

Lecture 3 : More Regressions

Scott Broderick

Department of Materials Design and Innovation
University at Buffalo

September 21, 2026

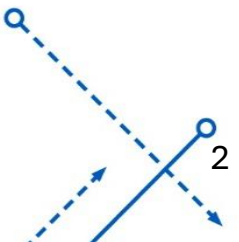
scottbro@buffalo.edu
mdi.buffalo.edu

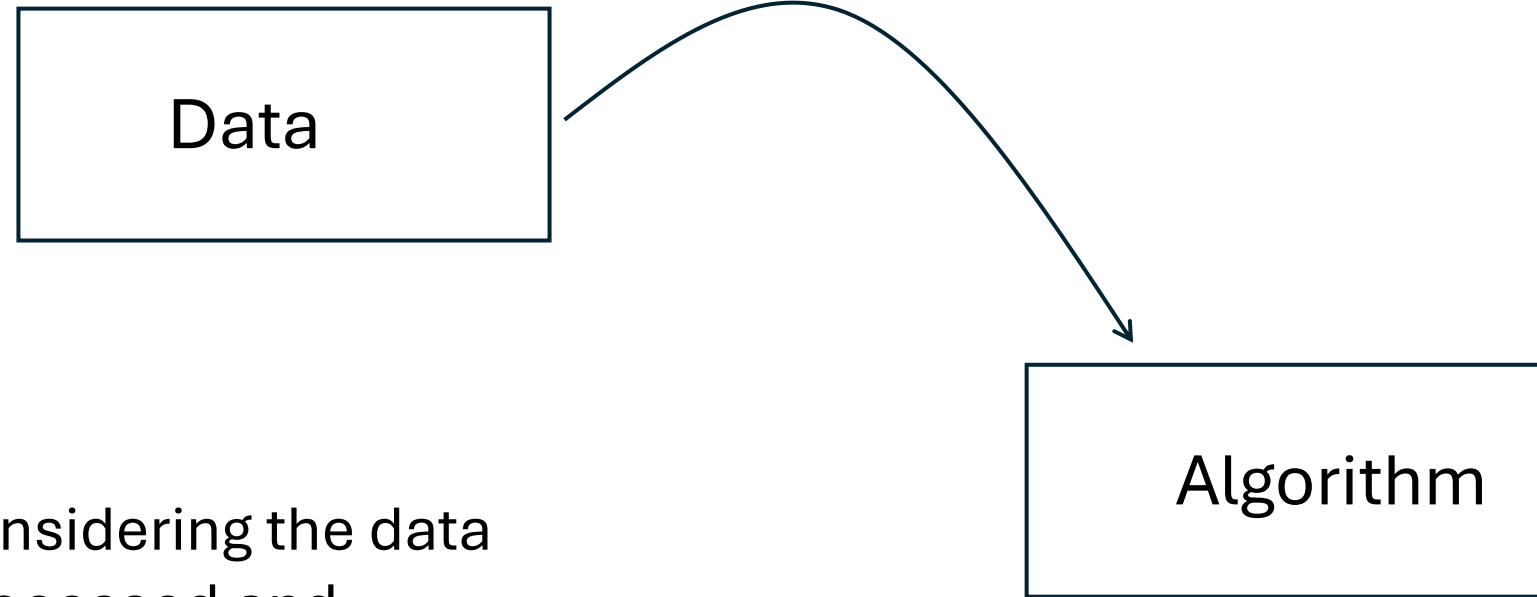


Last Week :

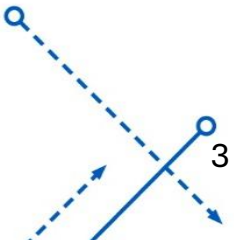
- Discussed unsupervised learning approaches (e.g. PCA, IsoMap) – ways to map data and find trends and better understand underlying physics
- Discussed some regression approaches (e.g. MLR, PCR, random forest, boosting)

Today we will continue the discussion on regressions, but with approaches which have different form and better account for uncertainty. And on Wednesday we will then discuss neural networks and machine learning approaches.





We have been considering the data input as being processed and completed. Perhaps some modification or convolution of the data is necessary. But that is separate from the analysis aspect (ie. two different isolated parts)



Onto A Couple Examples

➤ The following are the design requirements, in order of our defined preference:

1) Room temperature ductility

- Include both % elongation at room temperature from legacy data and B/G from DFT calculations
- Use the B/G values (both DFT and informatics) as inputs into the modeling for % elongation, to develop a more unified analysis

