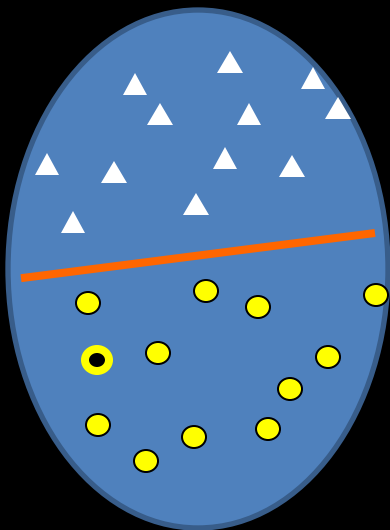


Slide Set 2

Jayadeva

- Which of the two is likely to have a lower test set error ?

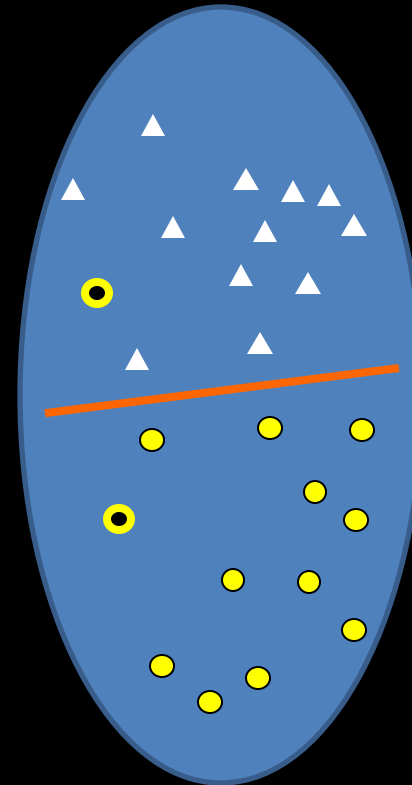


R^d , large d

Same support vectors,

same margin,

**same number of
training samples**



Notion of machine complexity

- ◆ Use the simplest solution for a given task
- ◆ Use the one that would be least likely to fail on a given task
- ◆ For a learning machine, the analogy is : which machine is less likely to fail on a prediction task, i.e. generalize better



Sir Mervs/ CC-BY-SA-2.0

**For learning machines,
complexity is measured by
the VC dimension**

Using all available degrees of freedom leads to overfitting !

Complexity of Learning

- Concept is a function f^* describing a mapping from X (input space) to a Boolean alphabet
- Concept class is a collection of concepts over the environment
- Prob. distribution specifies how likely an example from X is drawn for learning
- Given $0 \leq \varepsilon, \delta \leq 1$, the goal of a learning algorithm is to produce, with prob. of at least $(1 - \delta)$, a learning machine that will correctly predict $f^*(x)$ for at least a fraction $(1 - \varepsilon)$ of future (test) exemplars.

Some resources

- Vladimir N. Vapnik, "An Overview of Statistical Learning Theory", IEEE Trans. Neural Networks, 1999.
- Burgess's SVM tutorial:
<https://link.springer.com/article/10.1023/A:1009715923555>
- <https://sites.astro.caltech.edu/~george/aybi199/AMooreTutorials/vcdim.ppt>

- Given $0 \leq \varepsilon, \delta \leq 1$, the goal of a learning algorithm is to produce, with prob. of at least $(1 - \delta)$, a learning machine that will correctly predict $f^*(x)$ for at least a fraction $(1 - \varepsilon)$ of future (test) exemplars.
- The function realized by the learning machine is f , that approximates f^* .
- Learning is accomplished using exemplars $(x, f^*(x))$, $x \in X$, according to *any* distribution.
- A correct prediction based on learning from exemplars is called a valid generalization.

- A function is **efficiently learnable** if and only if \exists a learning algorithm with the following properties:
- The algorithm run time is polynomial in ε^{-1} , δ^{-1} , n
- For every sampling prob. distribution p , the algorithm produces, in polynomial time, with prob. of at least $(1 - \delta)$, f s.t. $f = f^*$ for at least $(1 - \varepsilon)$ future exemplars.
- The confidence parameter δ accounts for outlier distributions or rare events.

- Efficient learning
- Distribution Free learning
- In practice, performance does depend on distribution
- Boundary exemplars carry more information
- **Loading** (Judd)
- Decision Problem, NP-complete for many classes of networks
- NP-complete even for 2/3 case
- Example: 2, 1 network, simple in linearly separable case, not otherwise

- **Generalizability of Learning**
- How well does a learning machine learn using N exemplars
- Depends on no. of learnable parameters in the network and no. of parameters necessary to describe a function
- What is the guarantee that learning the training samples well will ensure good prediction on test samples ?

- **VC dimension**
- What is the number N of training exemplars required to pin down all free parameters in a learning machine exactly ?
- Training samples are drawn using a prob. distribution p .
- Find f s.t. with a prob. of at least $(1-\delta)$, $f = f^*$ for at least a fraction $(1-\varepsilon)$ of future exemplars drawn from the same distribution p .
- Generalization prob. $G(f) = 1 - e(f) = \text{Prob. } (f(x) = f^*(x))$ for x drawn using p .

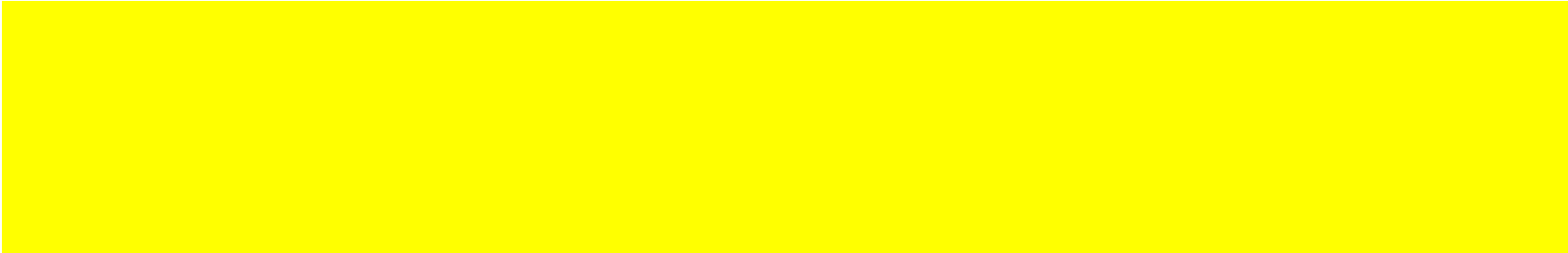
- Agreement set $A = \{x \in X \mid f(x) = f^*(x)\}$
- Consider a sample of independent and identically distributed random variables
- value of a particular random variable is an assignment, like the result of a coin toss
- Event 1, prob = p , other event : $(1-p)$
- Bernoulli process
- With N exemplars, consider each exemplar to belong to a Bernoulli process
- $g_N(f)$, $G(f)$

- Would like that the chance that $\exists f : g_N(f)$ of the specific “f” learnt by the machine is significantly different from $G(f)$ is small.
- This prob. of difference should be small in the worst case.
- $g_N(f)$ should converge uniformly in prob. to $G(f)$



- Conversely, $(1-\delta)$ is our confidence that



- Good learning requires that $g_N(f)$ be maximized and $g_N(f)$ should converge uniformly to $G(f)$
 - *Vapnik and Chervonenkis*
- 

- $\Delta_f(N)$ measures the size of F , i.e. the function representation capacity of the network.
- Also called the growth function

- What if we can ensure good learning on training samples ?
- If the training error $\leq (1 - \gamma)\epsilon$, then the probability that the test error exceeds ϵ is upper bounded by

$$8\Delta_f(2N)\exp\left(\frac{-\gamma^2\epsilon N}{4}\right)$$

Vapnik's Risk Formula

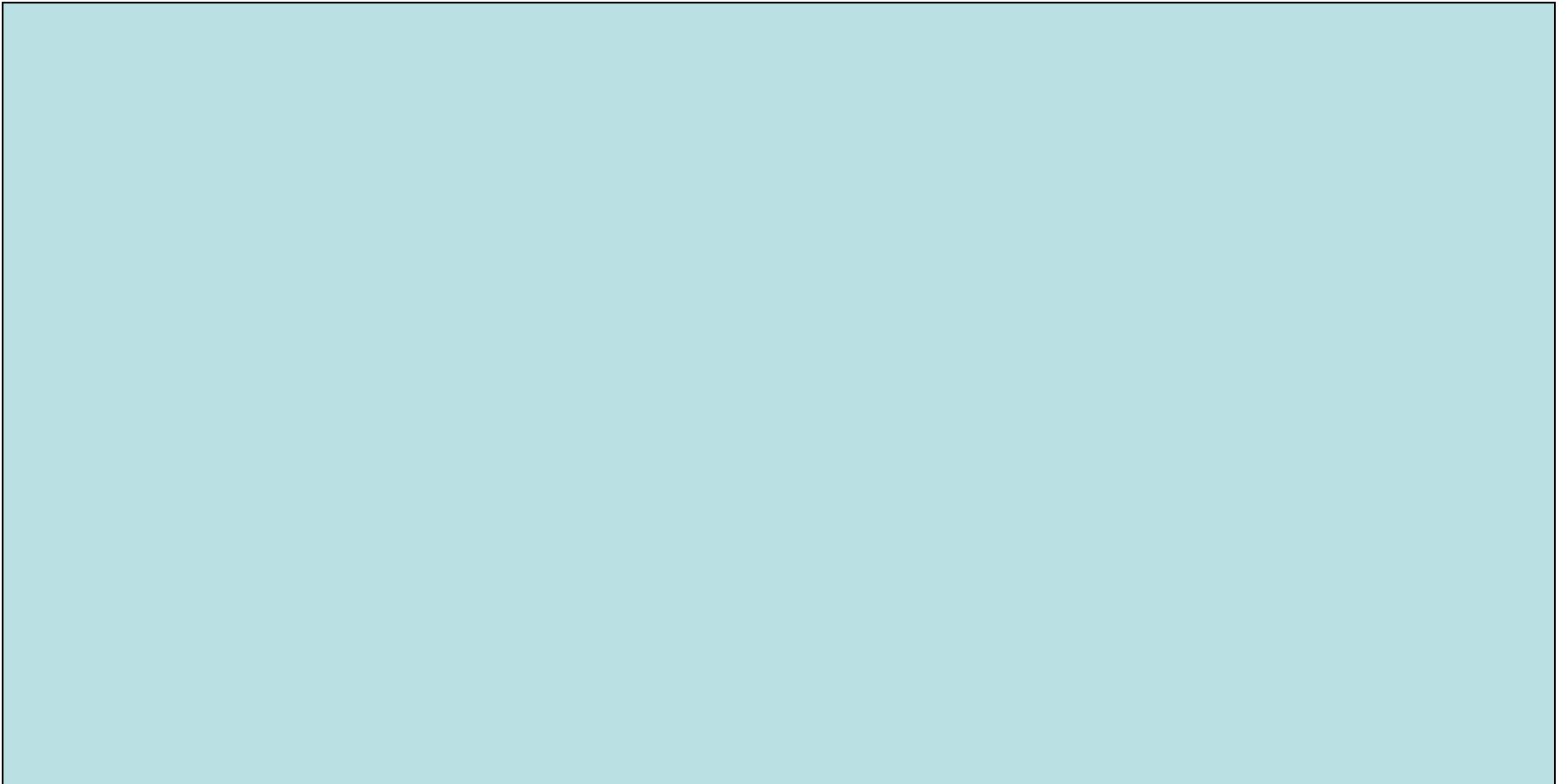
With probability $(1-\delta)$,

$$err_{true} \leq err_{train} + \frac{B\epsilon}{2} \left(1 + \sqrt{1 + \frac{4err_{train}}{B\epsilon}} \right)$$

$$where \epsilon = \left(\frac{4}{M} \right) \left[VC \left(\ln \left(\frac{2M}{VC} \right) + 1 \right) - \ln \delta \right]$$

- VC is the VC dimension of a classifier, M is the number of samples.
- Neural Nets: Pruning reduces VC.
- SVMs: Fewer Support Vectors usually leads to better generalization (but SVMs can have $VC = \infty$)

- *Quiz*



The Bias-Variance Decomposition (1)

- Recall the *expected squared loss*,

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \underbrace{\int \int \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt}_{\text{noise term}}$$

- where

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x}) dt.$$

<https://link.springer.com/article/10.1023/A:1009715923555>

- The second term of \mathbb{E} corresponds to the noise inherent in the random variable t .
- What about the first term?

The Bias-Variance Decomposition (2)

- Suppose we were given multiple data sets, each of size L . Any particular data set, \mathcal{D} , will give a particular function y . We then have

$$\begin{aligned} & \{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &\quad + 2\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}. \end{aligned}$$

The Bias-Variance Decomposition (3)

- Taking the expectation over \mathcal{D} yields

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \\ = \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}}. \end{aligned}$$

The Bias-Variance Decomposition (4)

- Thus we can write

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

- where

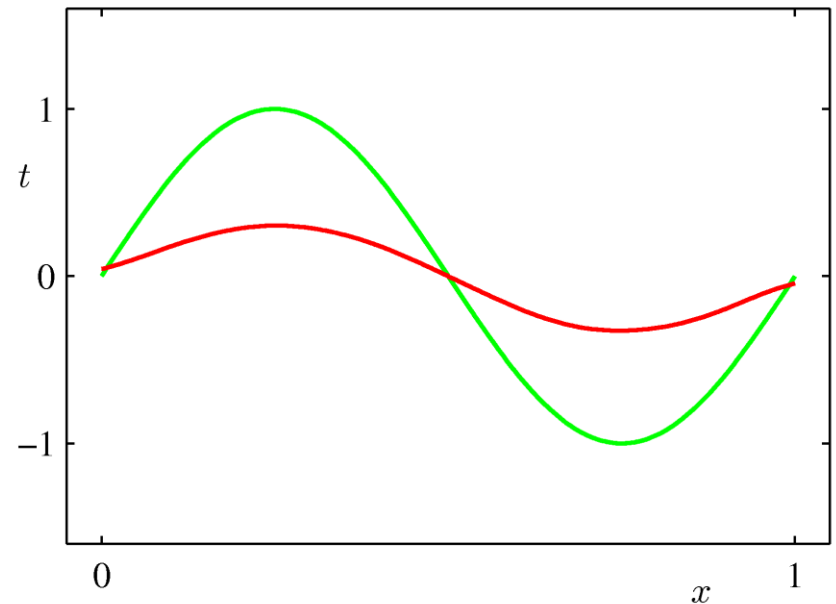
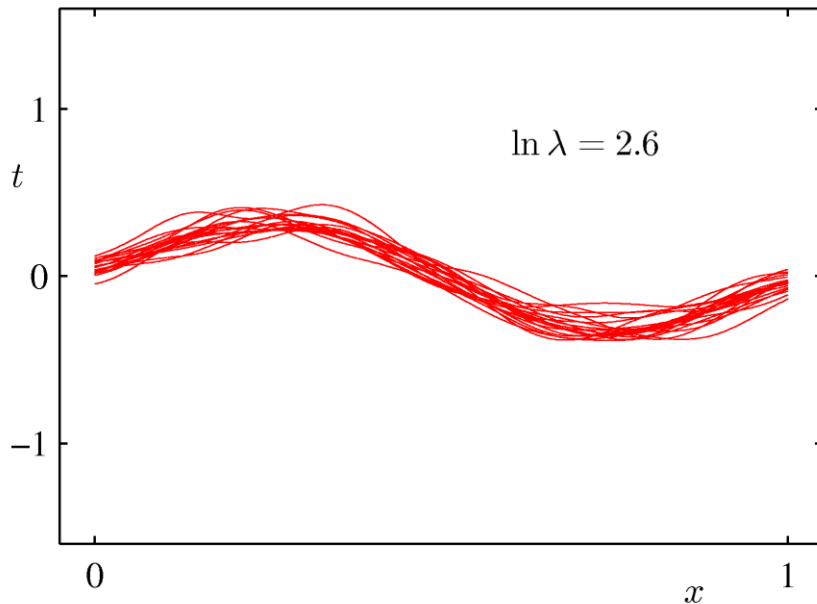
$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x}$$

