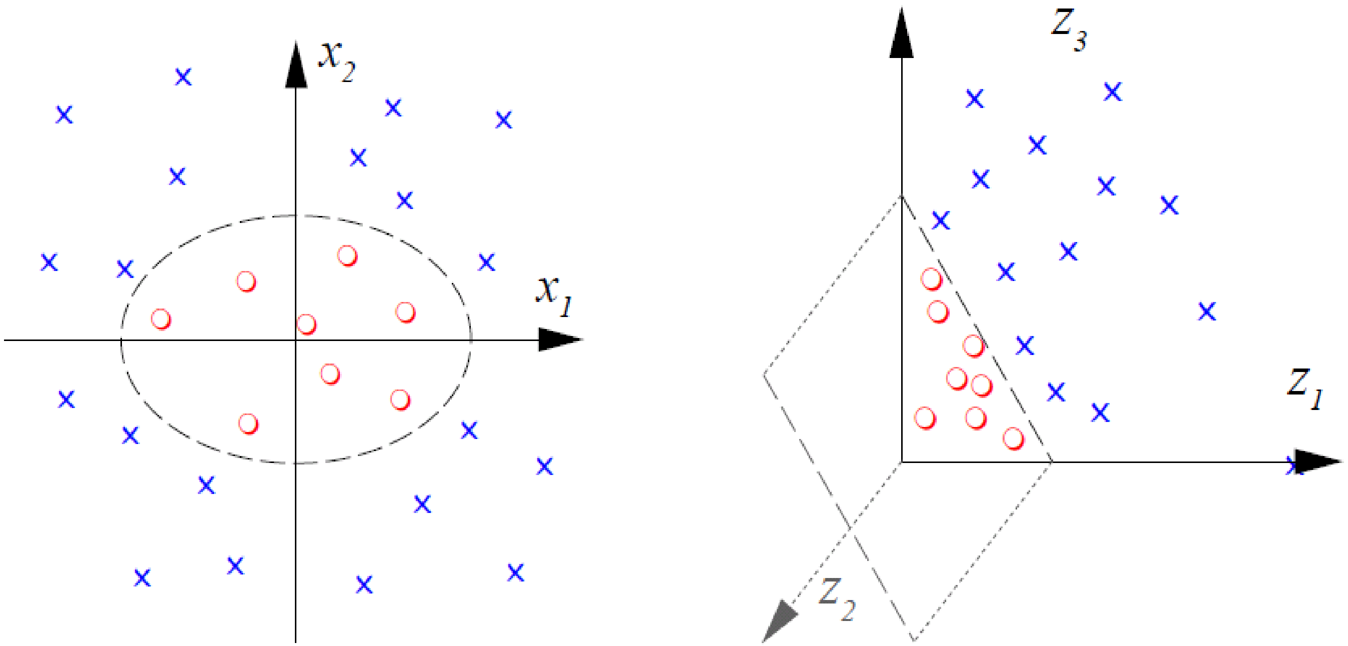
The *complexity* matters, not the *dimensionality* of the space.

Nonlinear Algorithms in Feature Space

Example: all second order monomials

$$\Phi : \mathbf{R}^2 \rightarrow \mathbf{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



The kernel trick: an example

(cf. Boser, Guyon & Vapnik 1992)

$$
\begin{aligned}
(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) &= (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)(y_1^2, \sqrt{2}\, y_1 y_2, y_2^2)^\top \\
&= (\mathbf{x} \cdot \mathbf{y})^2 \\
&= : k(\mathbf{x}, \mathbf{y})
\end{aligned}
$$

- Scalar product in (**high dimensional**) feature space can be computed in $\mathbf{R}^2$!
- works only for Mercer Kernels $k(\mathbf{x}, \mathbf{y})$

Kernology

[Mercer] If k is a continuous kernel of a positive integral operator on $L_2(\mathcal{D})$ (where $\mathcal{D}$ is some compact space),

$$\int f(\mathbf{x})k(\mathbf{x},\mathbf{y})f(\mathbf{y})\,d\mathbf{x}\,d\mathbf{y} \geq 0, \quad \text{for} \quad f \neq 0$$

it can be expanded as

$$k(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N_F} \lambda_i \psi_i(\mathbf{x})\psi_i(\mathbf{y})$$

with $\lambda_i > 0$, and $N_F \in \mathbf{N}$ or $N_F = \infty$. In that case

$$\Phi(\mathbf{x}) := \begin{pmatrix} \sqrt{\lambda_1}\psi_1(\mathbf{x}) \\ \sqrt{\lambda_2}\psi_2(\mathbf{x}) \\ \vdots \end{pmatrix}$$

satisfies $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})) = k(\mathbf{x},\mathbf{y})$.

Kernology II

Examples of common kernels:

$$
\begin{aligned}
\text{Polynomial} \quad k(\mathbf{x},\mathbf{y}) &= (\mathbf{x}\cdot\mathbf{y}+c)^d \\
\text{Sigmoid} \quad k(\mathbf{x},\mathbf{y}) &= \tanh(\kappa(\mathbf{x}\cdot\mathbf{y})+\Theta) \\
\text{RBF} \quad k(\mathbf{x},\mathbf{y}) &= \exp\left(-\|\mathbf{x}-\mathbf{y}\|^2/(2\,\sigma^2)\right) \\
\text{inverse multiquadric} \quad k(\mathbf{x},\mathbf{y}) &= \frac{1}{\sqrt{\|\mathbf{x}-\mathbf{y}\|^2+c^2}}
\end{aligned}
$$

Note: kernels correspond to regularization operators (a la Tichonov) with regularization properties that can be conveniently expressed in Fourier space, e.g. Gaussian kernel corresponds to general smoothness assumption (Smola et al 98)!

A RKHS representation of $\mathcal{F}$

$$\tilde{\Phi} : \mathbf{R}^N \longrightarrow \mathcal{H}, \qquad \mathbf{x} \mapsto k(\mathbf{x}, .)$$

Need a dot product $\langle ., . \rangle$ for $\mathcal{H}$ such that

$$\langle \tilde{\Phi}(\mathbf{x}), \tilde{\Phi}(\mathbf{y}) \rangle = k(\mathbf{x}, \mathbf{y}), \quad \text{i.e. require} \quad \langle k(\mathbf{x}, .), k(\mathbf{y}, .) \rangle = k(\mathbf{x}, \mathbf{y}).$$

For a Mercer kernel $k(\mathbf{x}, \mathbf{y}) = \sum_j \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{y})$, with $\lambda_i > 0$ for all i, and $(\psi_i \cdot \psi_j)_{L_2(\mathcal{C})} = \delta_{ij}$, this can be achieved by choosing $\langle ., . \rangle$ such that

$$\langle \psi_i, \psi_j \rangle = \delta_{ij} / \lambda_i.$$

$\mathcal{H}$, the closure of the space of all functions

$$f(\mathbf{x}) = \sum_i a_i k(\mathbf{x}, \mathbf{x}_i),$$

with dot product $\langle ., . \rangle$, is called reproducing kernel Hilbert space

Hyperplane $y = \mathrm{sgn}\left(\mathbf{w} \cdot \Phi(x) + b\right)$ in $\mathcal{F}$

$$\begin{array}{ll} \min & \|\mathbf{w}\|^2 \\ \text{subject to} & y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] \geq 1 \quad \text{for } i = 1 \ldots N \end{array}$$

(i.e. training data separated correctly, otherwise introduce slack variables).

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \cdot ((\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b) - 1\right).$$

obtain unique α_i by QP (no local minima!): dual problem

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

$$\text{i.e.} \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i).$$

Substitute both into L to get the *dual problem*

Hyperplane in $\mathcal{F}$ with slack variables: SVM

min $\quad \|\mathbf{w}\|^2 + C\sum_{i=1}^{N} {\xi_i}^p$

subject to $\quad y_i \cdot [(\mathbf{w}\cdot\Phi(\mathbf{x}_i)) + b] \geq 1 - \xi_i$ and $\xi_i \geq 0 \quad$ for $i = 1 \ldots N$

(introduce slack variables if training data not separated correctly)

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \cdot ((\mathbf{w}\cdot\Phi(\mathbf{x}_i)) + b) - 1 \right).$$

obtain unique α_i by QP (no local minima!): dual problem

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0, \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0,$$

i.e. $\quad \sum_{i=1}^{N} \alpha_i y_i = 0 \quad$ and $\quad \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i).$

Substitute both into L to get the *dual problem*

Dual Problem

$$\text{maximize} \qquad W(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

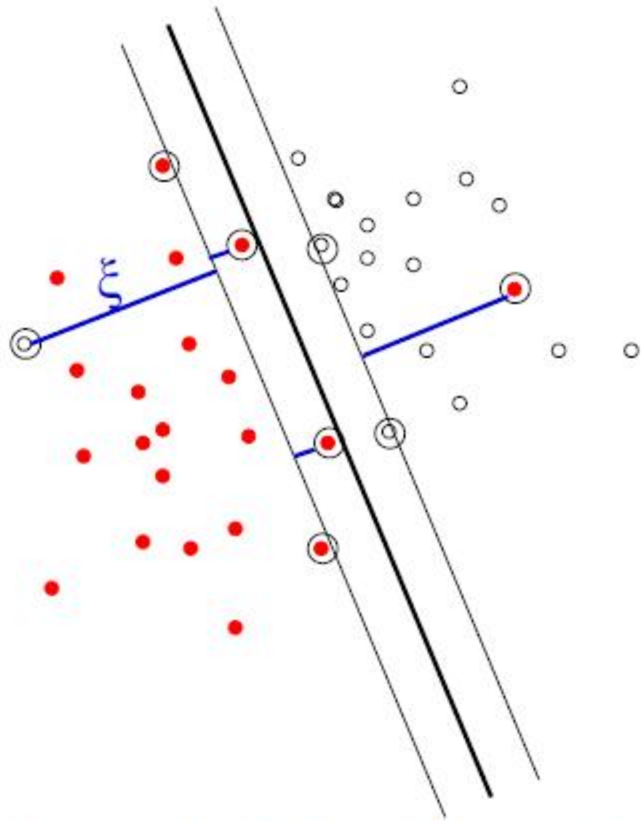
$$\text{subject to} \qquad C \geq \alpha_i \geq 0, \quad i = 1, \ldots, N, \quad \text{and} \quad \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Note: solution determined by training examples (SVs) on /in the margin. Remark: duality gap.