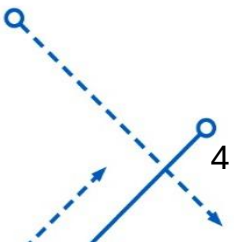
2) Strength

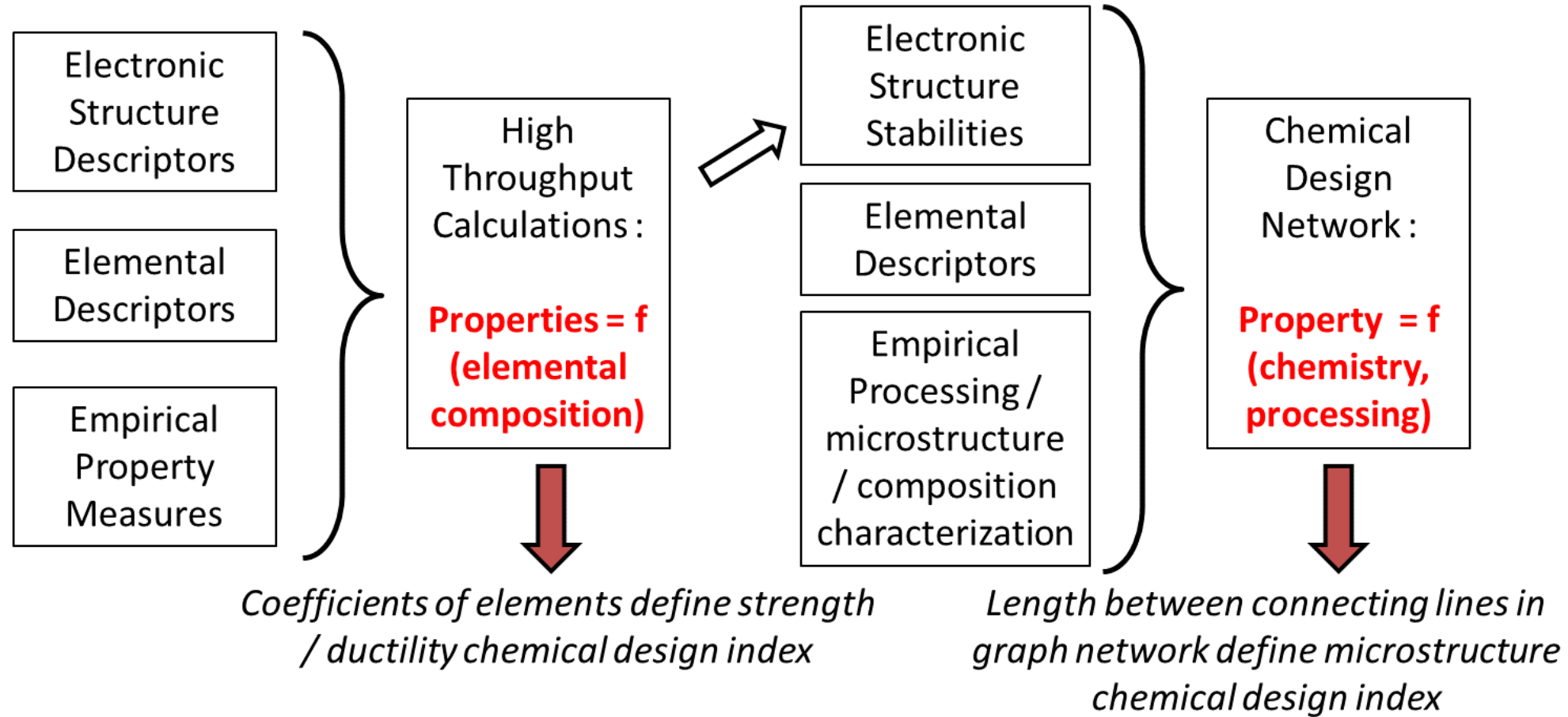
3) Maintain duplex microstructure with processing