$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt$$

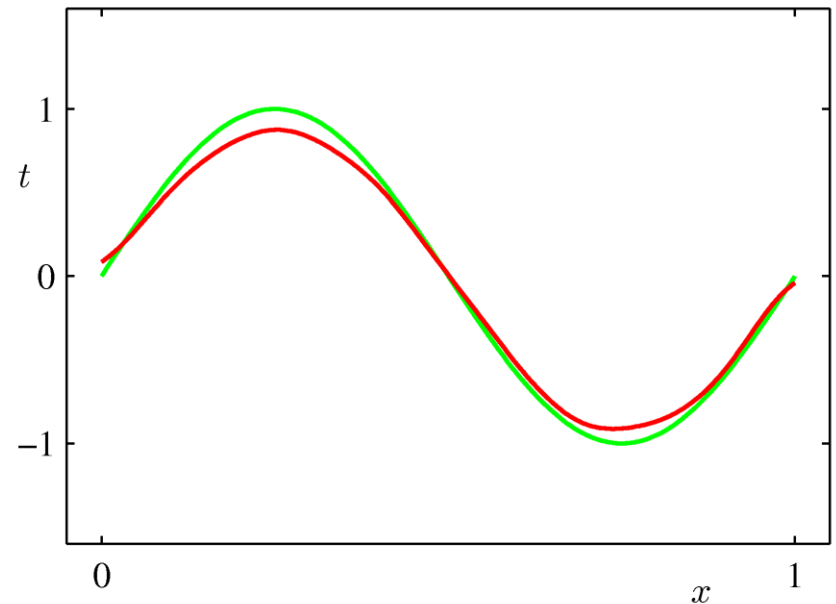
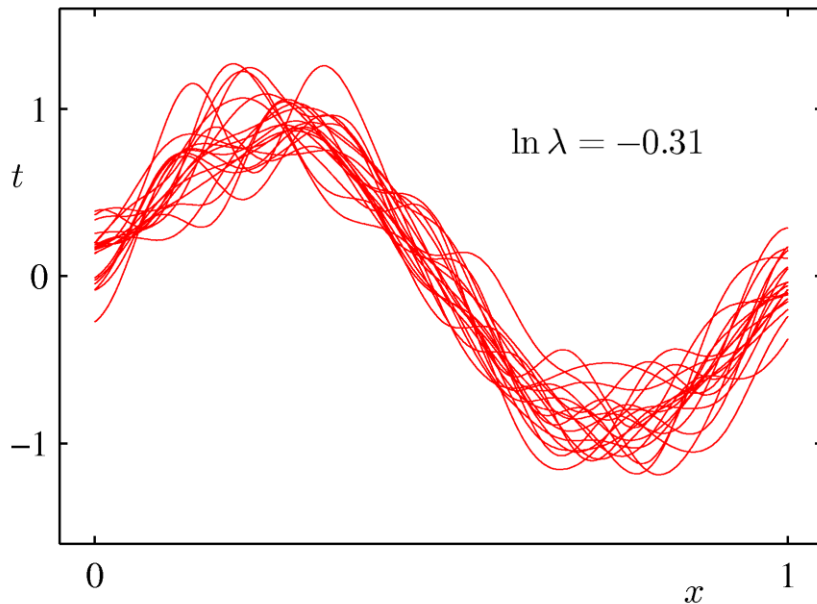
The Bias-Variance Decomposition (5)

- Example: 25 data sets from the sinusoidal, varying the degree of regularization.



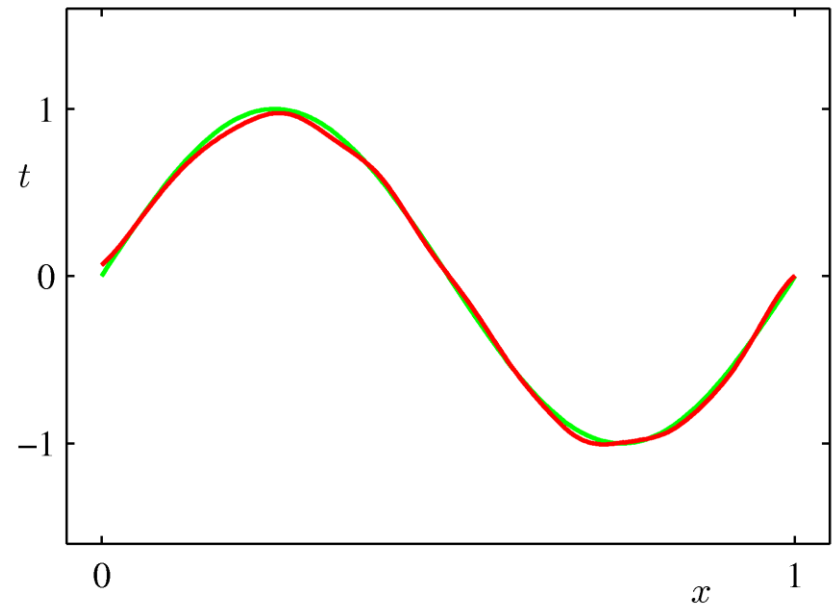
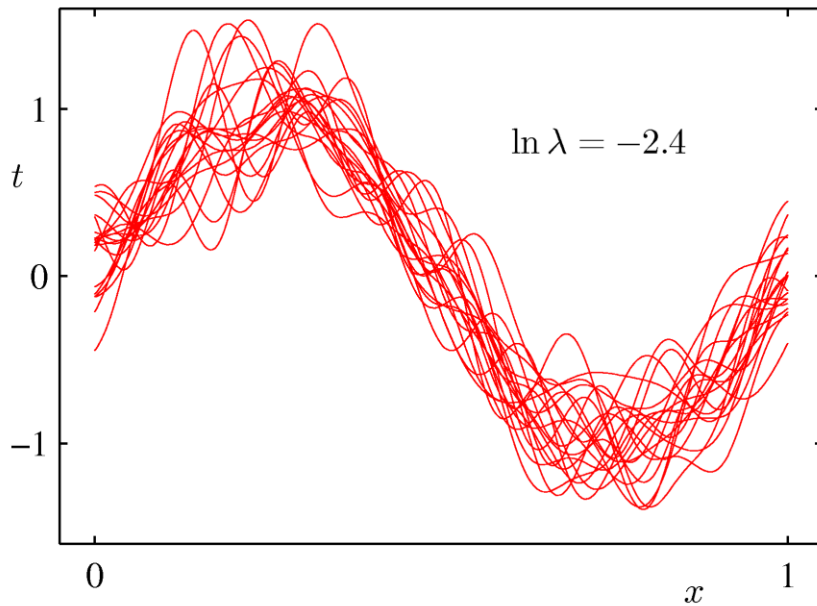
The Bias-Variance Decomposition (6)

- Example: 25 data sets from the sinusoidal, varying the degree of regularization



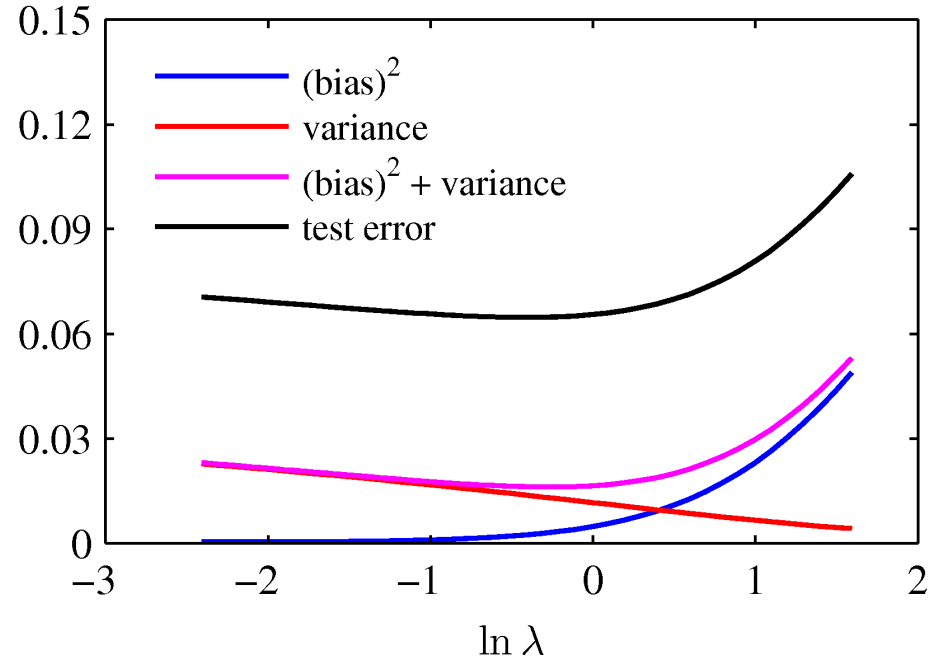
The Bias-Variance Decomposition (7)

- Example: 25 data sets from the sinusoidal, varying the degree of regularization.



The Bias-Variance Trade-off

- From these plots, we note that an over-regularized model (large λ) will have a high bias, while an under-regularized model (small λ) will have a high variance.

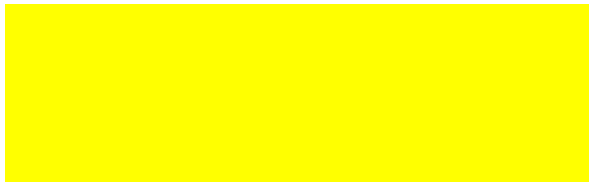


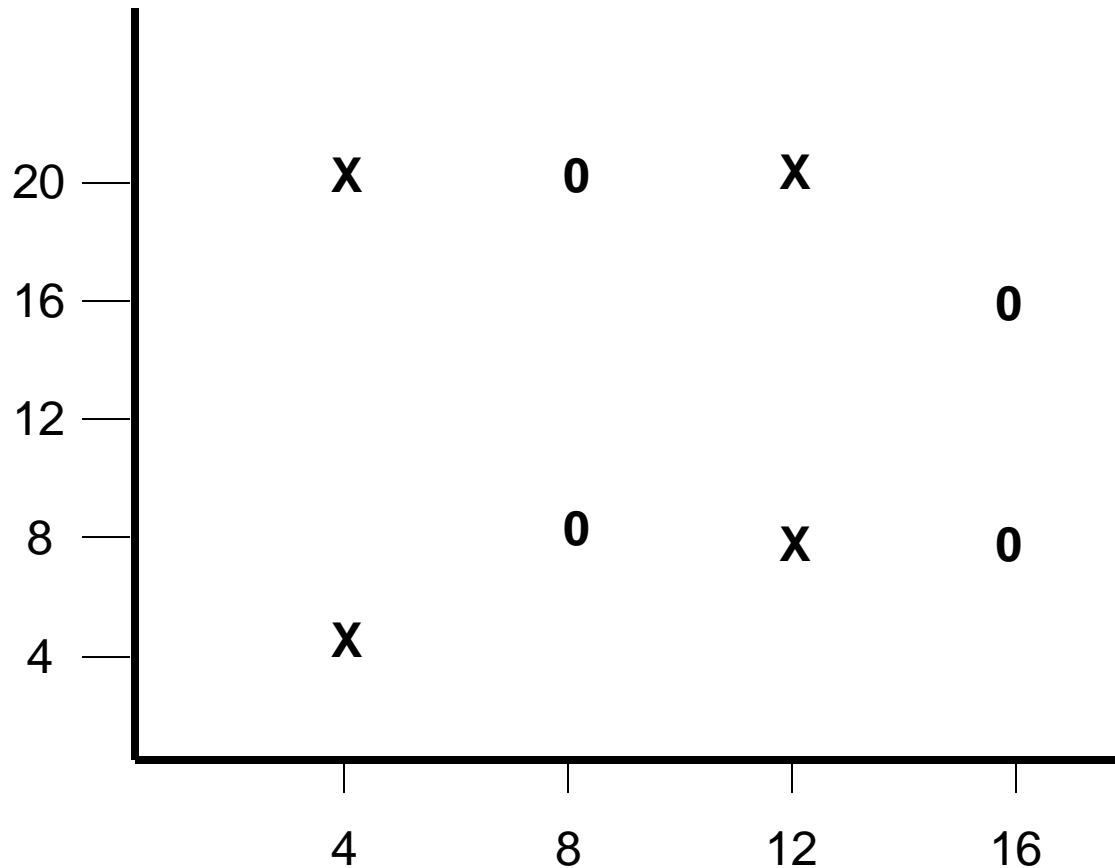
Generalization and VC dimension of SVMs

- Expected error = $R(\alpha)$ =



- Empirical error = $R_{\text{emp}}(\alpha)$ =



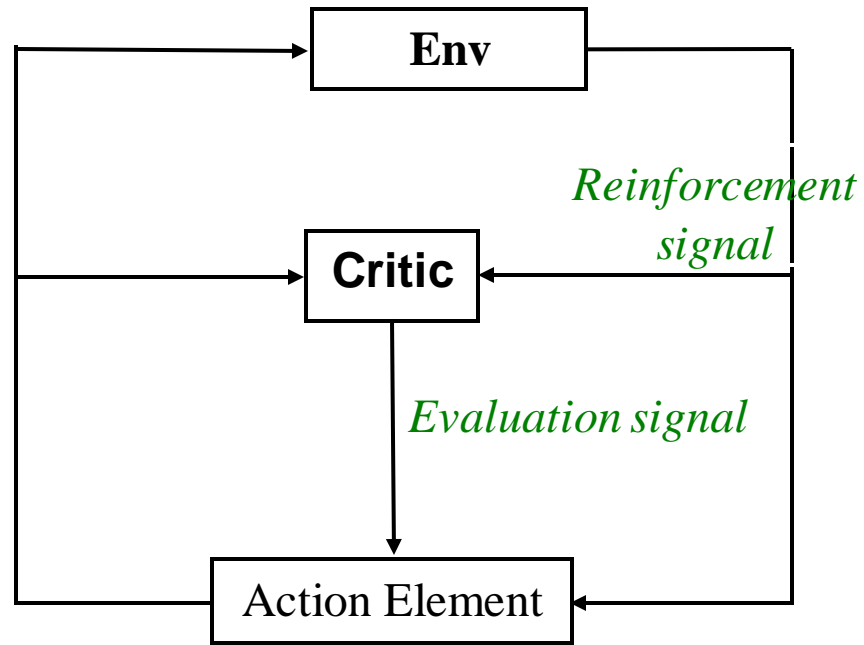


- Find a pyramidal delayed perceptron network by hand. Find all neurons and their weights. At each step, determine patterns in classes C1, C2, C3, and C4. Use as few neurons as possible.

Back-propagation Variants

- Pattern mode and batch mode training
- Stochastic Gradient (shuffle patterns)
- Termination Criteria
 - absolute squared error
 - absolute rate of change of squared error
 - gradient
- Bipolar activation function
- Weight initialization
- Weight decay
- Restart

- Fan in dependent weights = $\pm 2.4/f_i$
- Check input and output ranges
- Adjusting Learning Rates
- Network Architecture
- Training, Testing, and Validation Sets
- Cross Validation
- Leapfrog connections [Lang & Witbrock]



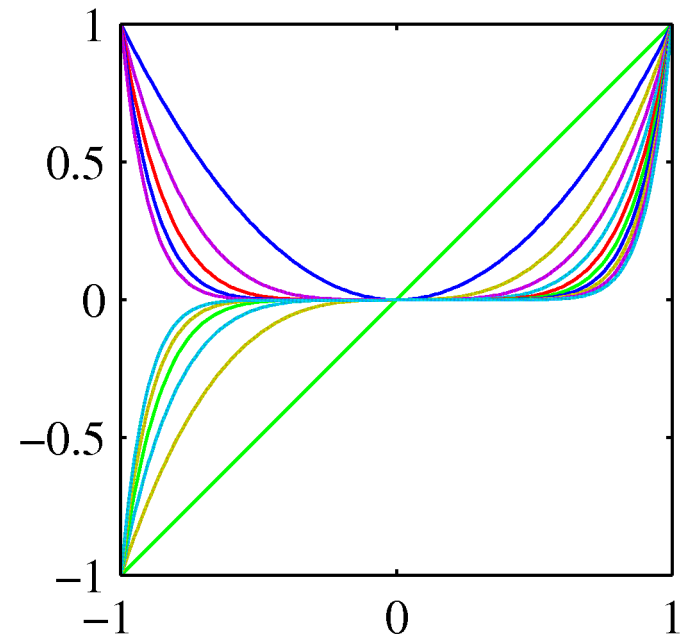
Reinforcement Learning

Linear Basis Function Models (3)

- Polynomial basis functions:

$$\phi_j(x) = x^j.$$

- These are global; a small change in x affect all basis functions.

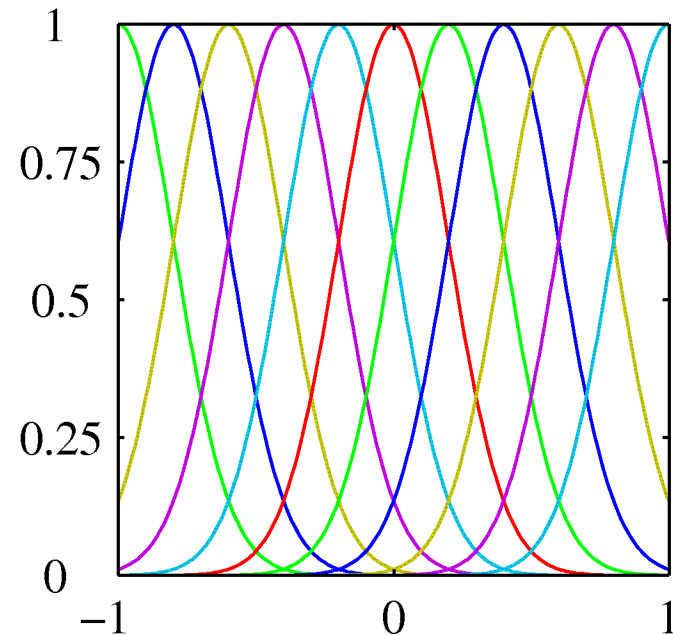


Linear Basis Function Models (4)

- Gaussian basis functions:

$$\phi_j(x) = \exp \left\{ -\frac{(x - \mu_j)^2}{2s^2} \right\}$$

- These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).