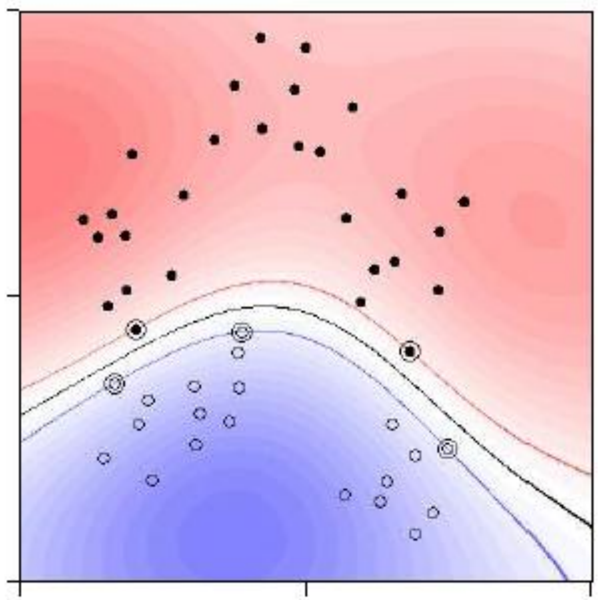
$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] > 1 \qquad \Longrightarrow \alpha_i = 0 \longrightarrow \quad \mathbf{x}_i \text{ irrelevant or}$$

$$y_i \cdot [(\mathbf{w} \cdot \Phi(\mathbf{x}_i)) + b] = 1 \qquad (on\ /\text{in margin}) \longrightarrow \quad \mathbf{x}_i \text{ Support Vector}$$

A Toy Example: $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2)$



linear SV with slack variables



nonlinear SVM, Domain: $[-1, 1]^2$

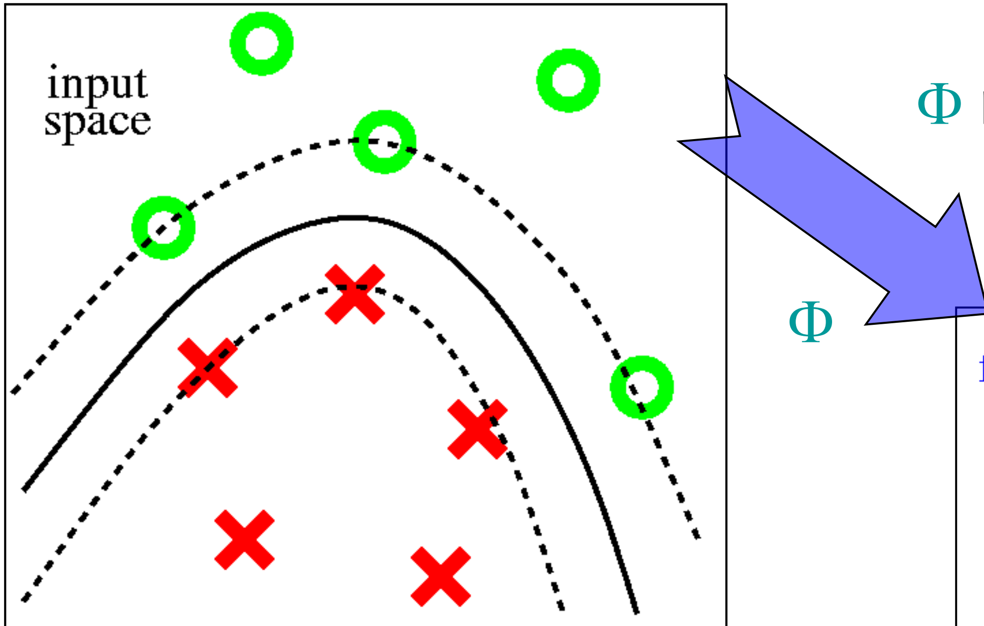
Kernel Trick

- Saddle Point: $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)$.
- Hyperplane in $\mathcal{F}$: $y = \operatorname{sgn}(\mathbf{w} \cdot \Phi(x) + b)$
- putting things together "kernel trick"

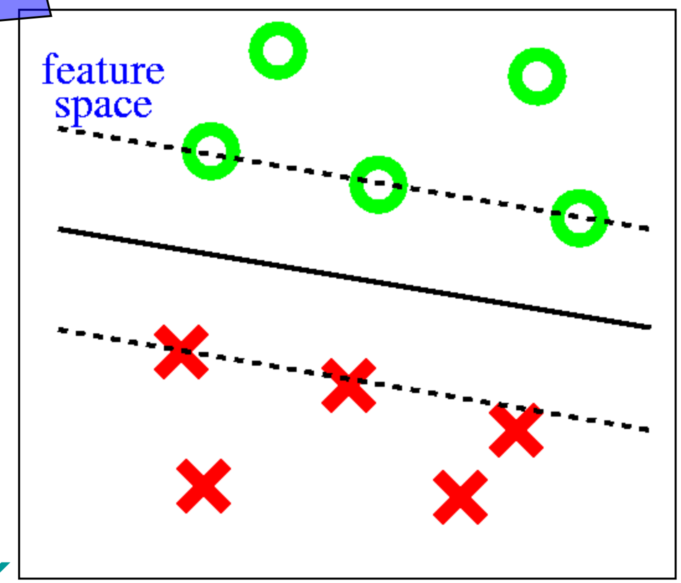
$$
\begin{aligned}
f(\mathbf{x}) &= \operatorname{sgn}(\mathbf{w} \cdot \Phi(\mathbf{x}) + b) \\
&= \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i y_i \, \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}) + b \right) \\
&= \operatorname{sgn}\left(\sum_{i \in \#\mathrm{SVs}} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b \right) \qquad \text{sparse!}
\end{aligned}
$$

- trick: $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$, i.e. never use Φ: only k!!!

Support Vector Machines in a nutshell



Φ rsp. $K(x,y) = \Phi(x) \bullet \Phi(y)$



good theory

non-linear decision by
implicitely mapping the data
into feature space by SV **kernel** function **K**