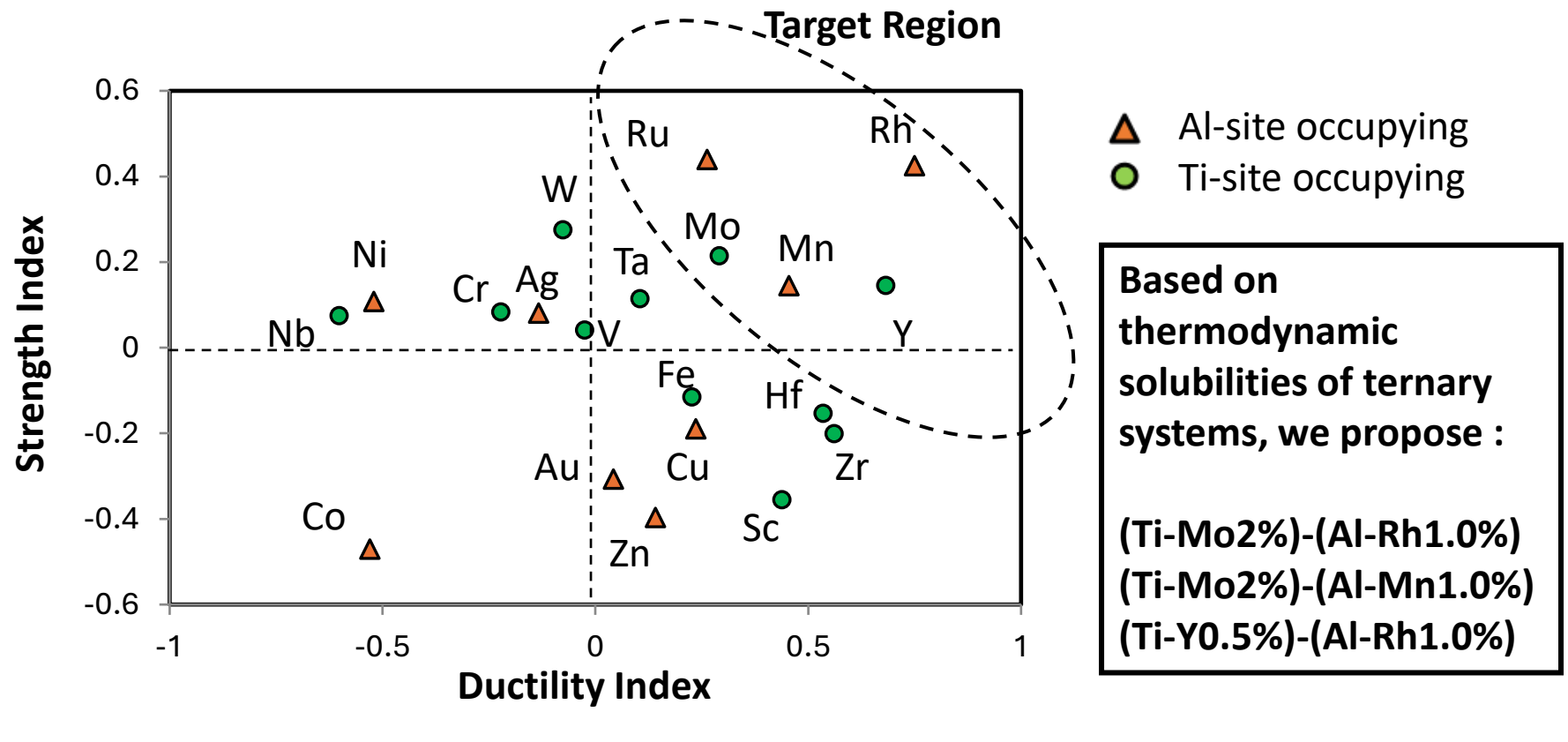
4) Oxidation resistance

5) Cost

- Oxidation resistance and cost are qualitatively defined, based on conclusions from the literature





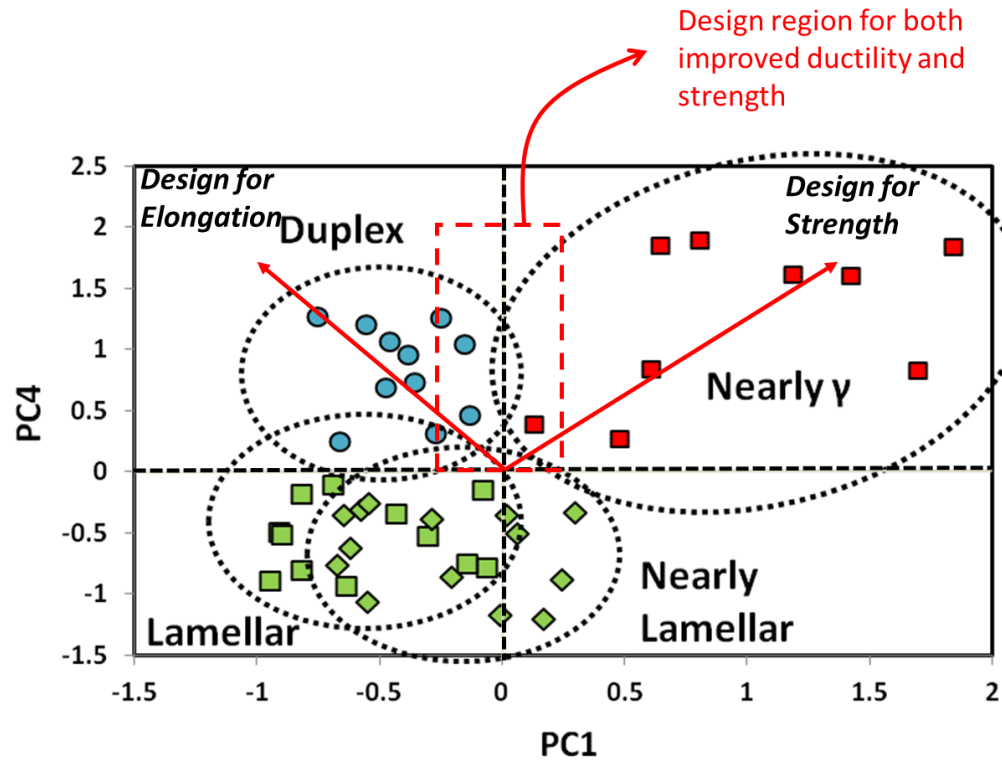


Overlay other information :