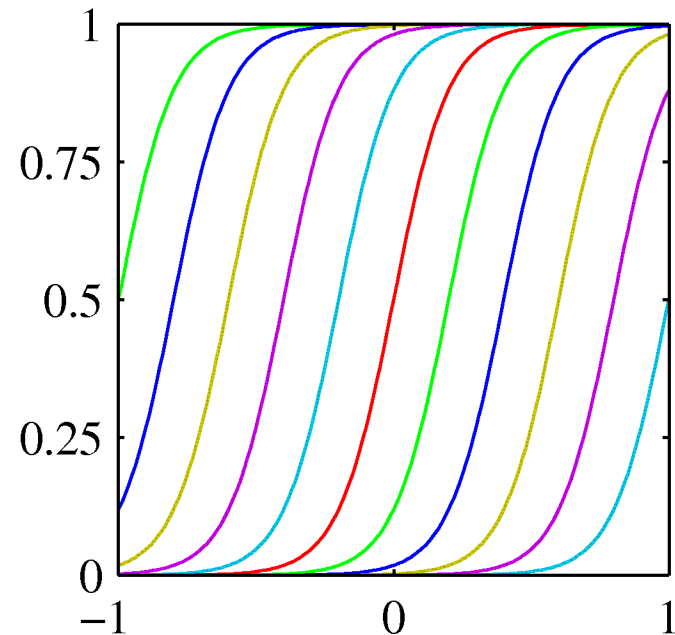
Linear Basis Function Models (5)

- Sigmoidal basis functions:

$$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

- where $\sigma(a) = \frac{1}{1 + \exp(-a)}$.

- Also these are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (slope).



The Bias-Variance Decomposition (1)

- Recall the *expected squared loss*,

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x} + \underbrace{\iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt}_{\text{noise}}$$

- where

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x}) dt.$$

- The second term of \mathbb{E} corresponds to the noise inherent in the random variable t .
- What about the first term?

The Bias-Variance Decomposition (2)

- Suppose we were given multiple data sets, each of size L . Any particular data set, \mathcal{D} , will give a particular function y . We then have

$$\begin{aligned} & \{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 \\ &\quad + 2\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}. \end{aligned}$$

The Bias-Variance Decomposition (3)

- Taking the expectation over \mathcal{D} yields

$$\begin{aligned} \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \\ = \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2]}_{\text{variance}}. \end{aligned}$$

The Bias-Variance Decomposition (4)

- Thus we can write

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

- where

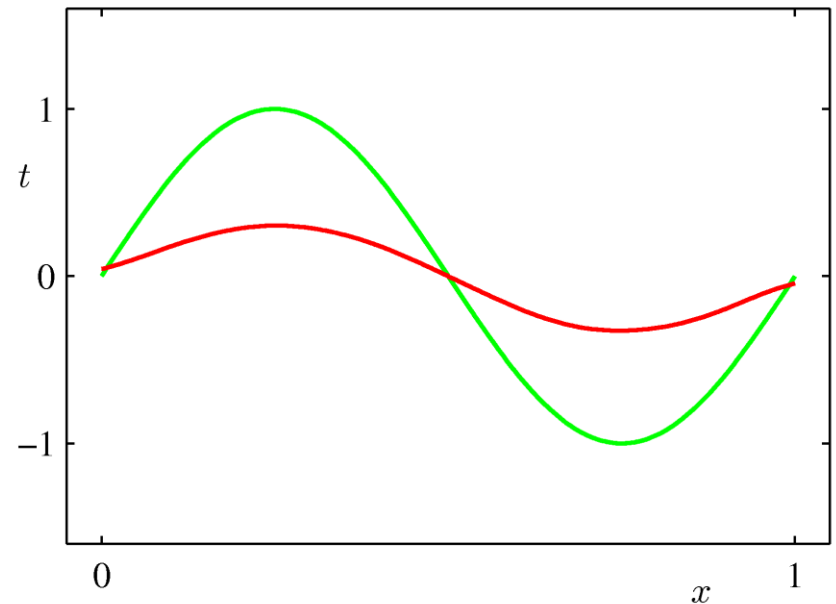
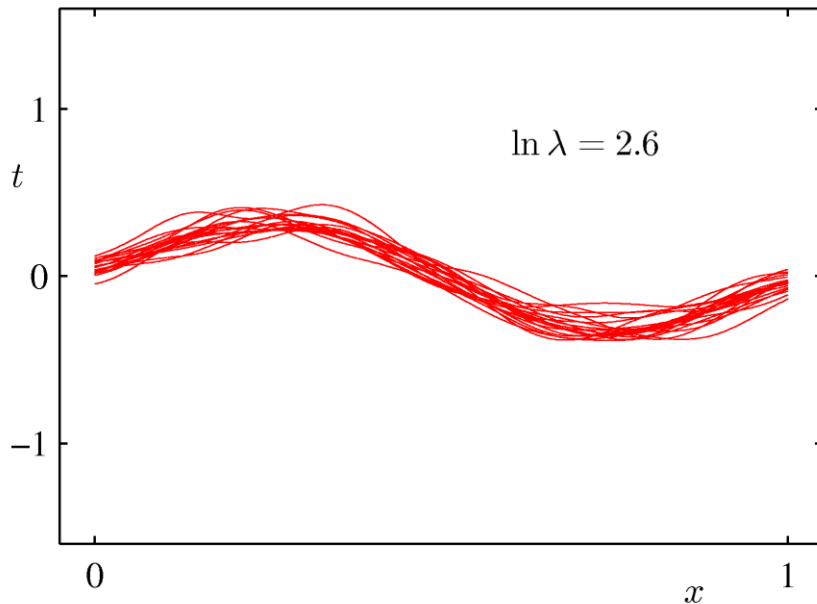
$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x}) d\mathbf{x}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}} [\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]\}^2] p(\mathbf{x}) d\mathbf{x}$$

$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t) d\mathbf{x} dt$$

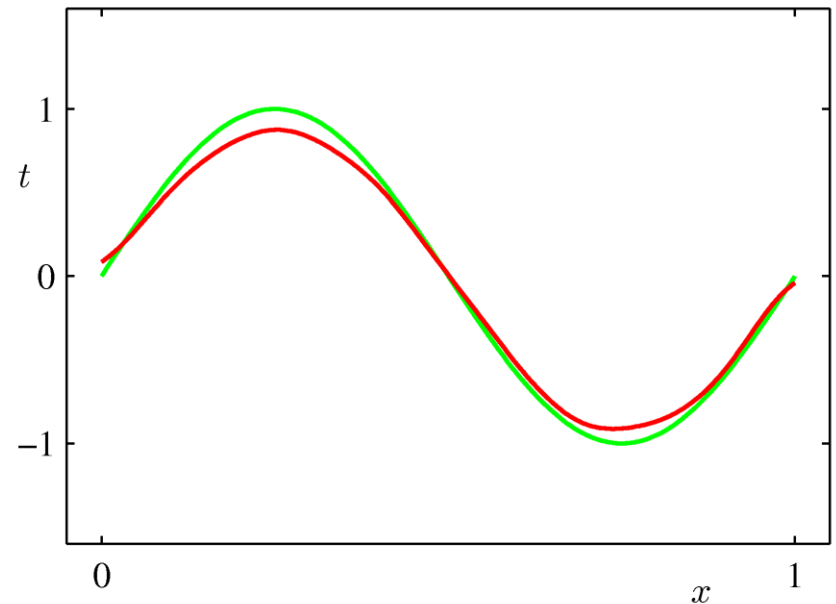
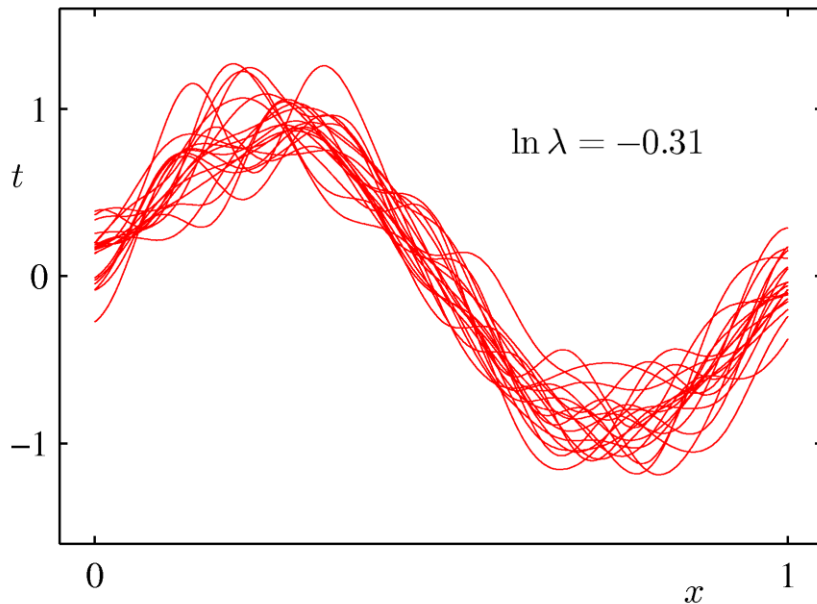
The Bias-Variance Decomposition (5)

- Example: 25 data sets from the sinusoidal, varying the degree of regularization.



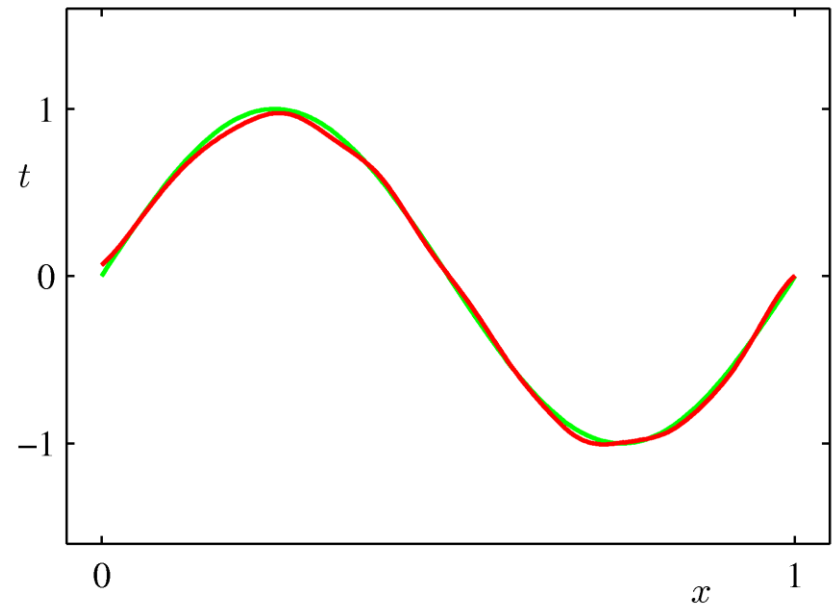
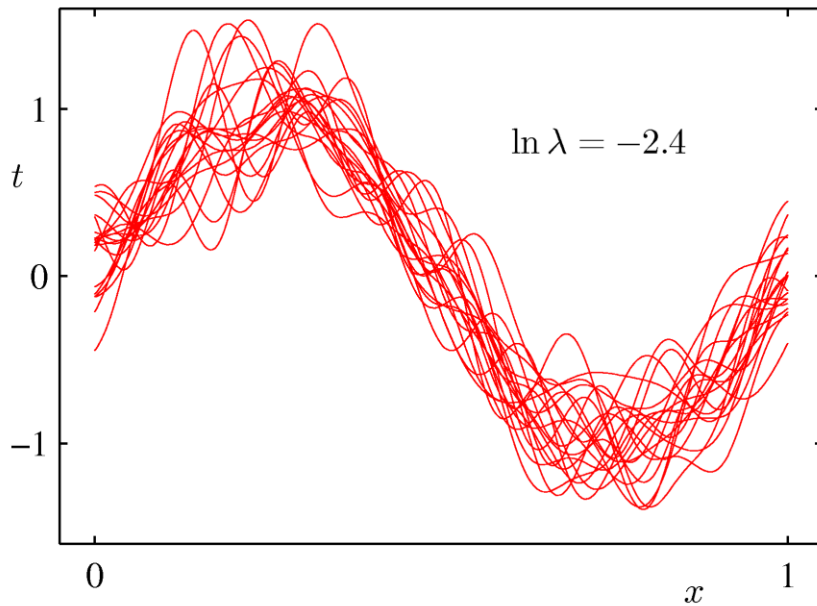
The Bias-Variance Decomposition (6)

- Example: 25 data sets from the sinusoidal, varying the degree of regularization



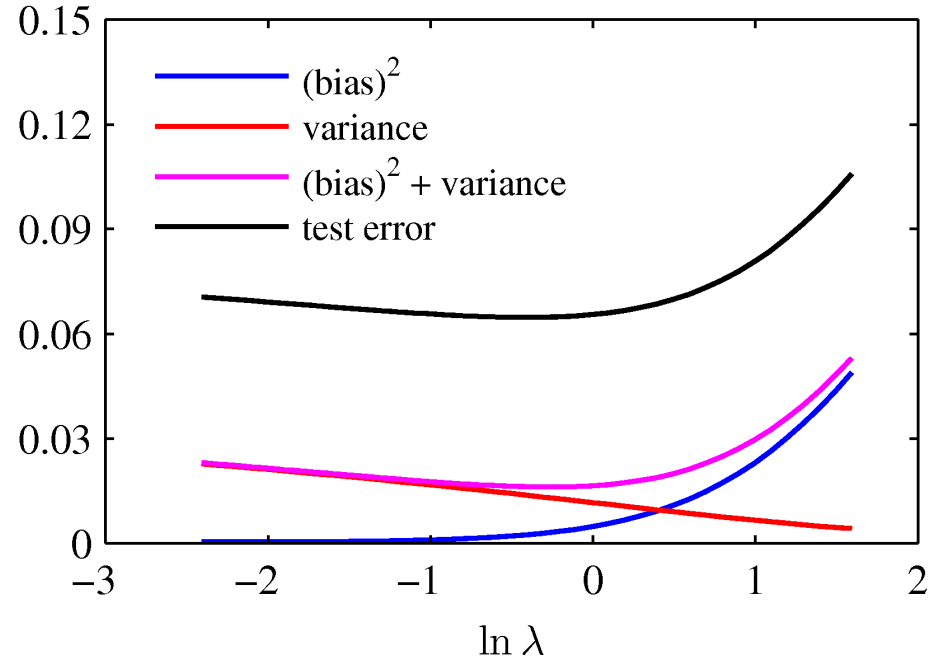
The Bias-Variance Decomposition (7)

- Example: 25 data sets from the sinusoidal, varying the degree of regularization.



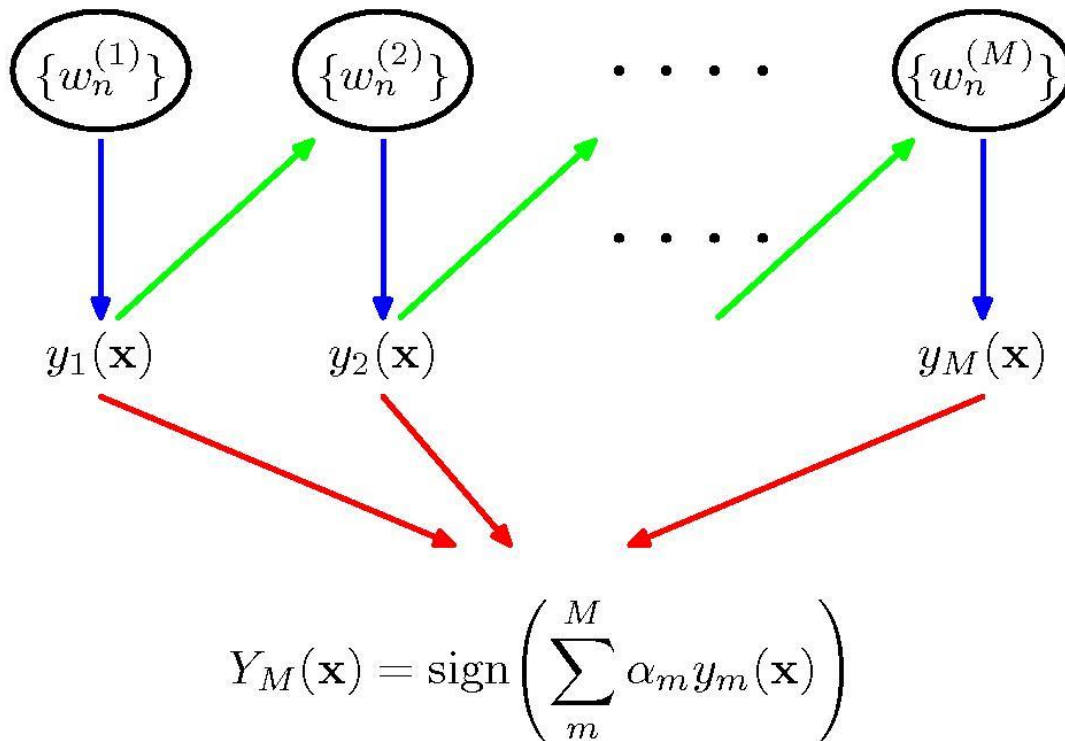
The Bias-Variance Trade-off

- From these plots, we note that an over-regularized model (large λ) will have a high bias, while an under-regularized model (small λ) will have a high variance.