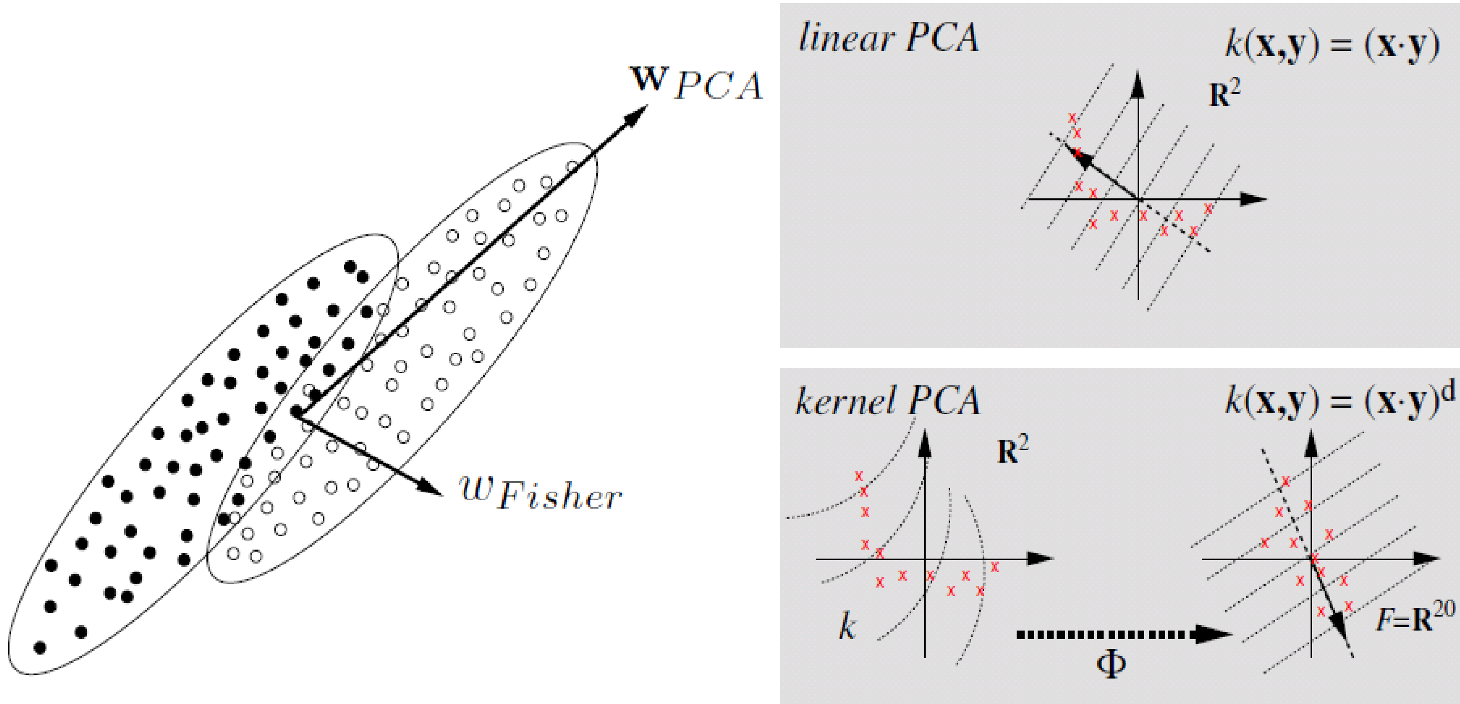
Digestion: Use of kernels

- Question: What makes kernel methods (e.g. SVM) perform well?
- Answer:
  - In the first place: a good idea/theory.
  - But also: The kernel
- Using kernels, we work explicitly in extremely high dimensional spaces (RKHS) with interesting features for themselves (depending on the kernel) [SSM et al. 98]
- Common choices: Gaussian kernel $\exp(-\|\boldsymbol{x}-\boldsymbol{y}\|^2/c)$ or polynomial kernel $(\boldsymbol{x}\cdot\boldsymbol{y})^d$.
- Almost any linear algorithm can be transformed to feature space. [SSM et al. 98]
- With suitable regularization it outperforms its linear counterpart. [Mika et al. 02]

  [Zien et al. 00, Tsuda et al. 02, Sonnenburg et al. 05]
- The kernel can be adopted to specific tasks, e.g. using prior knowledge

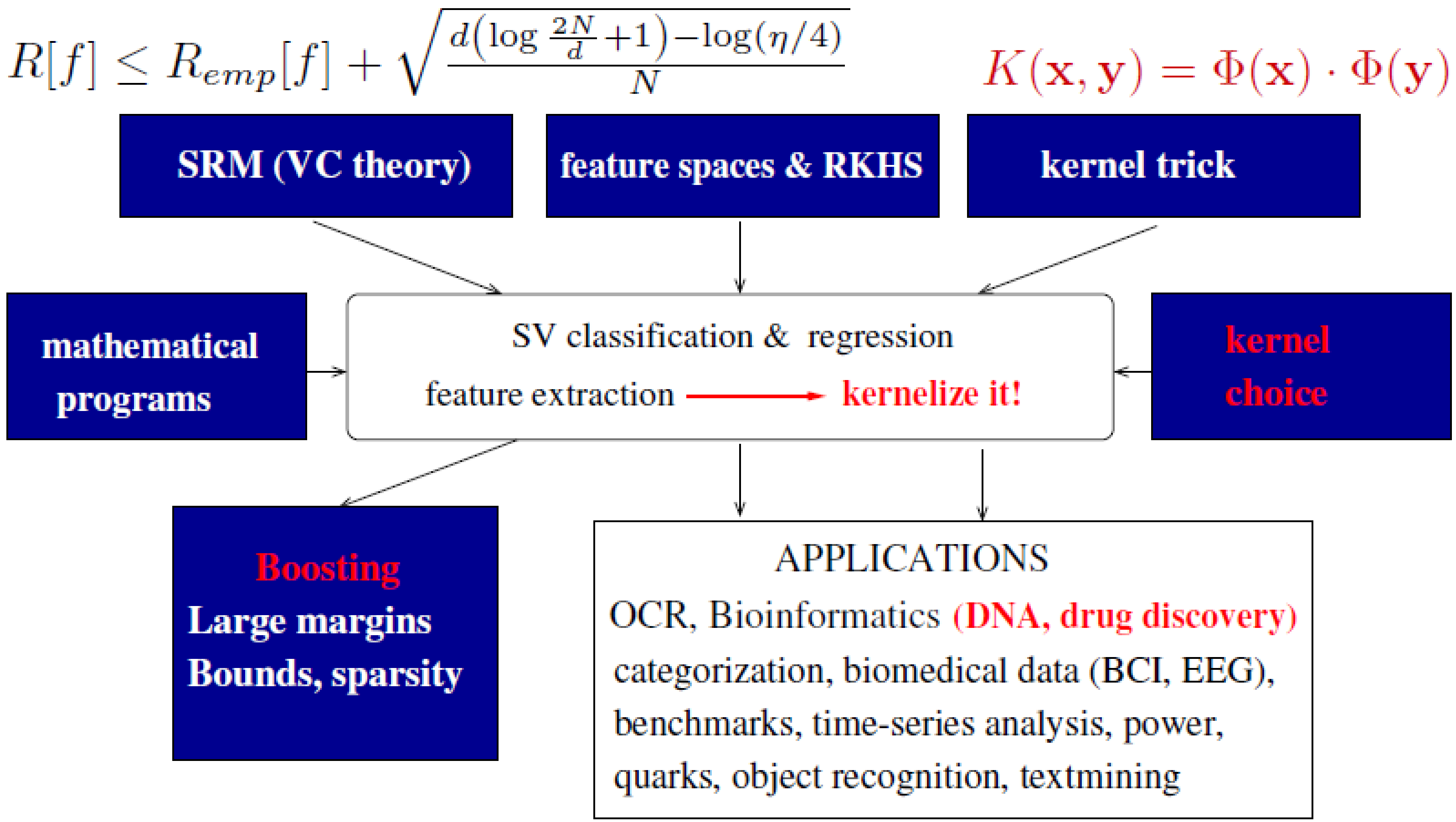
Kernels for graphs, trees, strings etc.

Remark: Kernelizing linear algorithms



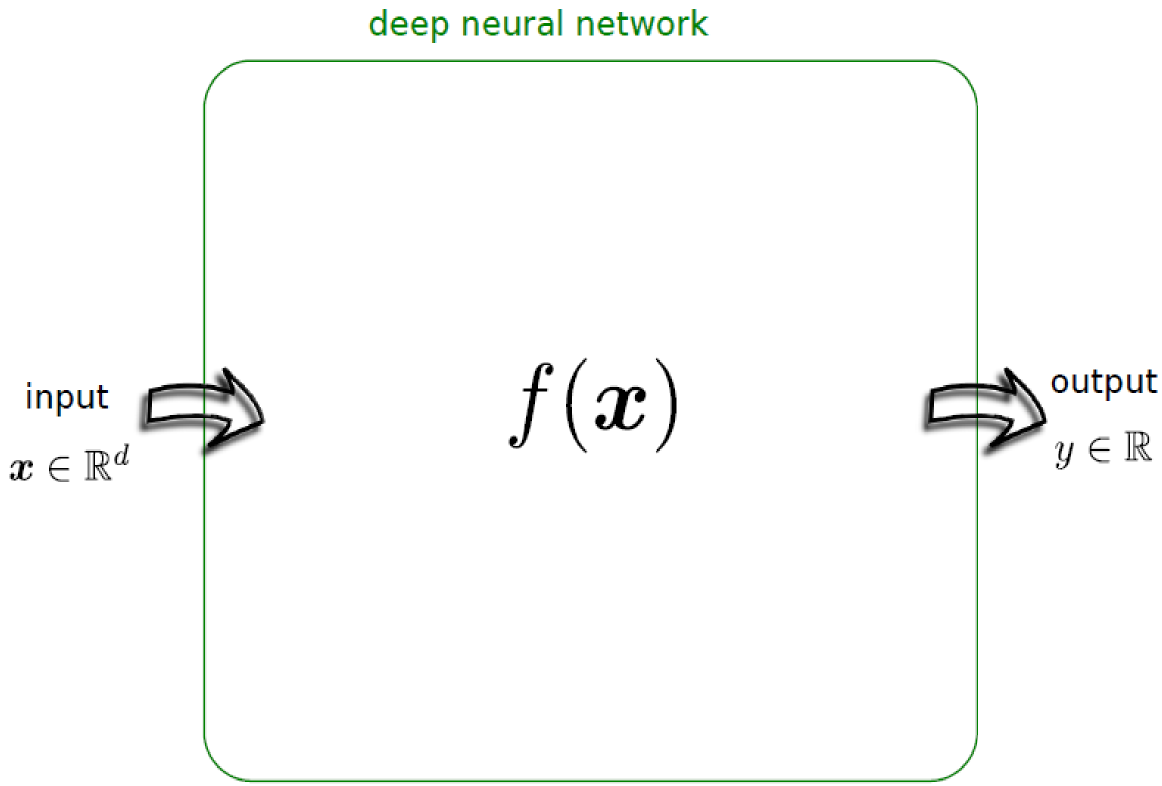
(cf. Schölkopf, Smola and Müller 1996, 1998, Schölkopf et al 1999, Mika et al, 1999, 2000, 2001, Müller et al 2001, Harmeling et al 2003, . . .)