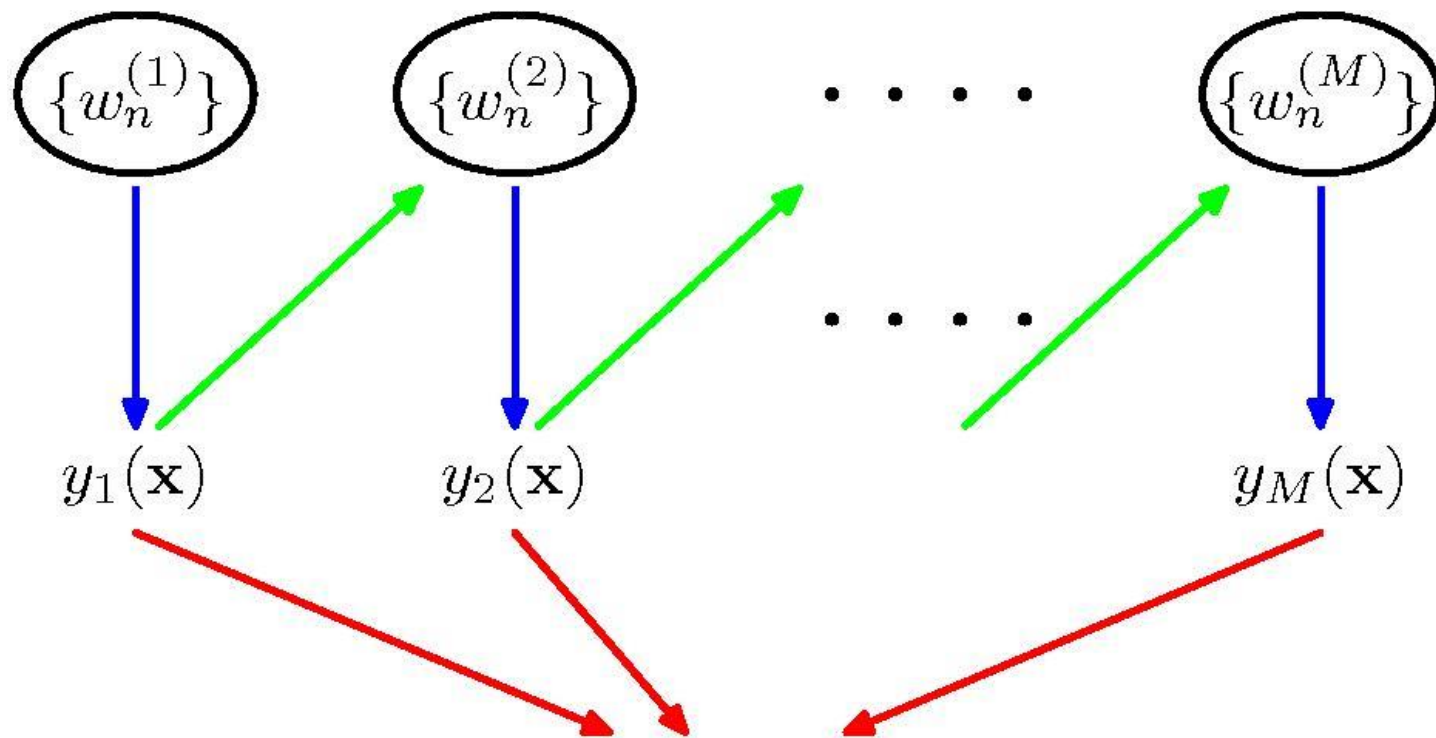


Committee of Machines

- Bagging = Bootstrapped Aggregation
- $E_{\text{COM}} = (E_{\text{AV}})/M$
- Boosting



Courtesy: C.M. Bishop, PRML



Courtesy: C.M. Bishop, PRML

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_m^M \alpha_m y_m(\mathbf{x}) \right)$$

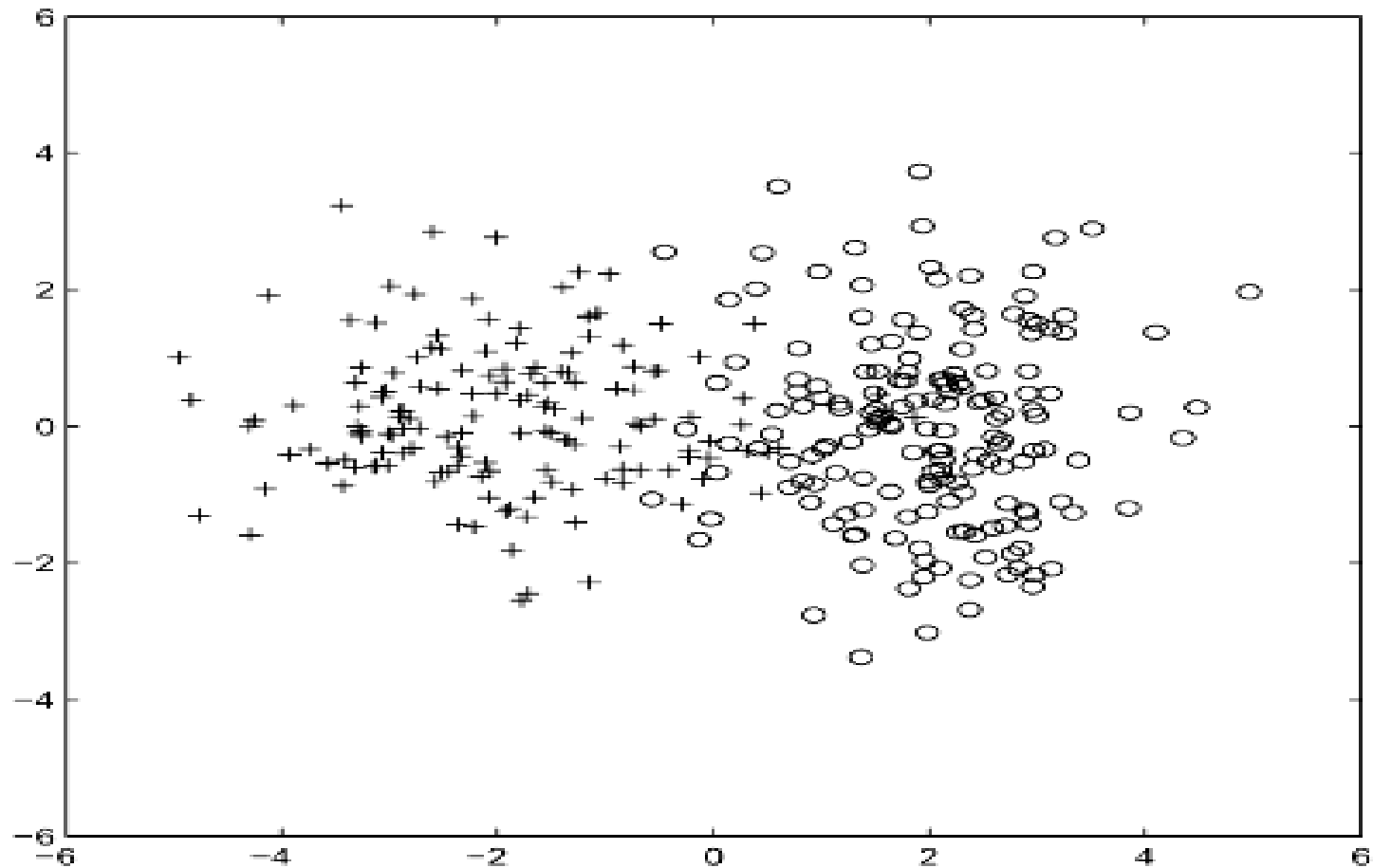
With so many options ... which kernel do we choose ?

But before we answer that question, we need to answer

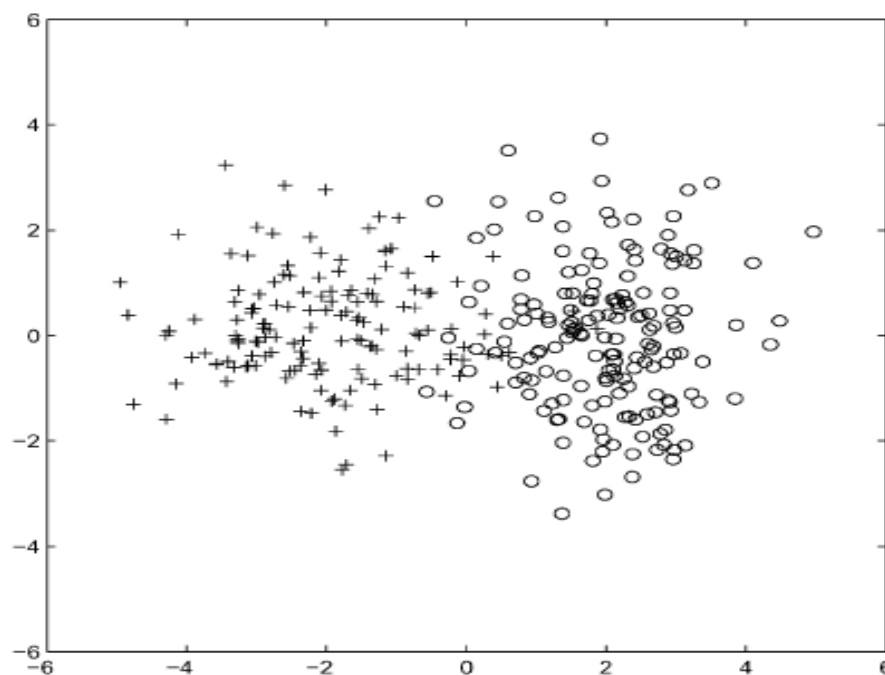
What is a good kernel ?

A kernel that makes it easier to classify the data in the feature space.

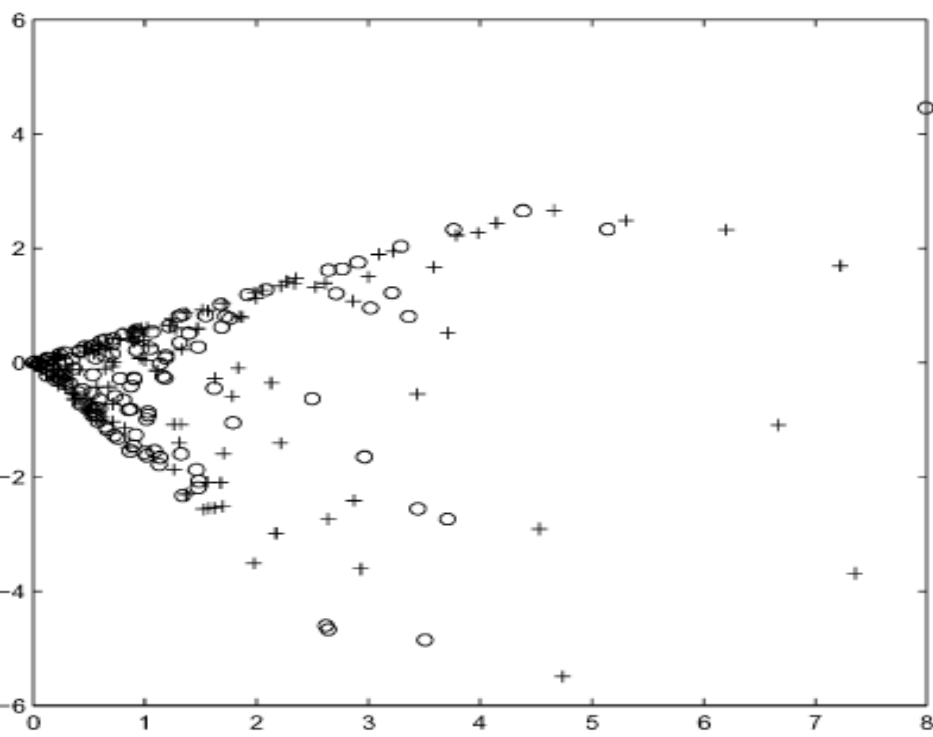
All kernels are not born equal



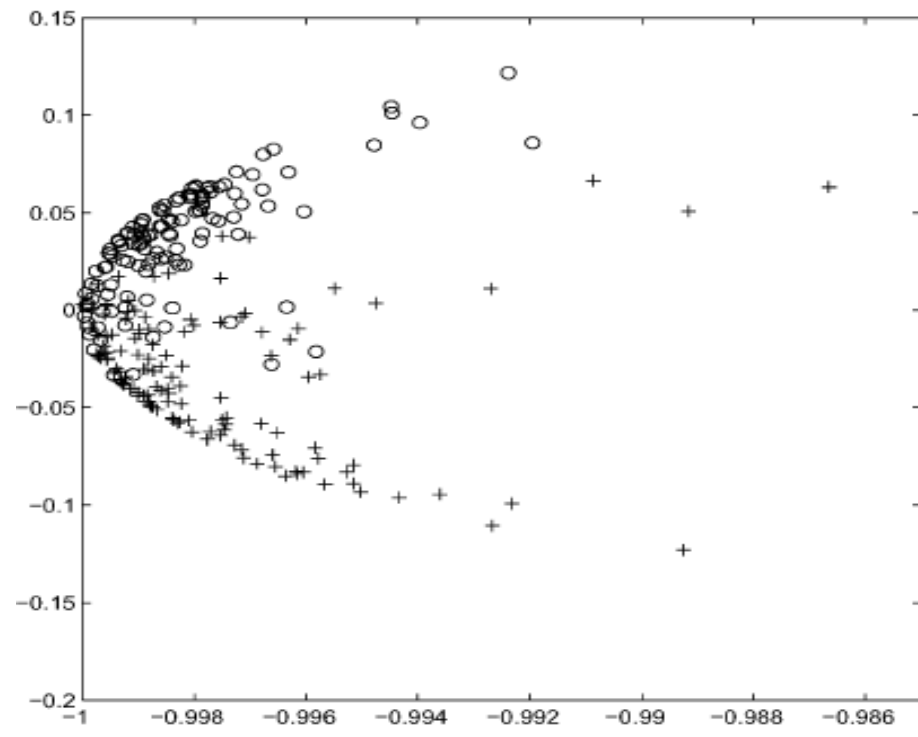
Input space



Polynomial kernel



Gaussian kernel



Q. How do we choose a kernel ?

A. Either choose by trial and error or optimize the kernel.

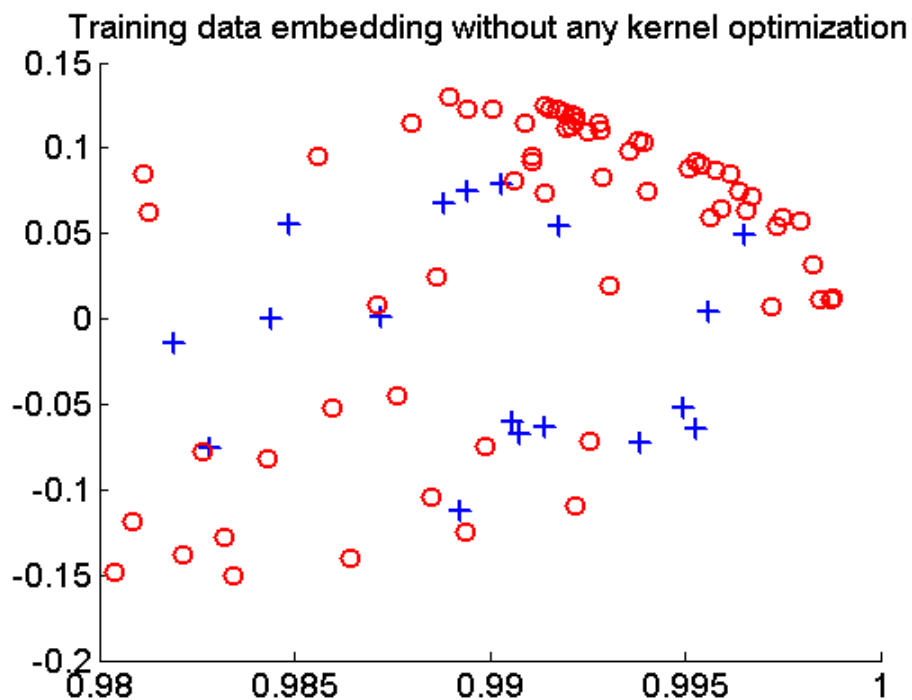
Caveat !