Digestion


$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log \frac{2N}{d}+1\right)-\log(\eta/4)}{N}}$
$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$
SRM (VC theory)
feature spaces & RKHS
kernel trick
mathematical programs
SV classification & regression
feature extraction
kernelize it!
kernel choice
Boosting
Large margins
Bounds, sparsity
APPLICATIONS
OCR, Bioinformatics (DNA, drug discovery)
categorization, biomedical data (BCI, EEG),
benchmarks, time-series analysis, power,
quarks, object recognition, textmining

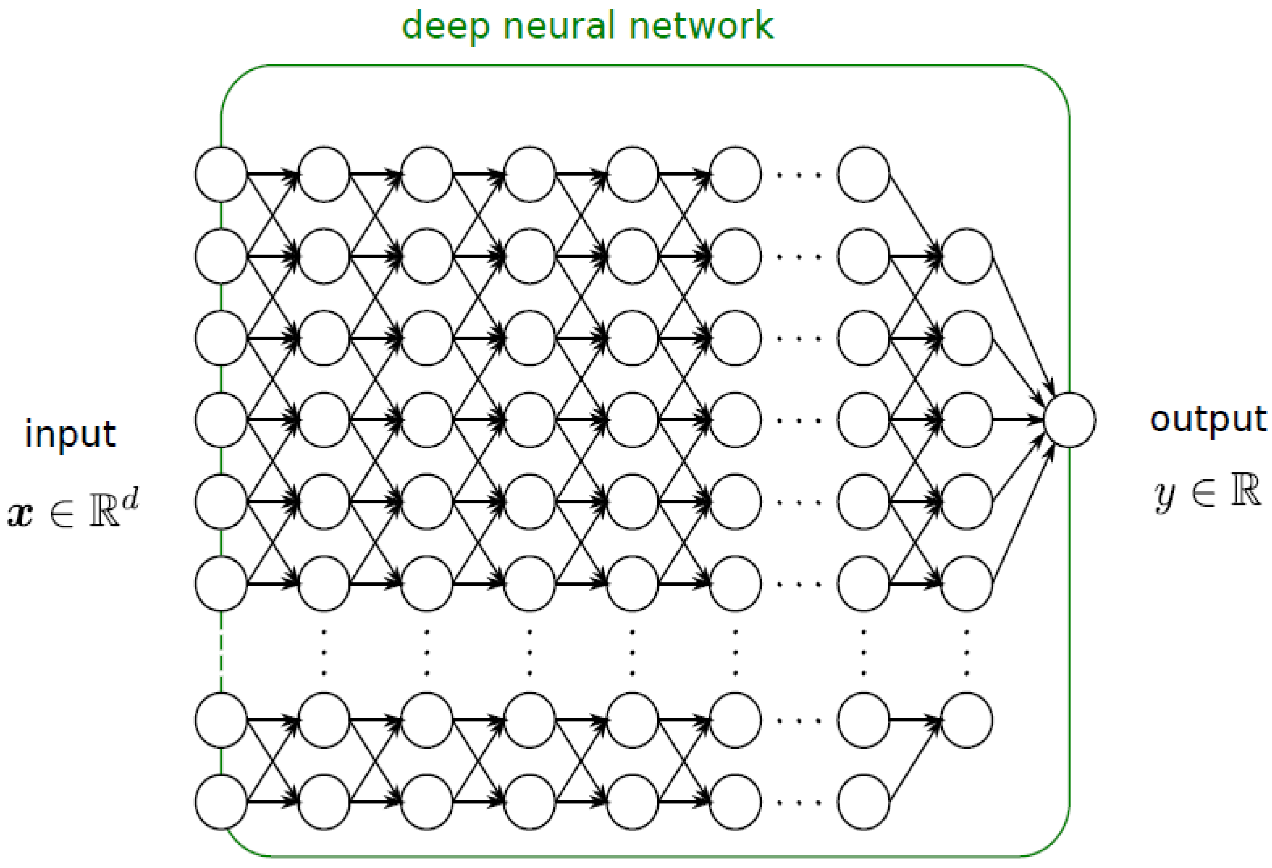
Neural Networks

What is a deep network?



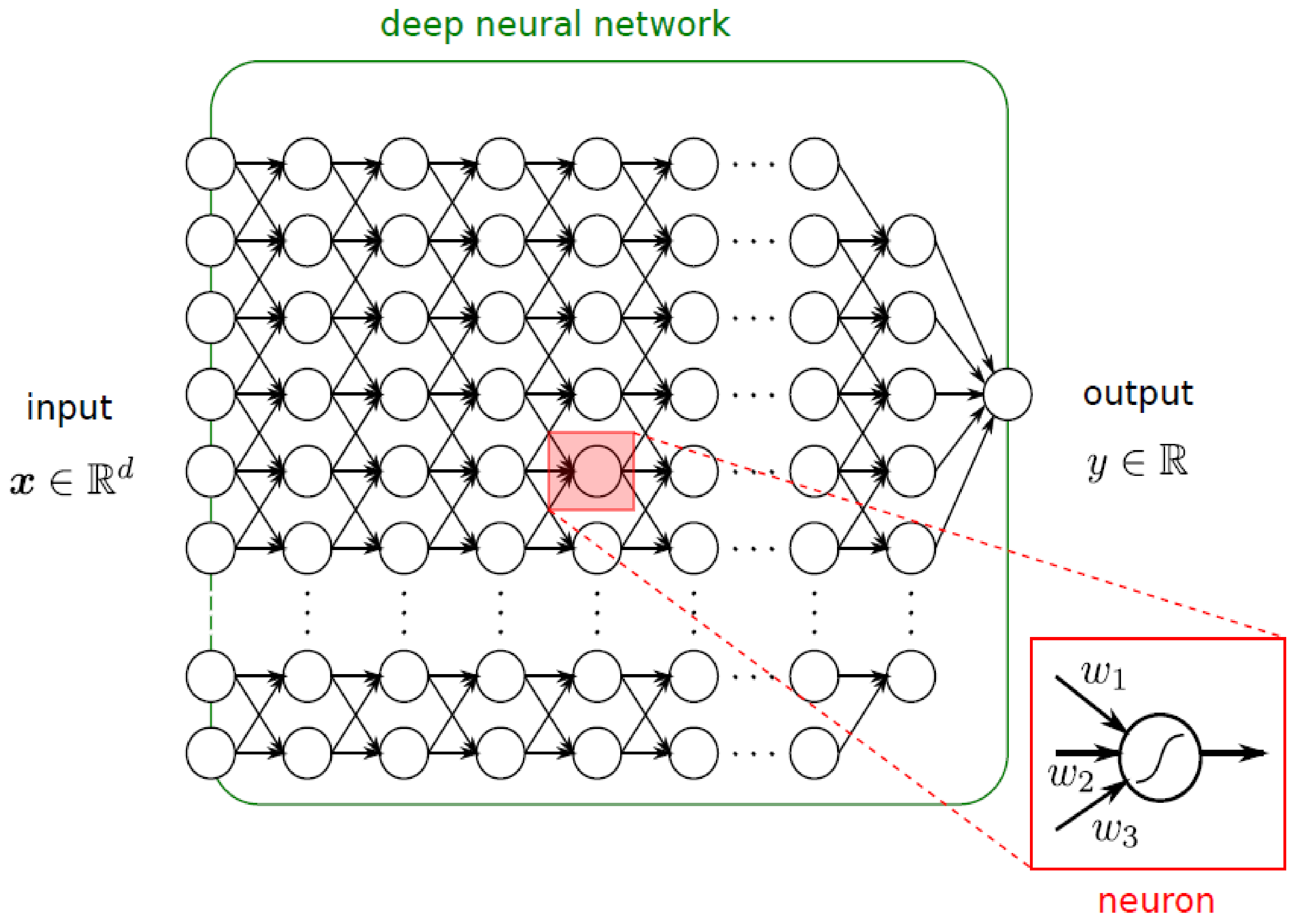
- Complex *nonlinear* function between *input* and *output*.

What is a deep network?



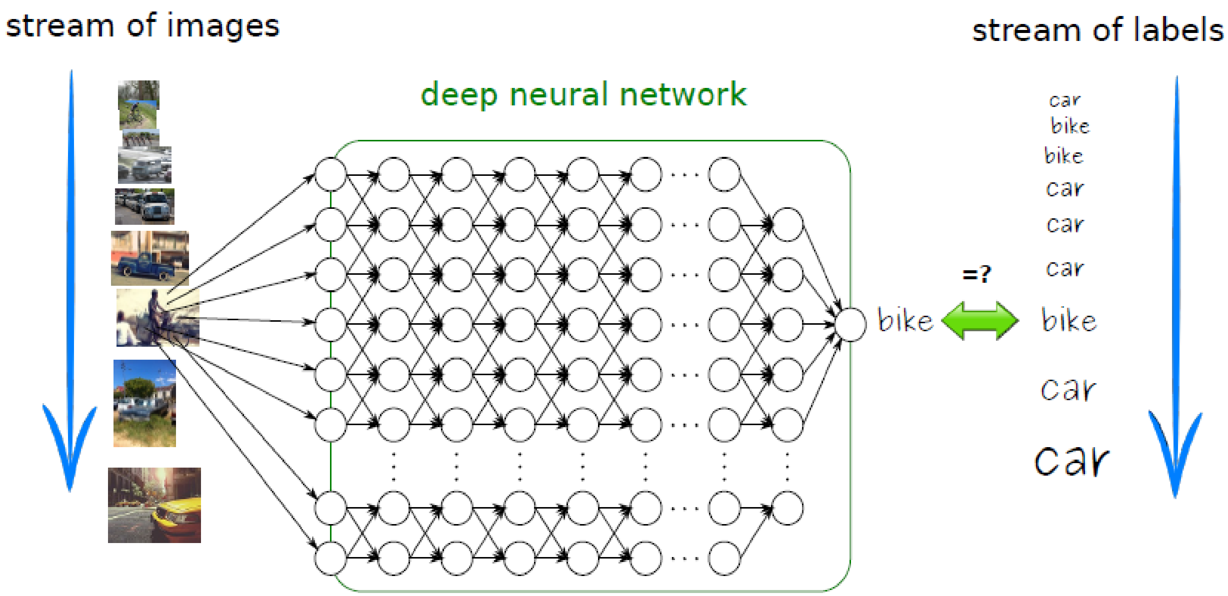
- Realized by a composition of many simple processing units called neurons.