Classification may be more difficult in the feature space if kernel chosen is not suited for the task at hand.

Finding the Right Kernel

- Subject of considerable research
- Needs to be data dependent
- Obviously, one needs to consider a part of the data set
- Another optimization task !

Consider a sample data set

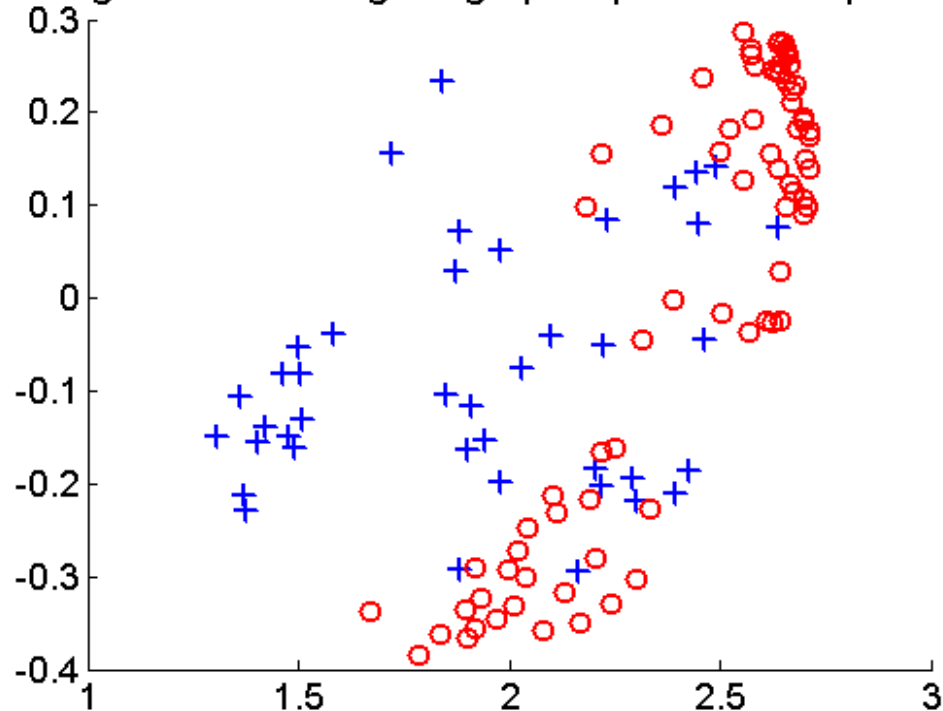


Obviously, not linearly separable.

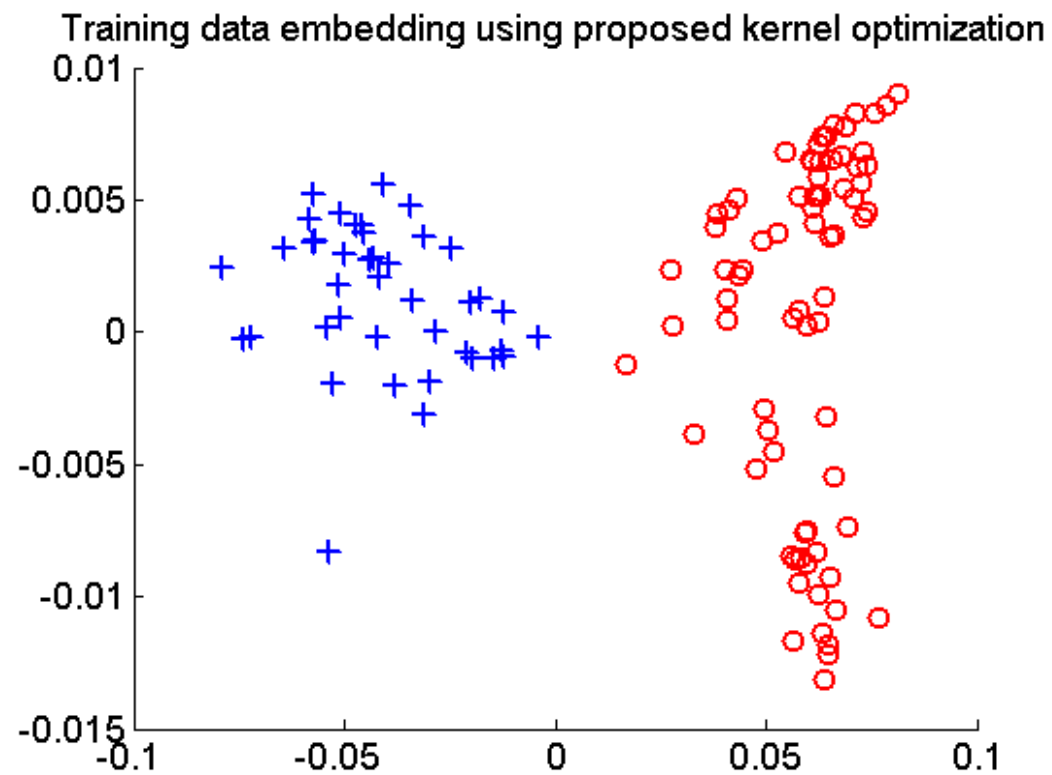
Need to use a mapping to a higher dimensional
feature space

Using a simple kernel optimization procedure (Xiong et al)

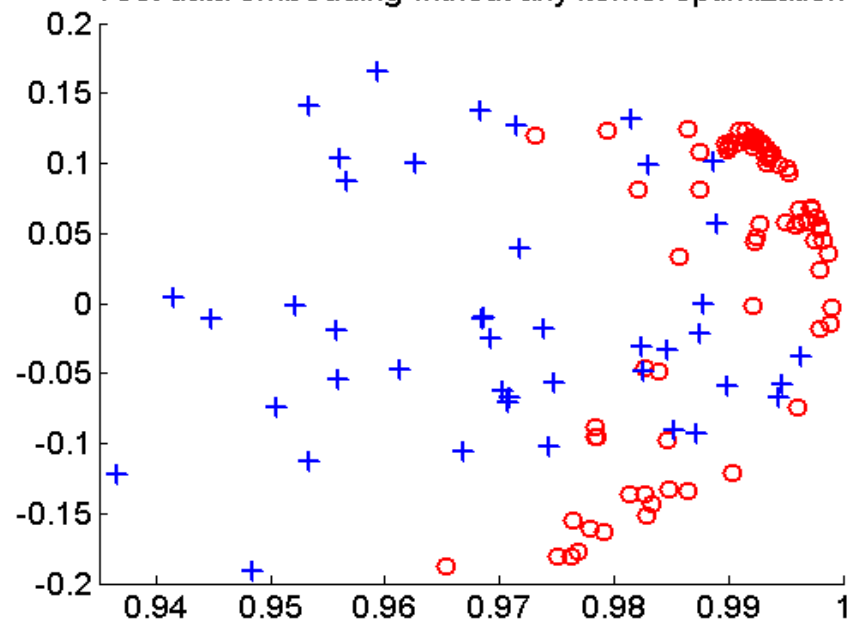
Training data embedding using alpha update kernel optimization



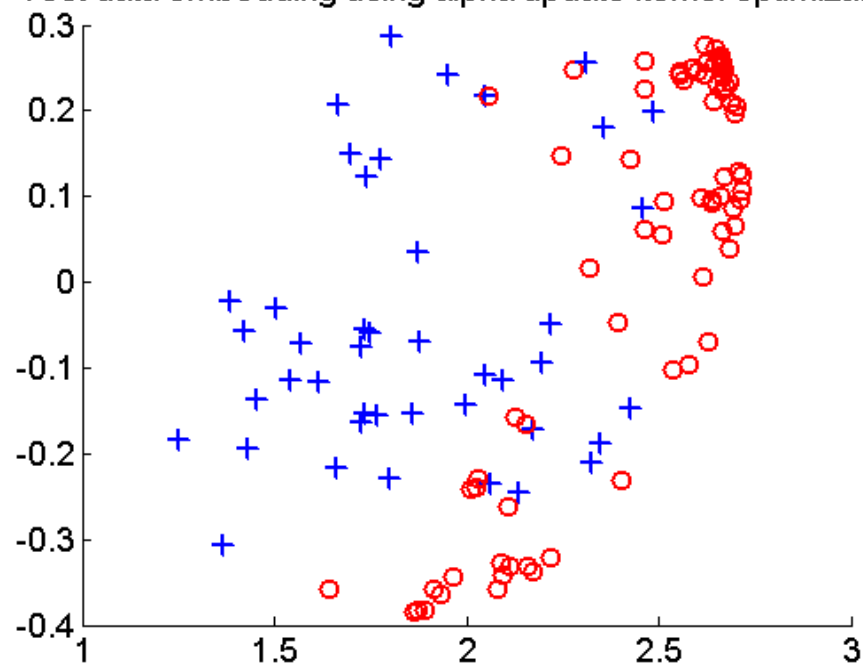
Using a more efficient kernel optimization procedure



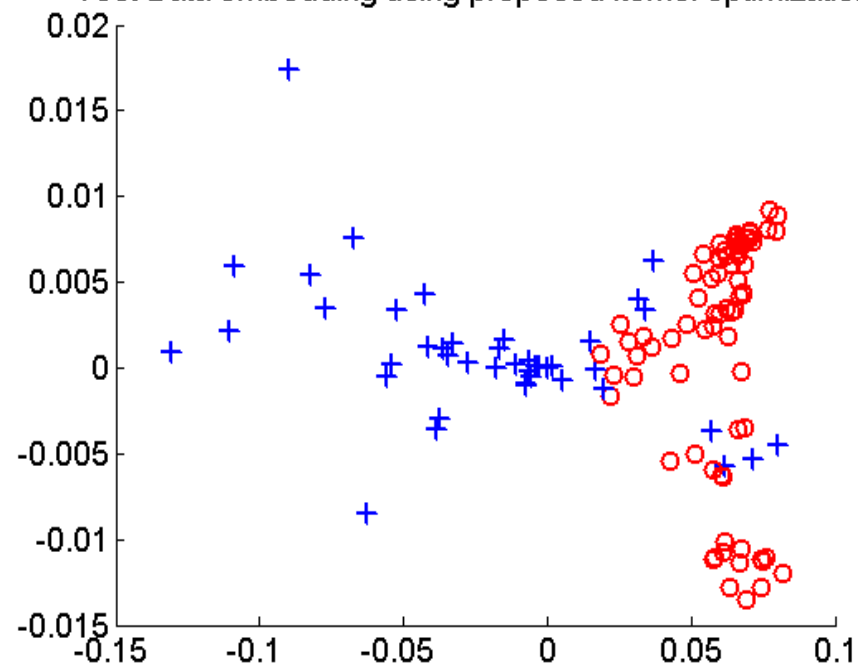
Test data embedding without any kernel optimization



Test data embedding using alpha update kernel optimization



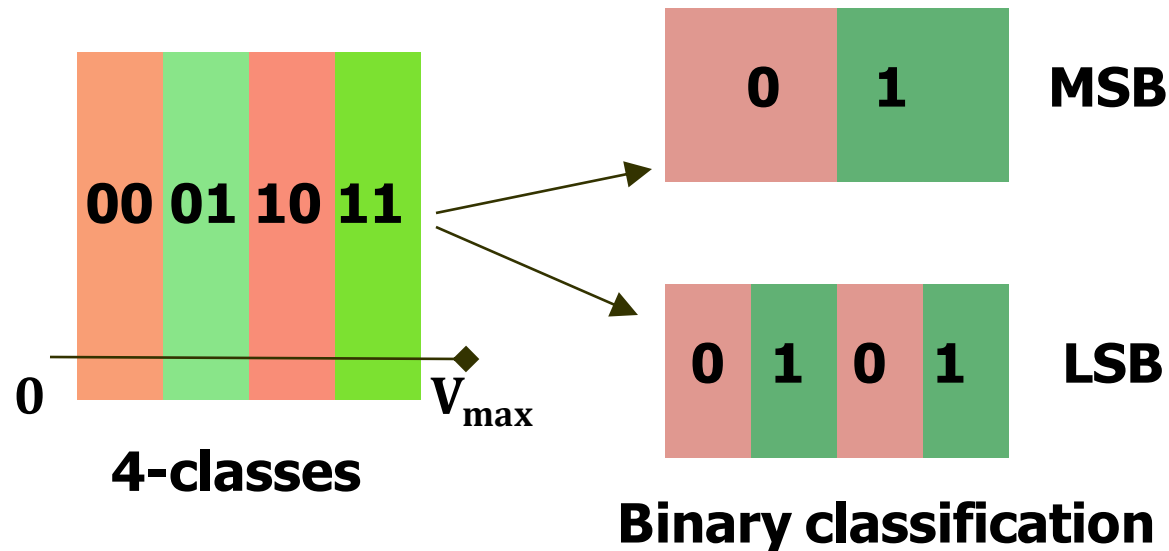
Test Data embedding using proposed kernel optimization



Many open questions

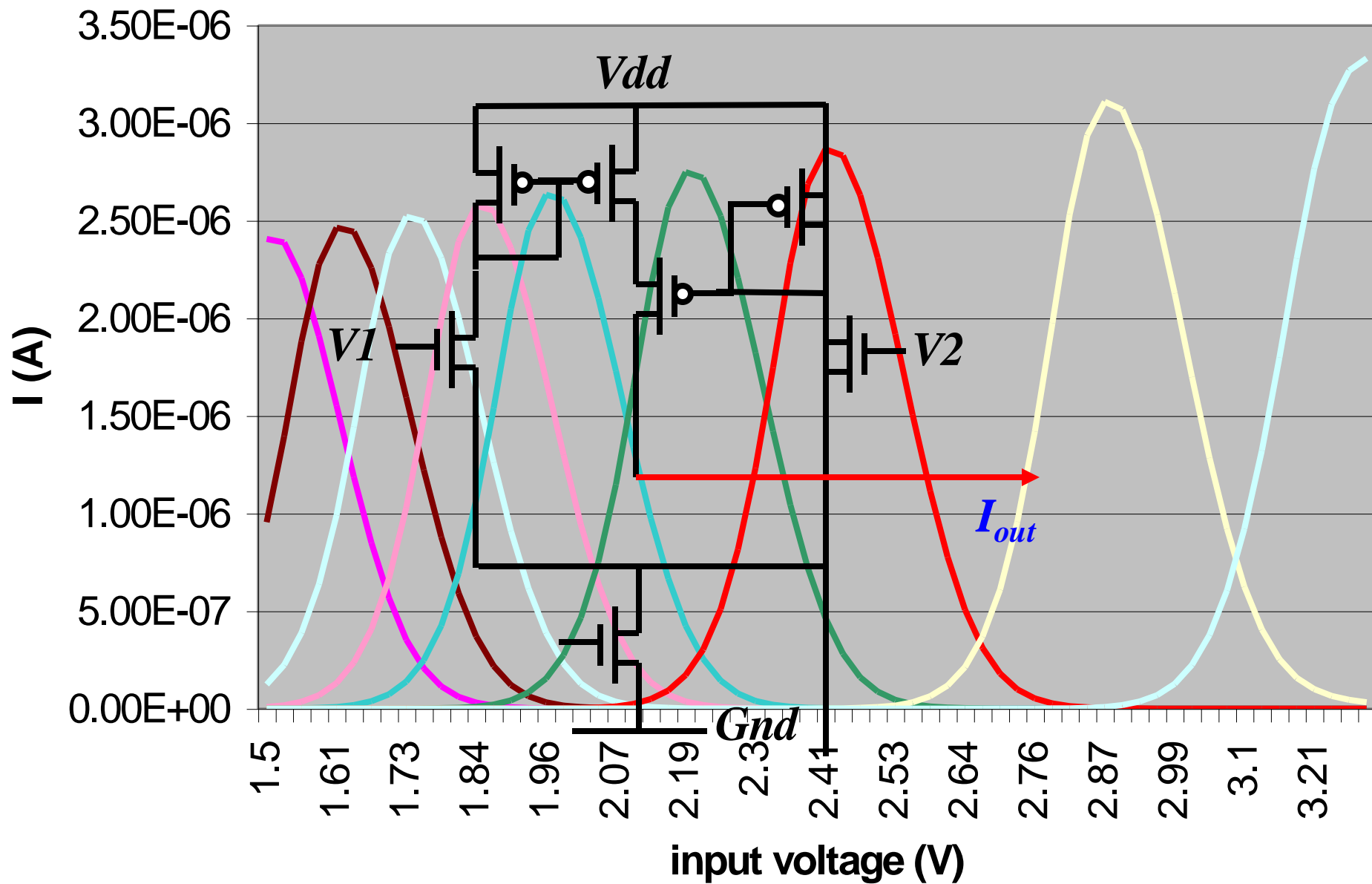
- What is the effect on outliers?
- How to extend it to other domains like text, speech, images, video etc ?
- Can we implement these in hardware ?