What is a deep network?



- Neuron applies a nonlinear function to its input.
- Examples of functions: hyperbolic tangent, rectification.

Training a (deep) Neural Network



- Deep networks are trained one example at a time.
- If the output differs from the label, the error signal is backpropagated in the network to adapt the parameters.
- Over time, parameters evolve to form a model that fits the data well.

Training a (deep) Neural Network

- **Idea:** Follow the gradient of error E w.r.t. parameters w_{ij}.
- For a neural network with neuron equation

$$x_j = \sigma(a_j) \quad \text{where} \quad a_j = \sum_i x_i w_{ij},$$

  the update direction of the *whole* network can be computed using only *two* rules:

$$\Delta w_{ij} = x_i \delta_j \qquad \text{(update rule)}$$

$$\delta_i = \sigma'(a_i) \sum_j w_{ij} \delta_j \qquad \text{(backpropagation rule)}$$

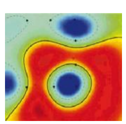
  applied uniformly to *all* neurons in the neural network, in a *backward* pass.
- **Deep networks can be trained on GPUs (10-100x performance boost!)**


KEEP
CALM
USE
MACHINE
LEARNING
person
dog
chair
person
car
helmet
motorcycle
C
Ni
ĤΨ = EΨ

ML4Physics @IPAM 2011: Part I

Klaus-Robert Müller, Matthias Rupp

Anatole von Lilienfeld and Alexandre Tkachenko



Machine Learning for chemical compound space

Ansatz:

$$\{Z_I, \mathbf{R}_I\} \xrightarrow{\text{ML}} E$$

instead of

$$\hat{H}(\{Z_I, \mathbf{R}_I\}) \stackrel{\Psi}{\longmapsto} E$$

$$\hat{H}\Psi = E\Psi$$

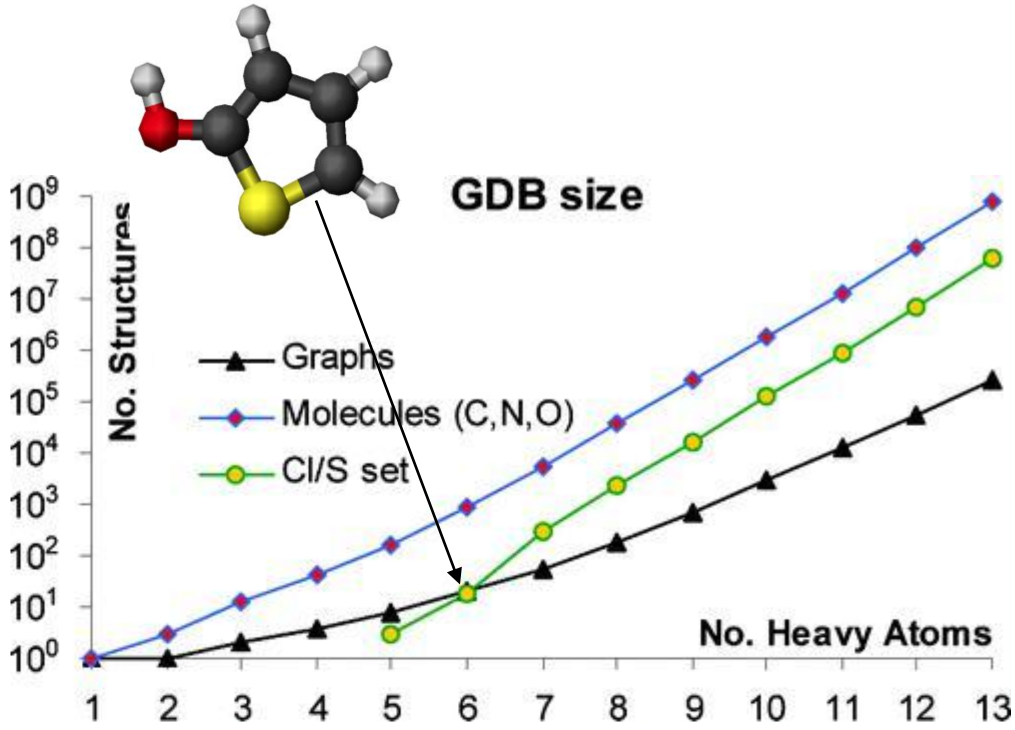
[from von Lilienfeld]

The data

GDB-13 database of all organic molecules (within stability & synthetic constraints) of 13 heavy atoms or less: 0.9B compounds

Table 1. Structure Generation Statistics for GDB-13

nodes[a]	graphs[b]	GDB[c]	CI/S[d]	CPU time (h)[e]
1	1	1	0	0.00
2	1	3	0	0.00
3	2	12	0	0.00
4	4	43	0	0.00
5	8	155	3	0.01
6	20	934	19	0.02
7	57	5726	315	0.05
8	194	37151	2438	0.33
9	706	255542	17056	2.68
10	2831	1784626	130465	25.26
11	12011	12961686	938704	223.49
12	53789	99821343	7240108	3023.79
13	250268	795244451	59027533	36606.45
Total	**319892**	**910111673**	**67356641**	**39882.08**



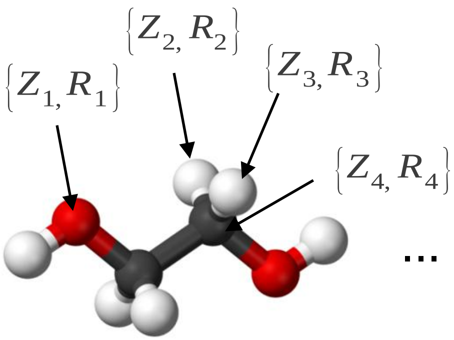
Blum & Reymond, *JACS* (2009)

[from von Lilienfeld]

Coulomb representation of molecules

$$M_{ii} = Z_i^{2.4}$$

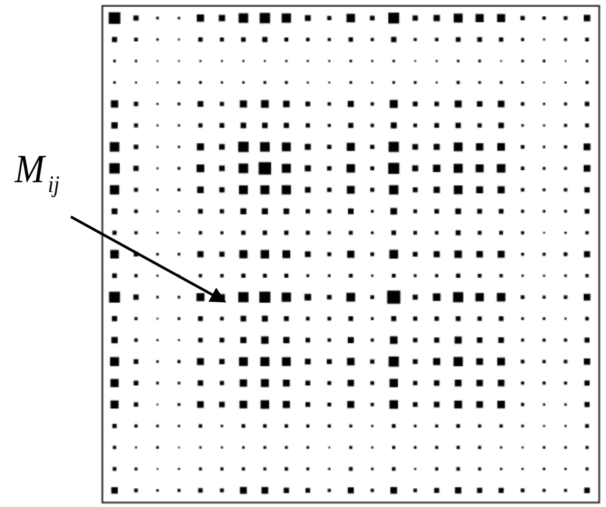
$$M_{ij} = \frac{Z_i Z_j}{\left\| R_i - R_j \right\|}$$



+ phantom atoms

$\{0, R_{21}\}$ $\{0, R_{22}\}$ $\{0, R_{23}\}$

$M \in {}^{23\times 23}$



Coulomb Matrix (Rupp, Müller et al 2012, PRL)

$$d(\mathbf{M}, \mathbf{M}') = \sqrt{\sum_{IJ} |M_{IJ} - M'_{IJ}|^2}$$

Kernel ridge regression

Distances between **M** define Gaussian kernel matrix **K**

$$k(\mathbf{M},\mathbf{M}') = \exp\left(-\frac{d(\mathbf{M},\mathbf{M}')^2}{2\sigma^2}\right)$$

Predict energy as sum over weighted Gaussians

$$E^{est}(\mathbf{M}) = \sum_i \alpha_i k(\mathbf{M},\mathbf{M}_i) + b$$

using weights that minimize error in training set

$$\min_{\alpha} \quad \sum_i \left(E^{est}(\mathbf{M}_i) - E_i^{ref}\right)^2 + \lambda \sum_i \alpha_i^2$$

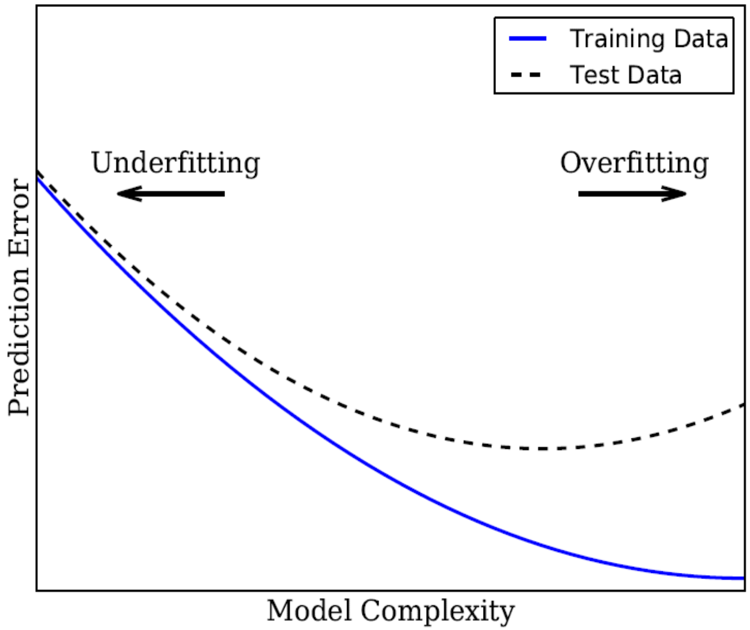
$$\alpha = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{E}^{ref}$$

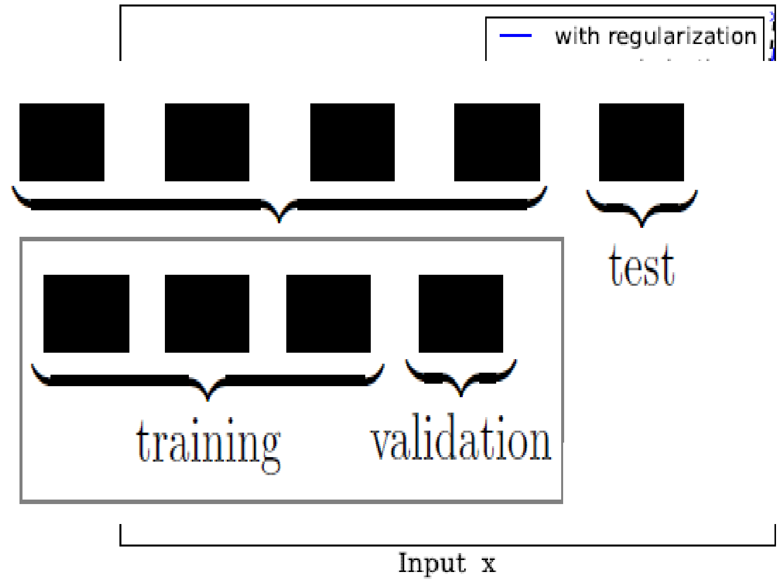
Exact solution

As many parameters as molecules + 2 global parameters, characteristic length-scale or kT of system (σ), and noise-level (λ)

[from von Lilienfeld]

Remarks on Generalization and Model Selection in ML





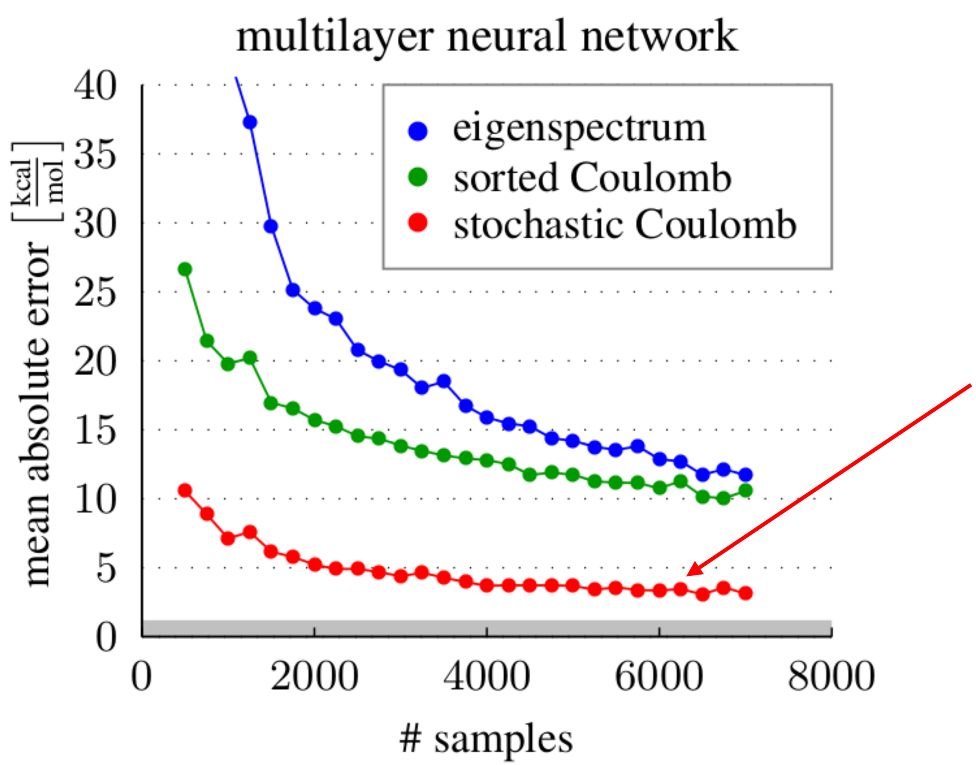
Kernel Ridge Regression Model

$$E^{est}(\mathbf{M}) = \sum_i \alpha_i k(\mathbf{M}, \mathbf{M}_i) + b$$

$$\min_{\alpha} \quad \sum_i \left(E^{est}(\mathbf{M}_i) - E_i^{ref}\right)^2 + \lambda \sum_i \alpha_i^2$$

$$\alpha = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{E}^{ref}$$

Results



March 2012
Rupp et al., PRL
9.99 kcal/mol
(kernels + eigenspectrum)

December 2012
Montavon et al., NIPS
3.51 kcal/mol
(Neural nets + Coulomb sets)

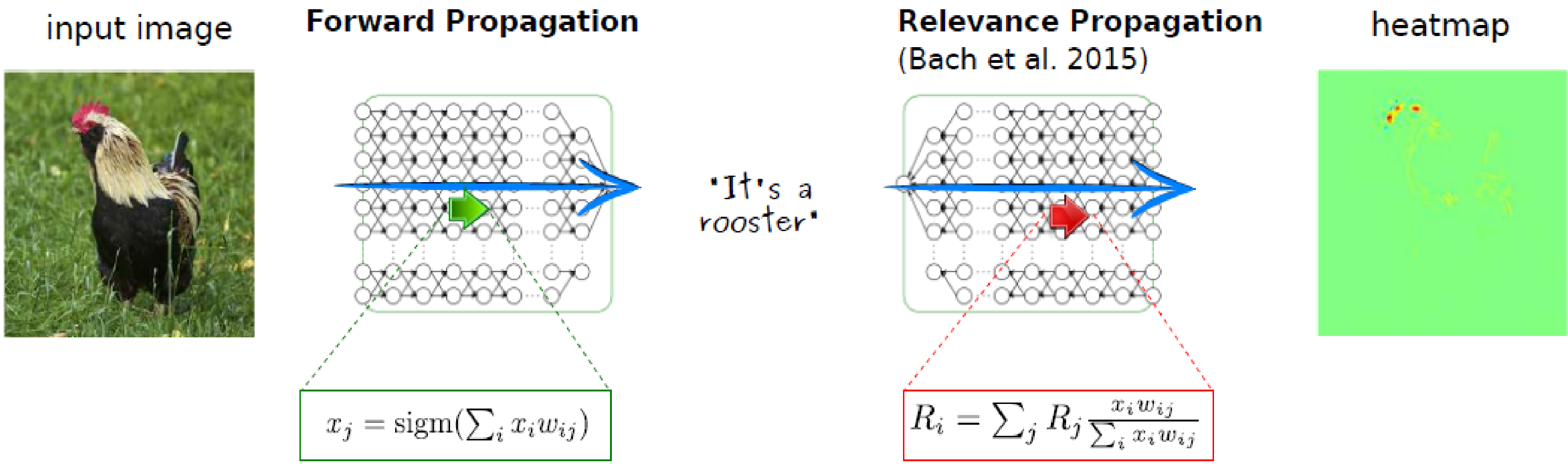
2015 Tkatchenko 1.3kcal/mol

Prediction considered chemically accurate when MAE is below **1 kcal/mol**

Dataset available at http://quantum-machine.org

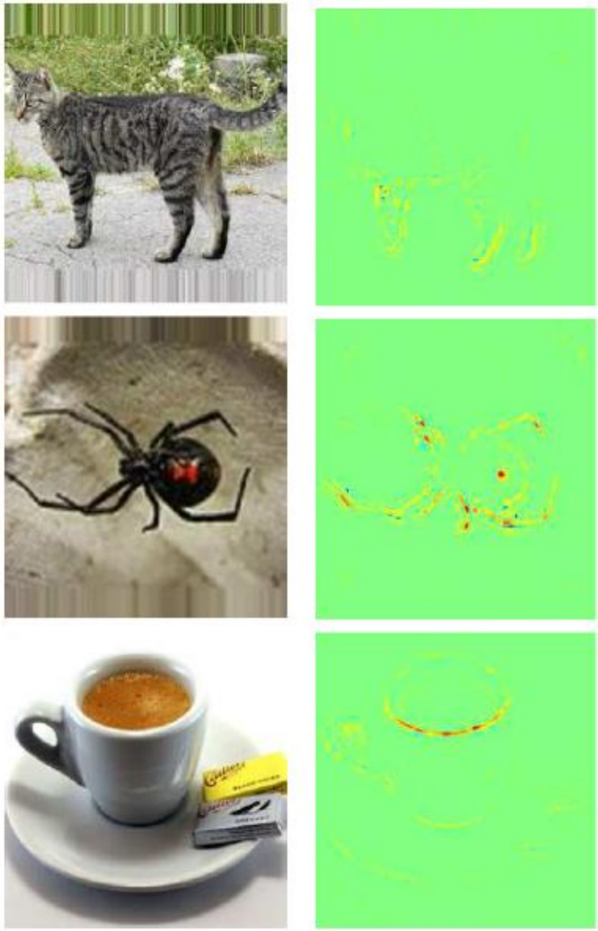
Perspectives

Explaining Predictions Pixel-wise



- The total relevance $\sum_p R_p$ (number of red pixels in the heatmap) corresponds to the amount of evidence $f(\mathbf{x})$ for the predicted class. ($\Rightarrow$ Relevance does not get *lost* or *created* out of nothing.)
- This equivalence is ensured by the *layer-wise conservation* property of the relevance propagation formula.