A/D Conversion Through SVMs

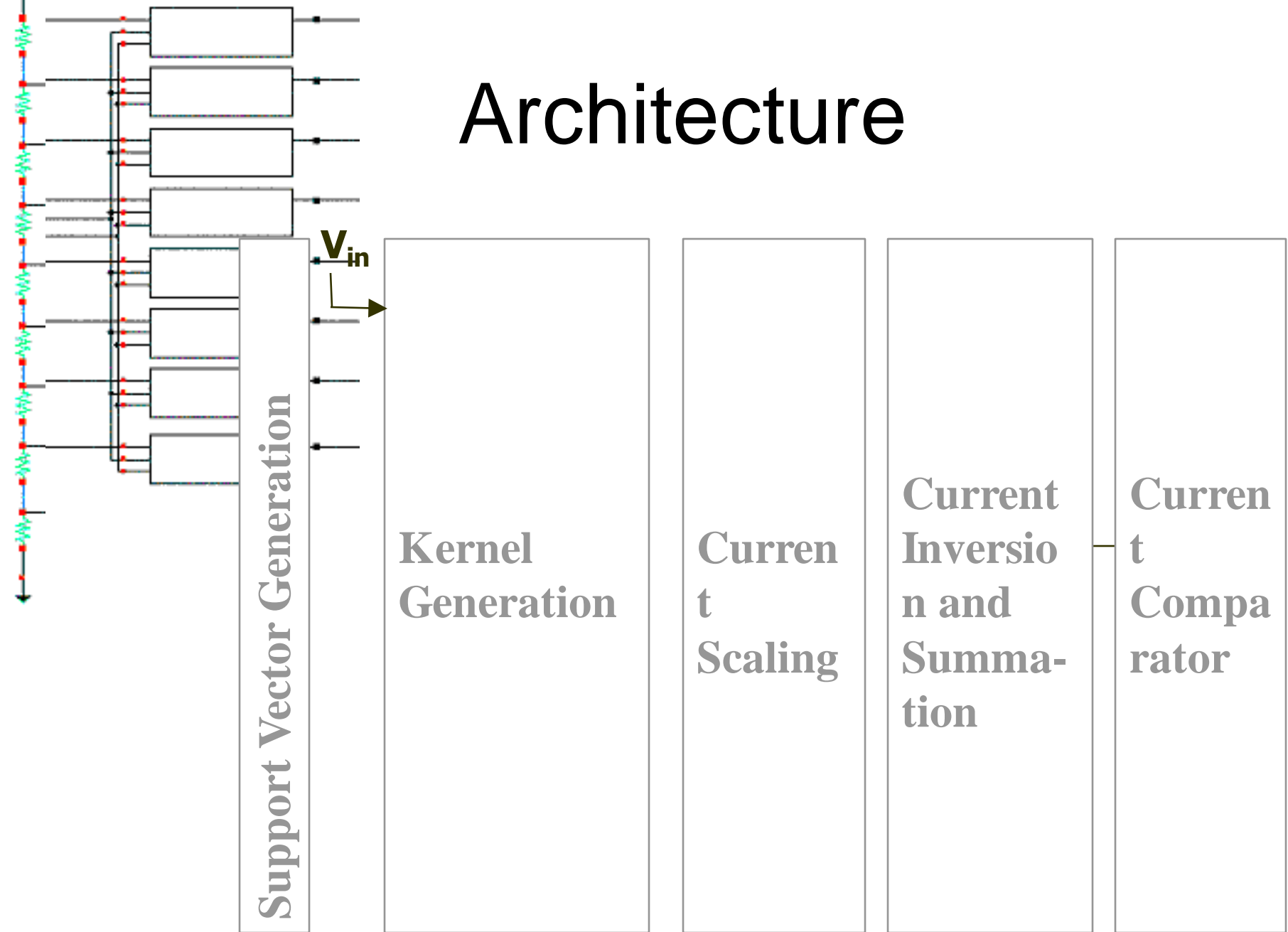


- Conventional SVR requires kernels to be symmetric and positive definite

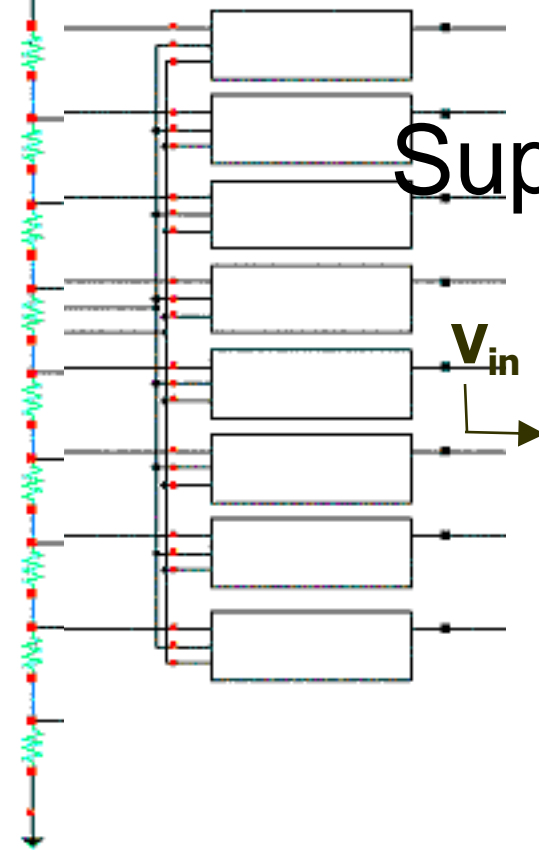
Kernel block currents



Architecture



Support Vector Generation



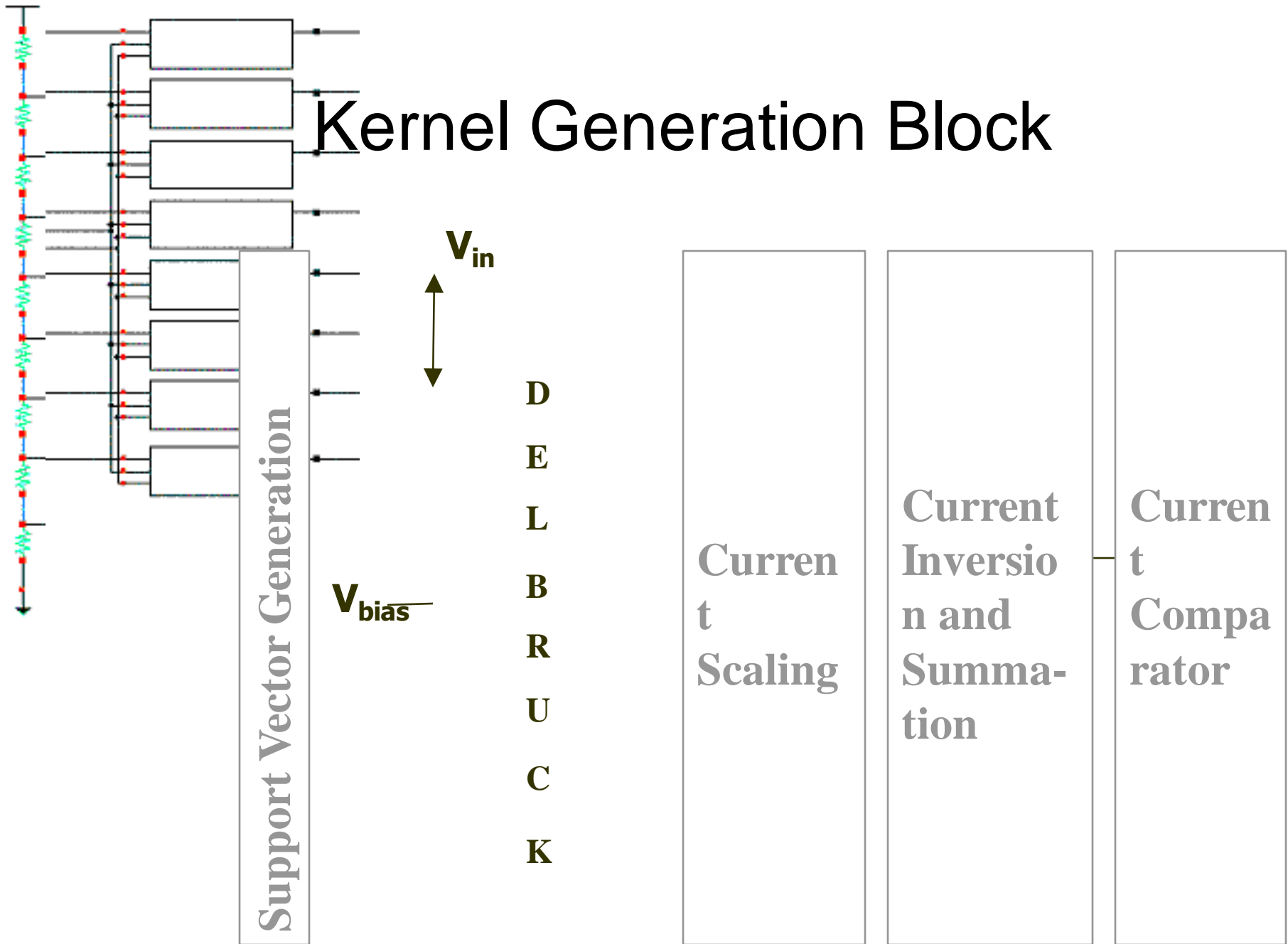
**Kernel
Generation**

**Current
Scaling**

**Current
Inversion and
Summa-
tion**

**Current
Comparator**

Kernel Generation Block



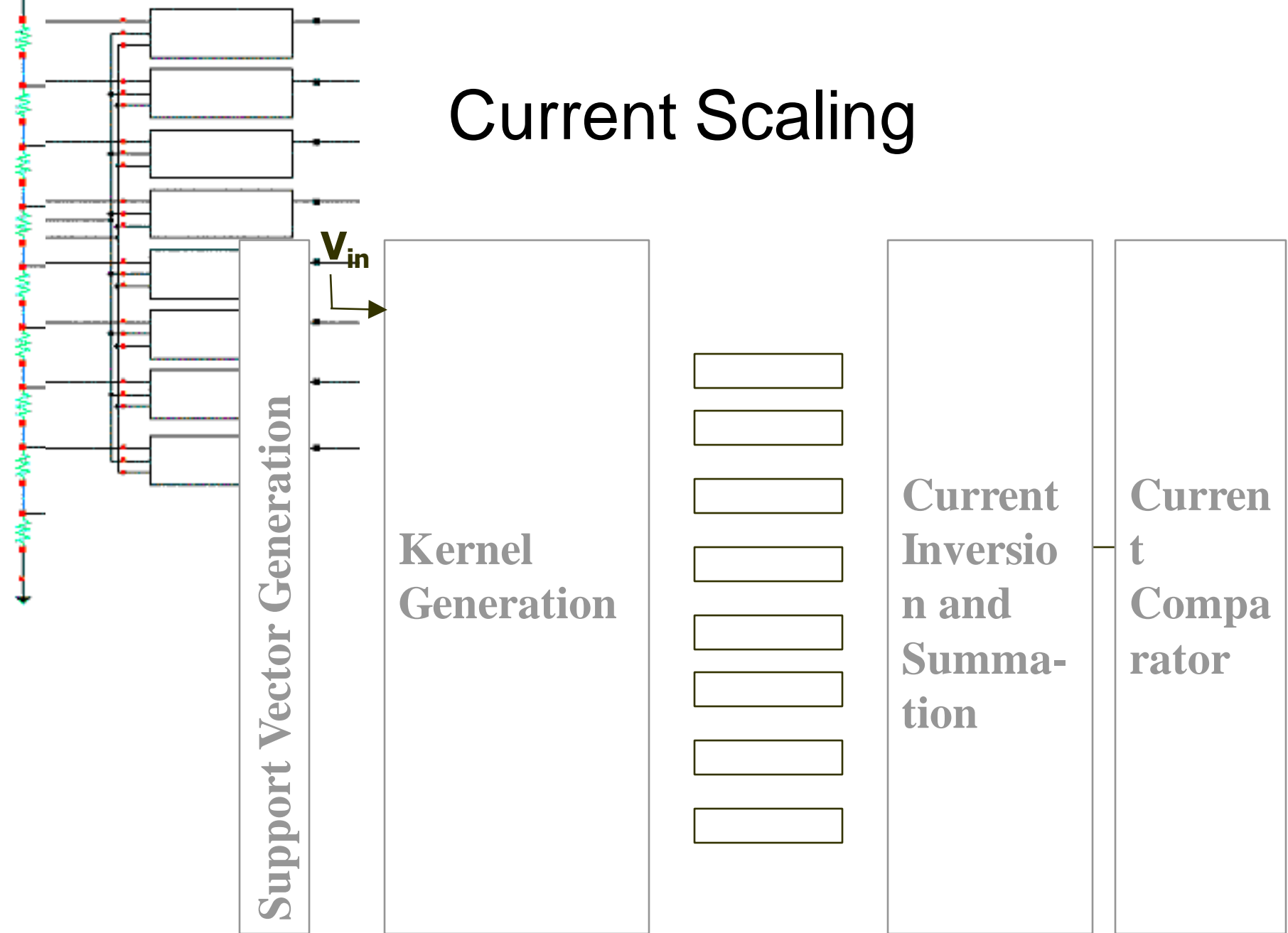
Estimated value of the function at a new point x is given by the weighted sum of kernel block outputs

Co-efficients determined from simulation

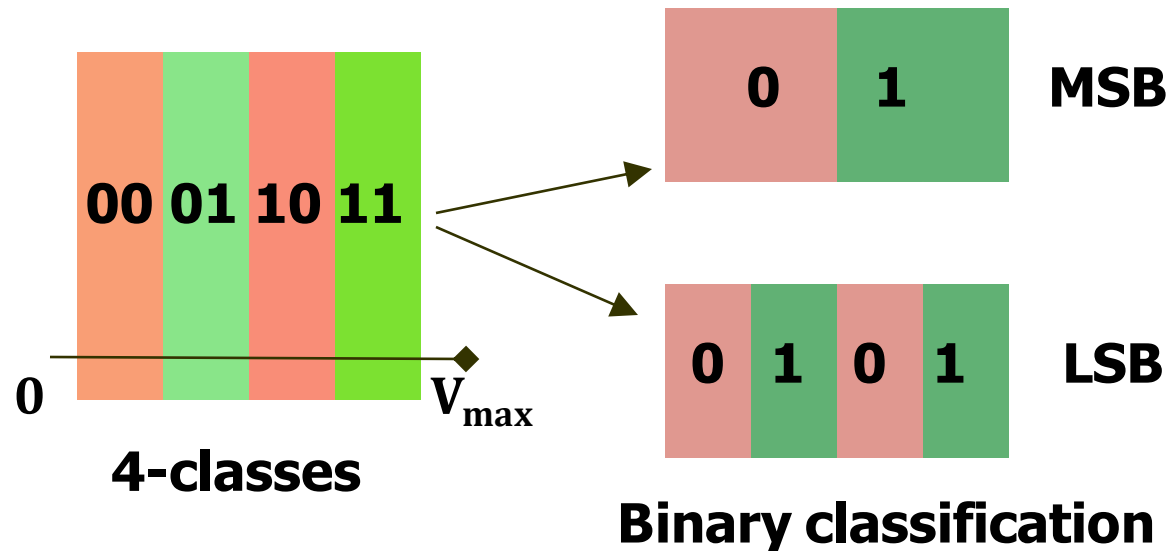
We choose digital weights

Tied to real data, can be re-programmed later

Current Scaling

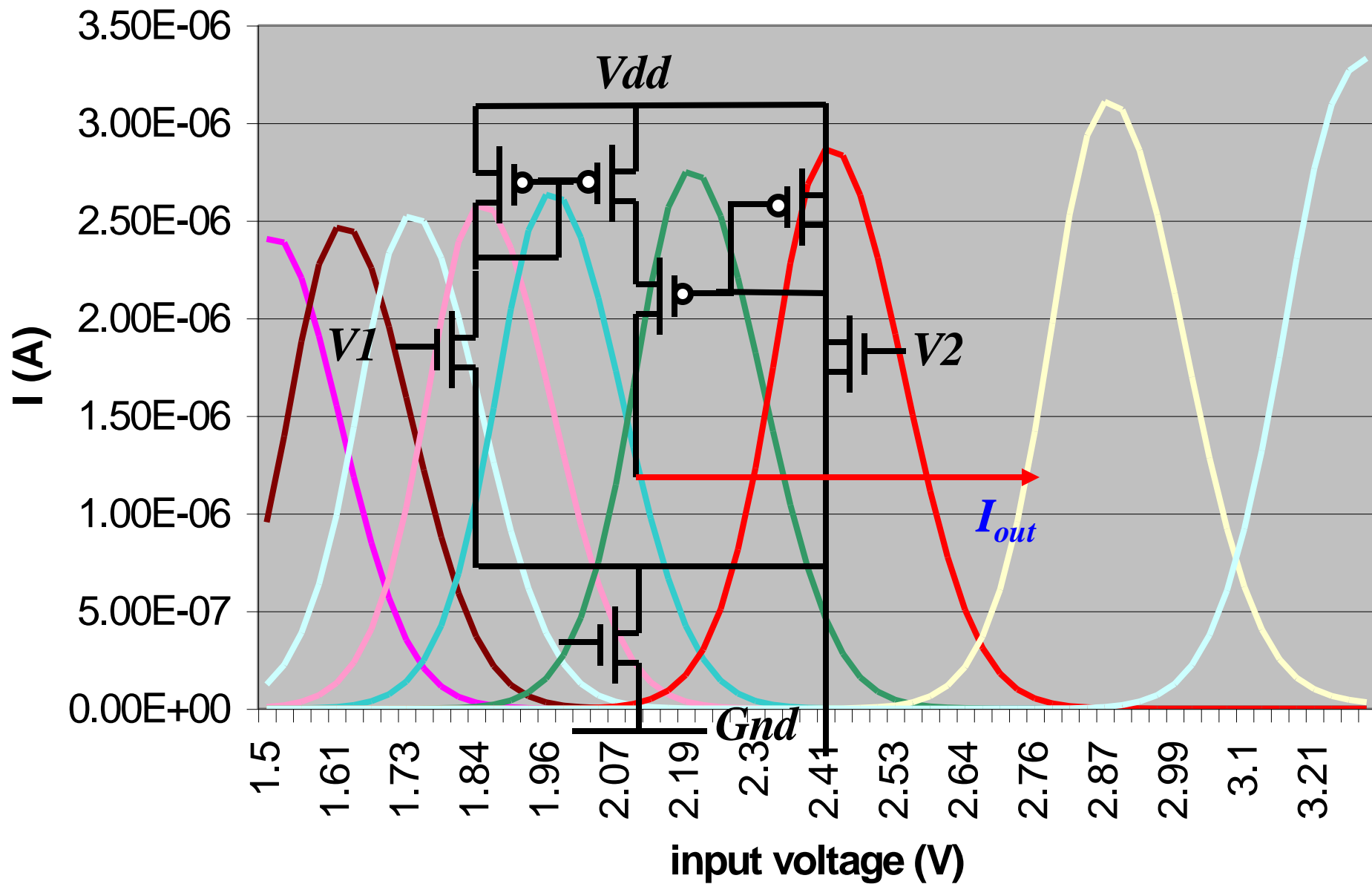


A/D Conversion Through SVMs

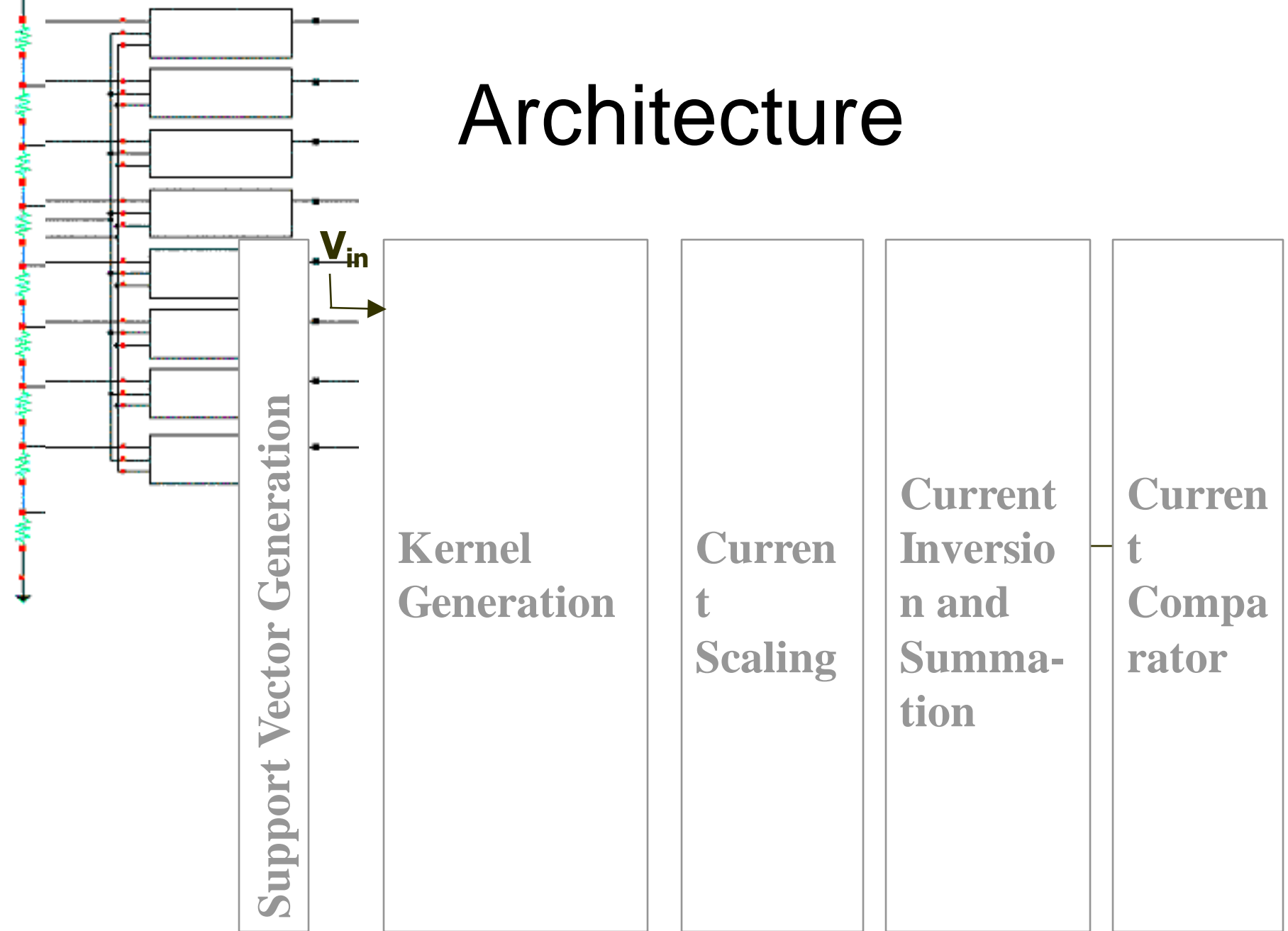


- Conventional SVR requires kernels to be symmetric and positive definite

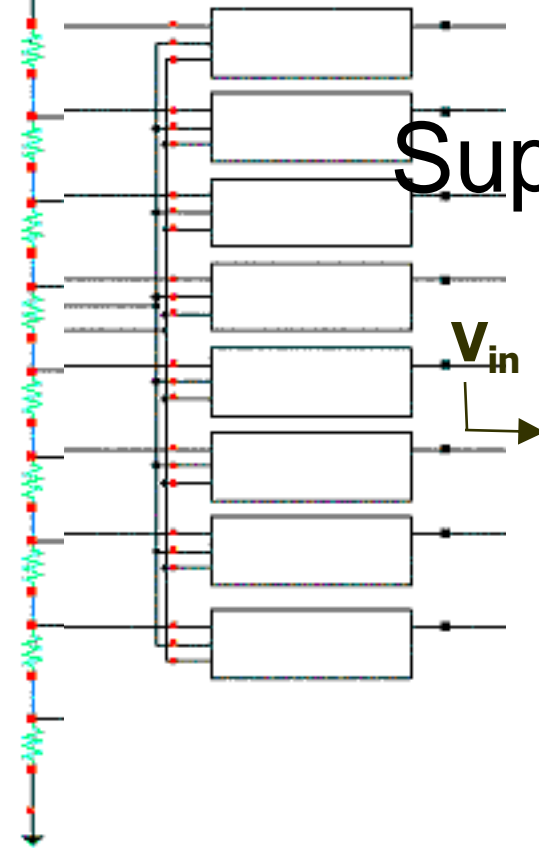
Kernel block currents



Architecture



Support Vector Generation



**Kernel
Generation**

**Current
Scaling**

**Current
Inversion and
Summa-
tion**

**Current
Comparator**

Diagram illustrating a neural network architecture for support vector generation. The network consists of a vertical stack of 8 **Kernel C** blocks. A **Support Vector Generation** block is connected to the inputs of the kernel blocks. A bias voltage V_{bias} is applied to the bottom of the kernel stack, and an input voltage V_{in} is applied to the top. A legend on the right lists: D, E, L, B, R, U, C, K.

DELBROUCK

Current Scaling

Current Inversion and Summation

Current Comparator

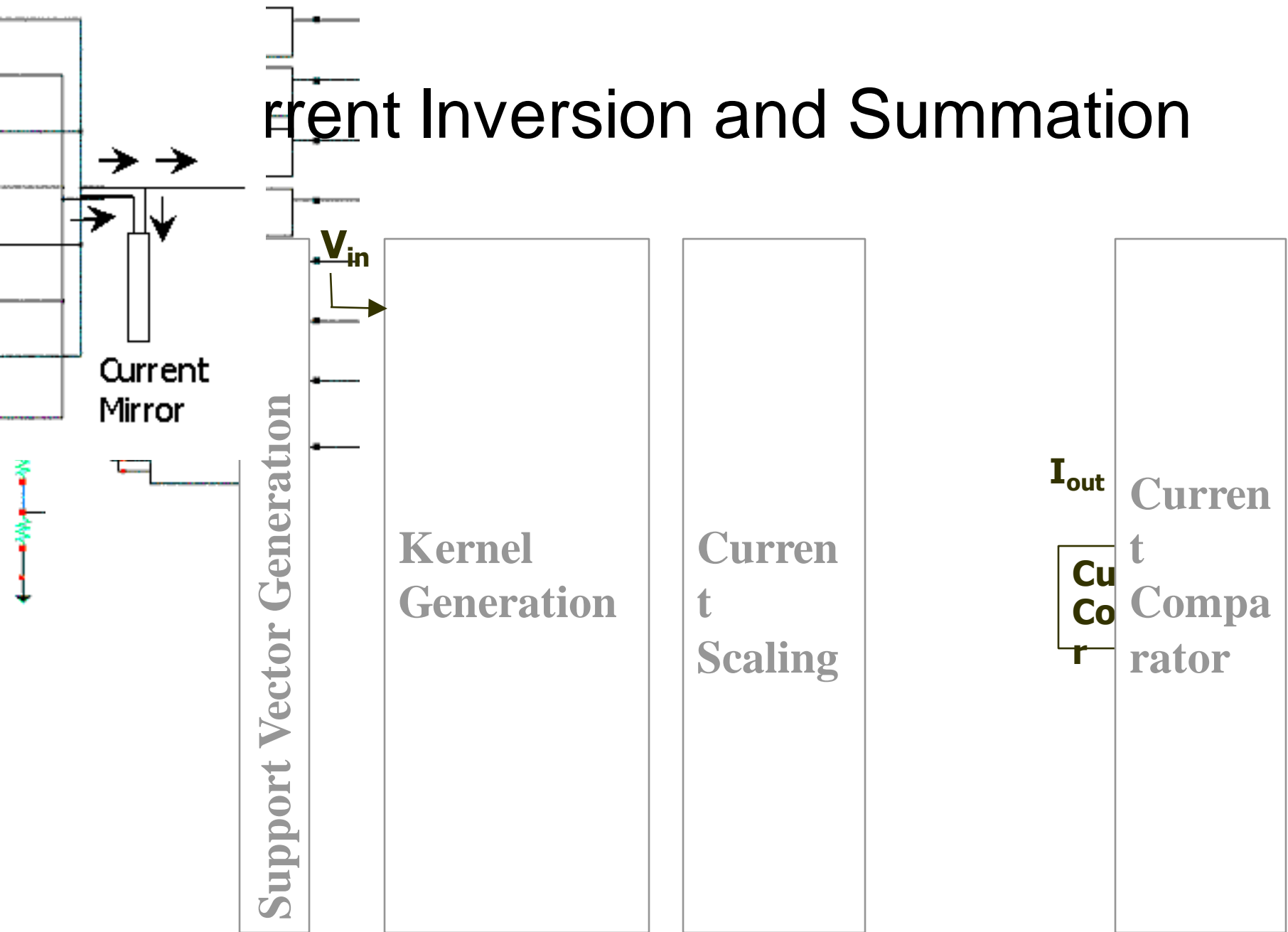
Estimated value of the function at a new point x is given by the weighted sum of kernel block outputs

Co-efficients determined from simulation

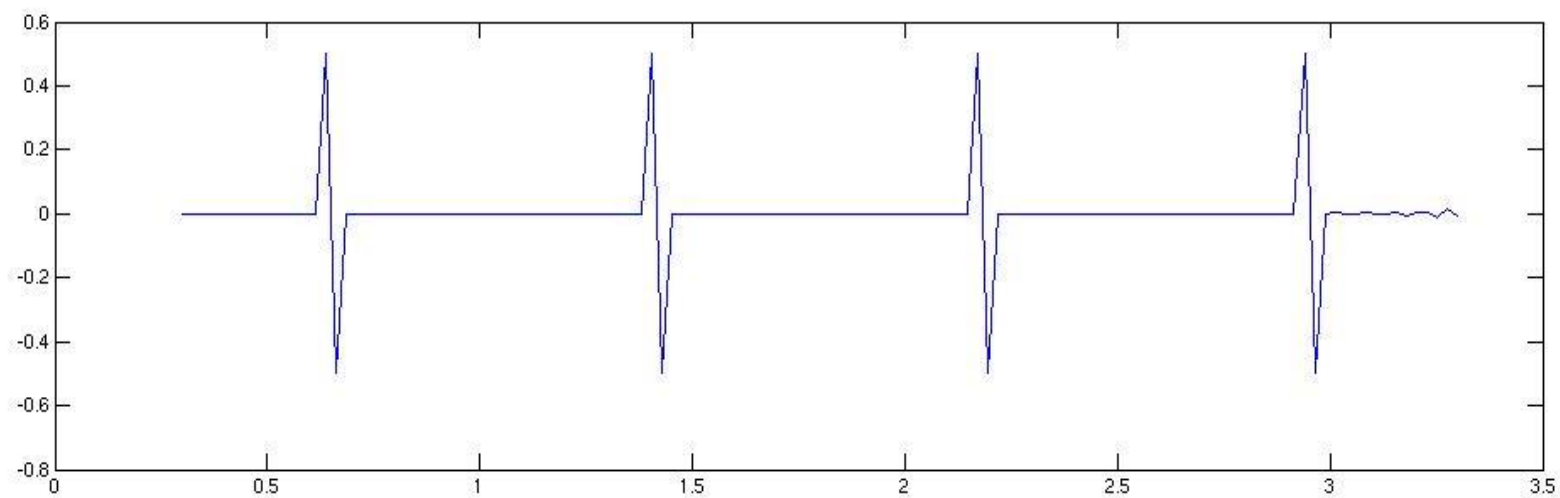
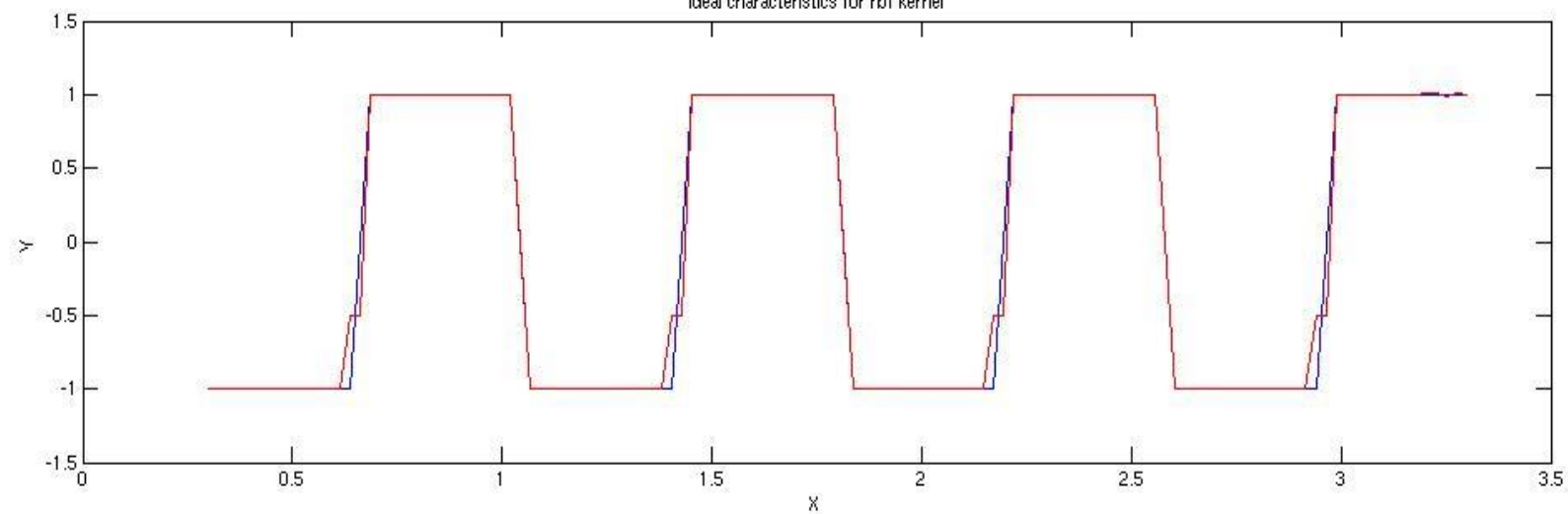
We choose digital weights