Explaining Predictions Pixel-wise



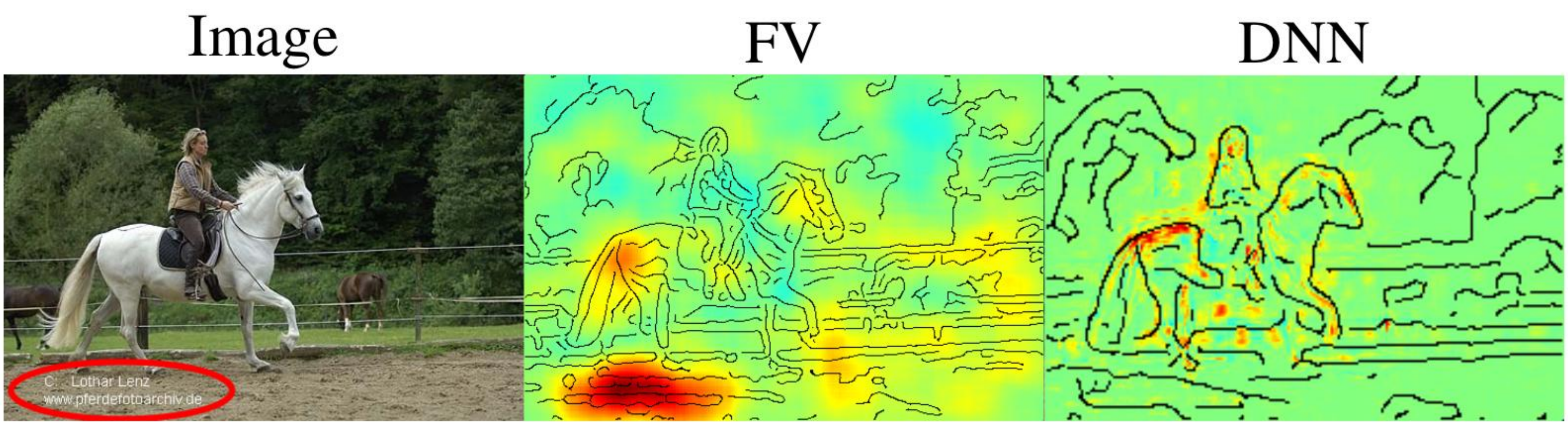
Neural networks

Kernel methods

Application: Comparing Classifiers

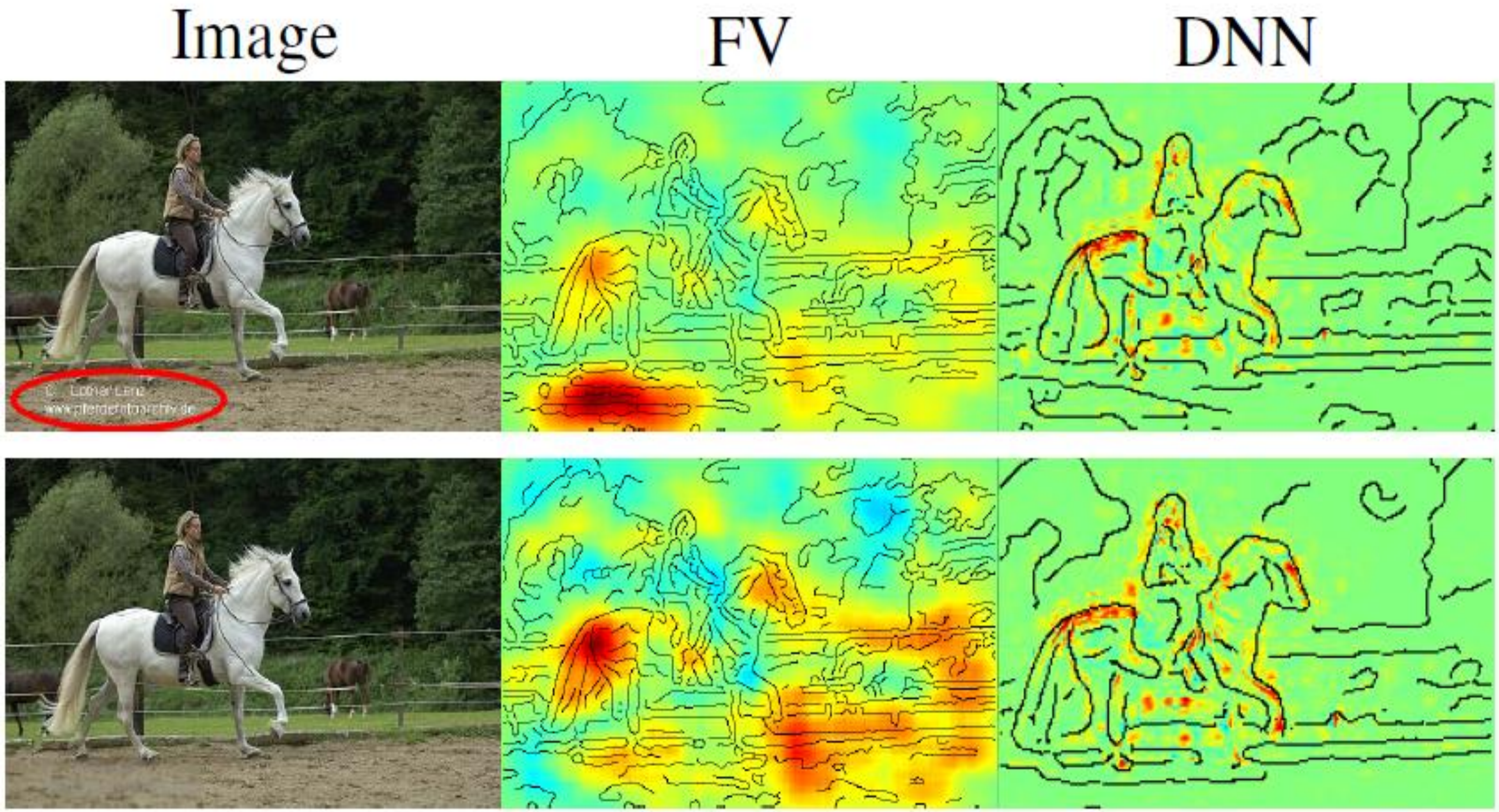
Test error for various classes:

	aeroplane	**bicycle**	**bird**	**boat**	**bottle**	**bus**	**car**
Fisher	79.08%	66.44%	45.90%	70.88%	27.64%	69.67%	80.96%
DeepNet	88.08%	79.69%	80.77%	77.20%	35.48%	72.71%	86.30%
	cat	**chair**	**cow**	**diningtable**	**dog**	**horse**	**motorbike**
Fisher	59.92%	51.92%	47.60%	58.06%	42.28%	80.45%	69.34%
DeepNet	81.10%	51.04%	61.10%	64.62%	76.17%	81.60%	79.33%
	person	**pottedplant**	**sheep**	**sofa**	**train**	**tvmonitor**	**mAP**
Fisher	85.10%	28.62%	49.58%	49.31%	82.71%	54.33%	59.99%
DeepNet	92.43%	49.99%	74.04%	49.48%	87.07%	67.08%	72.12%



[Bach et al. CVPR 2016]

Understanding Models is only possible if we explain

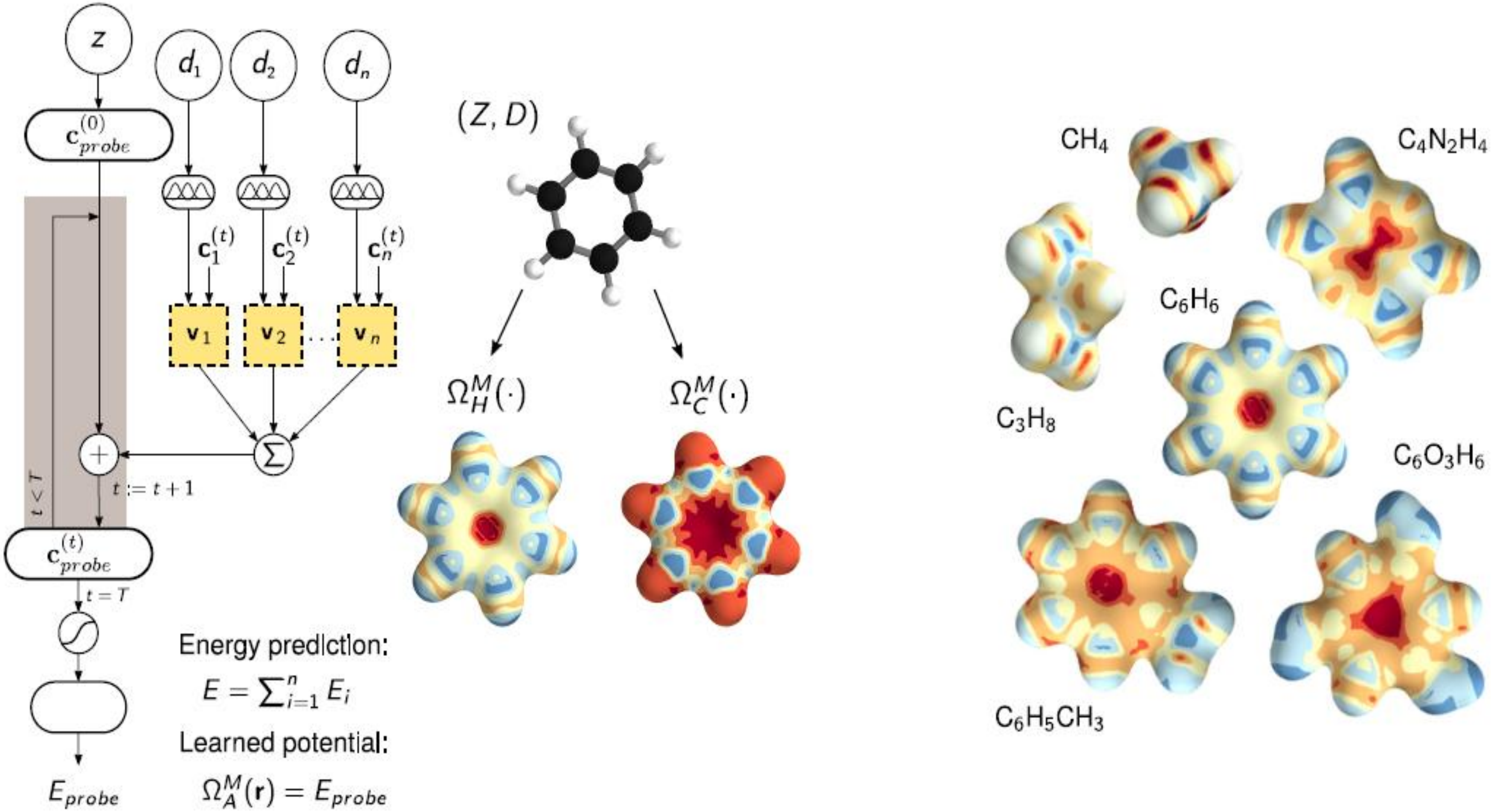


Fisher

Neural networks

Neural Networks for Molecules revisted

Quantum Chemical Insights



[Schütt et al. under review]

Conclusion

- Machine Learning & modern data analysis is of central importance in daily life
- input to ML algorithms can be vectors, matrices, graphs, strings, tensors etc.
- Representation is essential ! Modelselection, Optimization.
- ML 4 XC, ML for reaction transitions, ML for formation energy prediction etc.
- ML challenges from Physics: no noise, high dimensional systems, functionals …
- challenge: learn for Physics from ML representation: towards better understanding

See also: www.quantum-machine.org

Further Reading

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7) e0130140.

Bach, S., Binder, A., Montavon, G., Müller, K.-R. & Samek, W. (2016). Analyzing Classifiers: Fisher Vectors and Deep Neural Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

Lapuschkin, S., Binder, A., Montavon, G., Müller, K. R., & Samek, W. (2016). The LRP Toolbox for Artificial Neural Networks. Journal of Machine Learning Research, 17(114), 1-5.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. R. (2010). How to explain individual classification decisions. The Journal of Machine Learning Research, 11, 1803-1831.

Braun, M. L., Buhmann, J. M., & Müller, K. R. (2008). On relevant dimensions in kernel feature spaces. *The Journal of Machine Learning Research*, *9*, 1875-1908.

Müller, K-R., Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernhard Scholkopf. "An introduction to kernel-based learning algorithms." IEEE transactions on neural networks 12, no. 2 (2001): 181-201.

Montavon, G., Braun, M. L., & Müller, K. R. (2011). Kernel analysis of deep networks. *The Journal of Machine Learning Research*, *12*, 2563-2581.

Montavon, G., Orr, G. & Müller, K. R. (2012). *Neural Networks: Tricks of the Trade,* Springer *LNCS 7700*. Berlin Heidelberg.

Montavon, Grégoire, Katja Hansen, Siamac Fazli, Matthias Rupp, Franziska Biegler, Andreas Ziehe, Alexandre Tkatchenko, Anatole V. Lilienfeld, and Klaus-Robert Müller. "Learning invariant representations of molecules for atomization energy prediction." In *Advances in Neural Information Processing Systems*, pp. 440-448. 2012.

Further Reading

Montavon, Grégoire, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. "Machine learning of molecular electronic properties in chemical compound space." New Journal of Physics 15, no. 9 (2013): 095003.

Montavon, G., Braun, M., Krueger, T., & Muller, K. R. (2013). Analyzing local structure in kernel-based learning: Explanation, complexity, and reliability assessment. *IEEE Signal Processing Magazine, 30*(4), 62-74.

Pozun, Z. D., Hansen, K., Sheppard, D., Rupp, M., Müller, K. R., & Henkelman, G., Optimizing transition states via kernel-based machine learning. The Journal of chemical physics, 136(17), 174101. 2012 .

Rupp, M., Tkatchenko, A., Müller, K. R., & von Lilienfeld, O. A. (2012). Fast and accurate modeling of molecular atomization energies with machine learning. Physical review letters, 108(5), 058301.

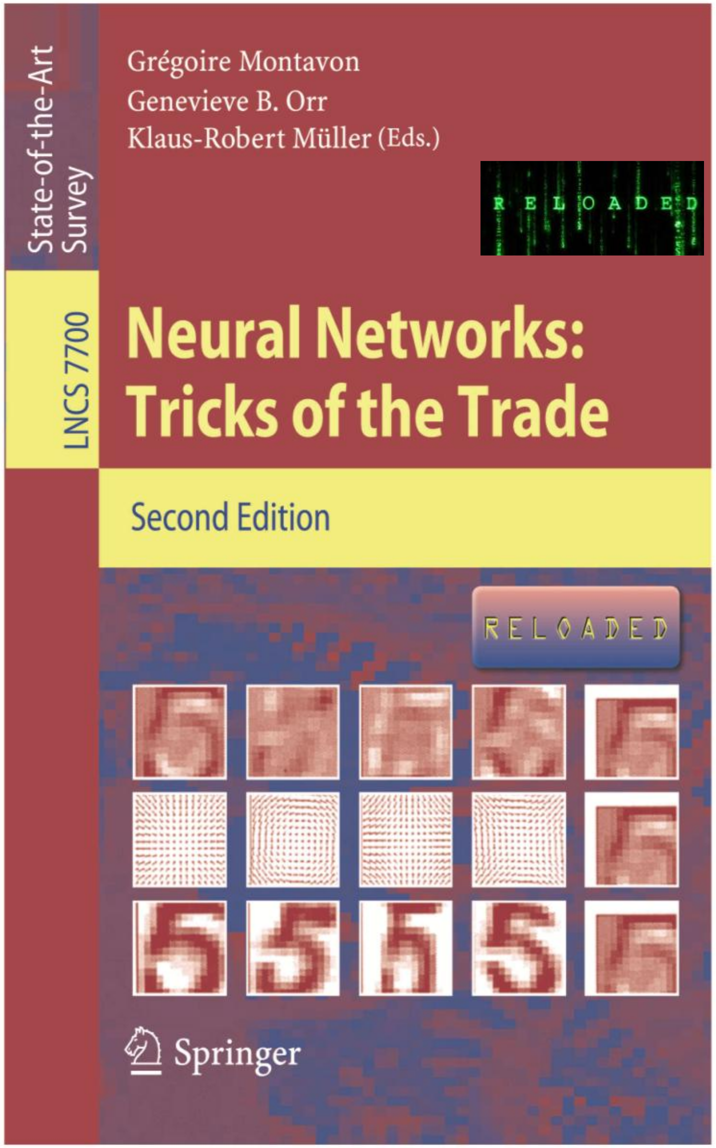
Schölkopf, B., Smola, A., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. Neural computation, 10(5), 1299-1319.

K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K. R. Müller, and E. K. U. Gross, How to represent crystal structures for machine learning: Towards fast prediction of electronic properties, Phys. Rev. B 89, 205118 (2014)

Snyder, J. C., Rupp, M., Hansen, K., Müller, K. R., & Burke, K. Finding density functionals with machine learning. Physical review letters, 108(25), 253002. 2012.

Sturm, I., Bach, S., Samek, W., & Müller, K. R. (2016). Interpretable Deep Neural Networks for Single-Trial EEG Classification. *arXiv preprint arXiv:1604.08201*.

Vapnik, VN, The nature of statistical learning theory, Springer. 1995


State-of-the-Art Survey
LNCS 7700
Grégoire Montavon
Genevieve B. Orr
Klaus-Robert Müller (Eds.)
RELOADED
Neural Networks:
Tricks of the Trade
Second Edition
RELOADED
Springer