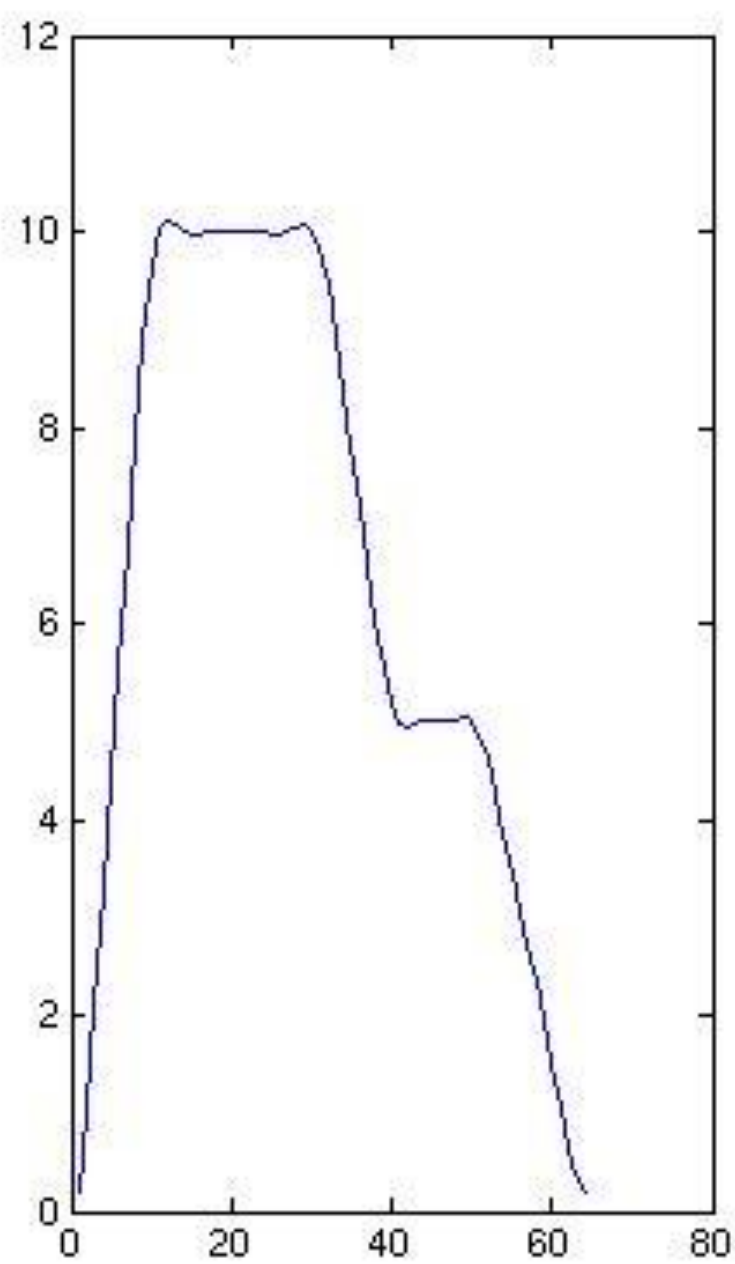
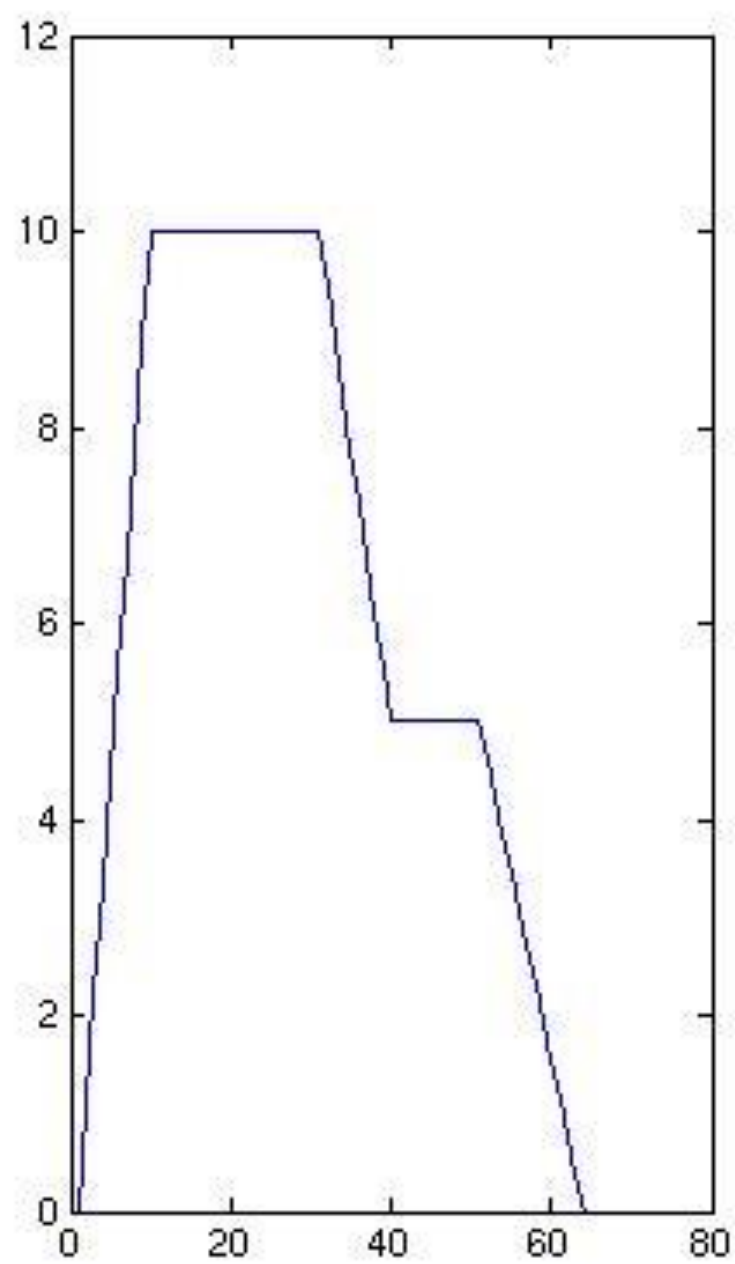
Tied to real data, can be re-programmed later

Current Inversion and Summation

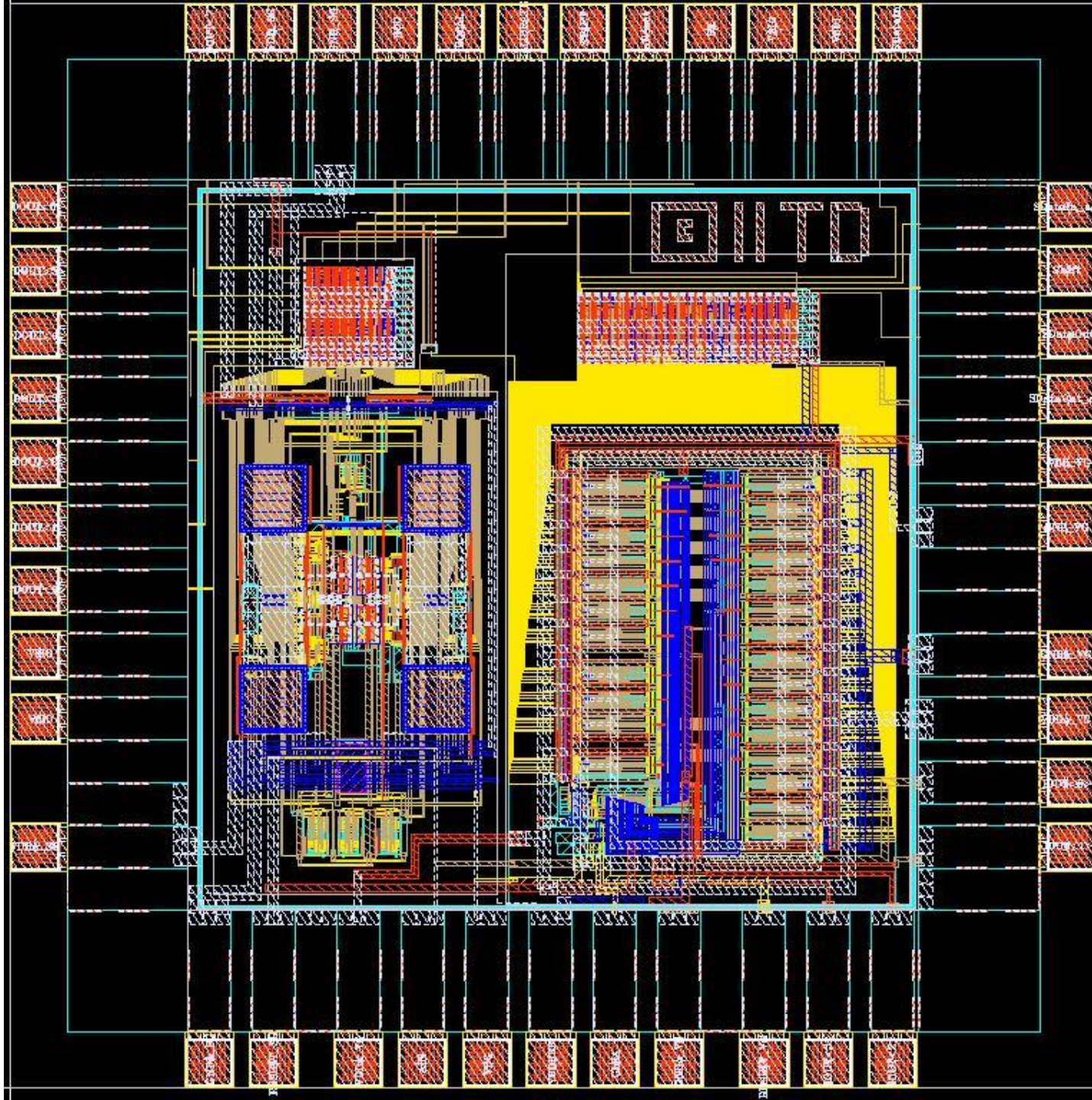


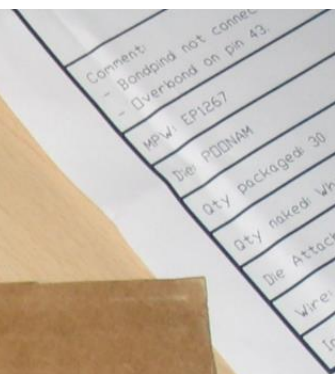
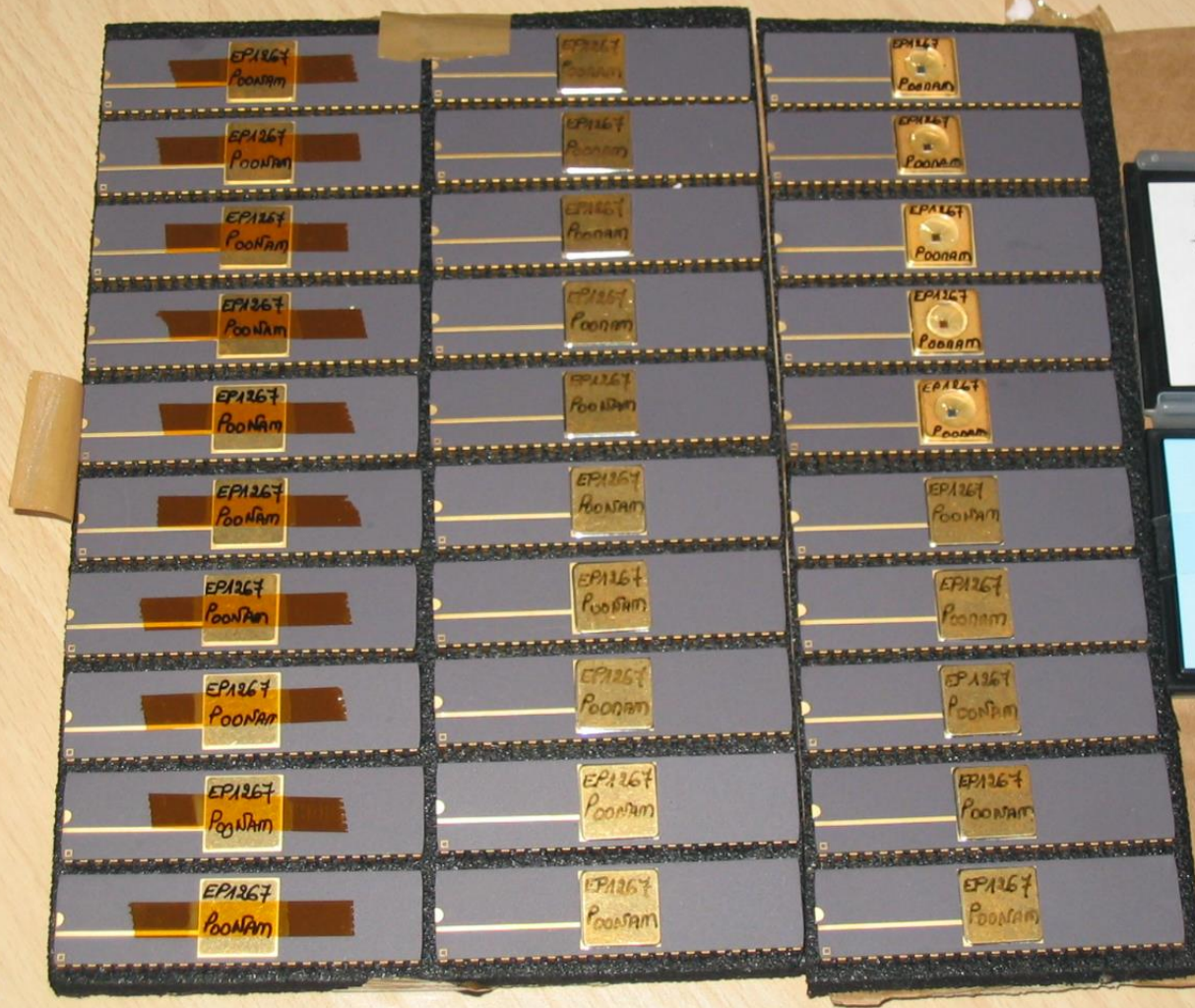
Ideal characteristics for rbf kernel





- Technique has the ability to digitally correct for changes or differences in kernel characteristics *after* fabrication
- Chip fabricated in 0.18 micron CMOS through *Europpractice*

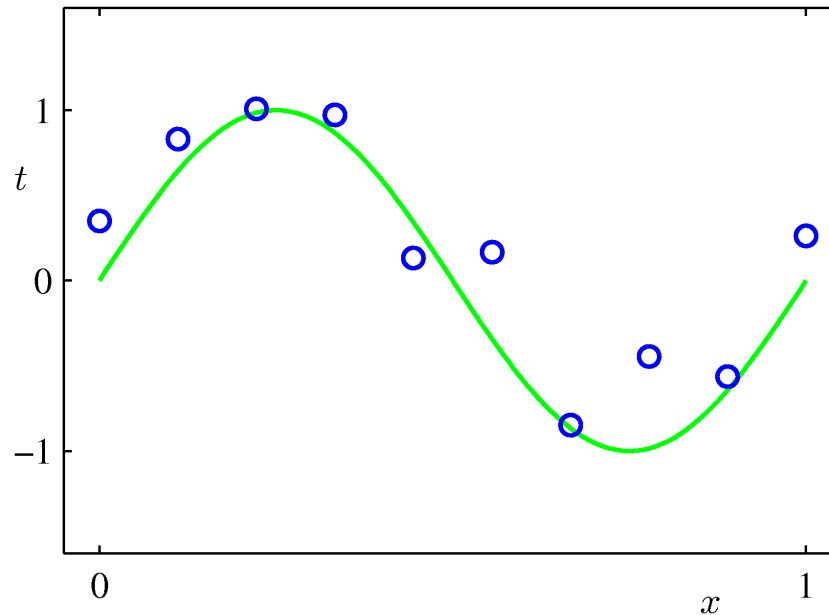




Comment:
- Bonding not connect
- Overbond on pin 43
MPW: EP1267
Die: Poonam
Qty packaged: 30
Qty boxed: 30
Die Attach:
Wire:

Linear Basis Function Models (1)

- Example: Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$