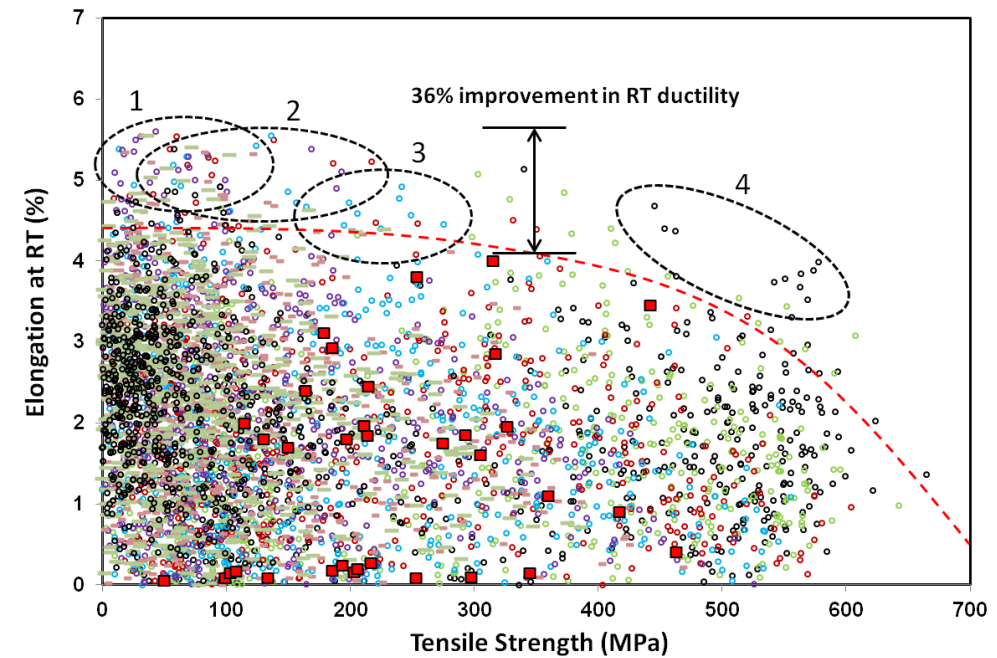
- Site occupancies from DFT
- Solubilities from thermodynamic modeling

- From regression models, the parameters for each property can be defined, and mapped to define the indices associated with each property. An alternate approach to PCA like we talked about last time.
- Eg. : % elongation = $f(\text{chemistry}) \rightarrow$ index based on the weighting of the individual components
- Higher index means as that elements composition increases, the property also increases relative to the index.

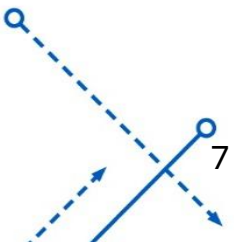
Microstructural Design



Chemical Design



The circled regions correspond to 3,4,5-d valencies for the chemical additions (eg. Cr, Mn, Ta)



Incorporating DFT

High B/G

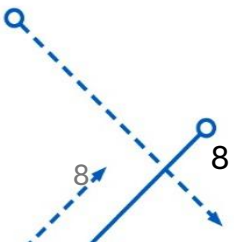
↑

Low B/G

Ti Occupying Element			Al Occupying Elements		
0 K (Without ZPE)	0 K (With ZPE)	300 K	0 K (Without ZPE)	0 K (With ZPE)	300 K
Mo	Mo	Mo	Ru	Ru	Mn
Y	Y	V	Mn	Mn	Ru
Fe	Fe	Nb	Rh	Rh	Rh
Cr	W	W	Ir	Ir	Pd
W	Cr	Ta	Ni	Ni	Cu
Nb	Nb	Zr	Co	Os	Ag
Zr	Zr	Cr	Os	Cu	Zn
V	V	Y	Pt	Pd	Ir
Ta	Ta	Hf	Re	Pt	Au
Sc	Hf	Fe	Cu	Re	Pt
Hf	Sc	Sc	Pd	Co	Re
			Ag	Zn	Cd
			Au	Ag	Ni
			Zn	Au	Os
			Cd	Cd	Hg
			Hg	Hg	Co

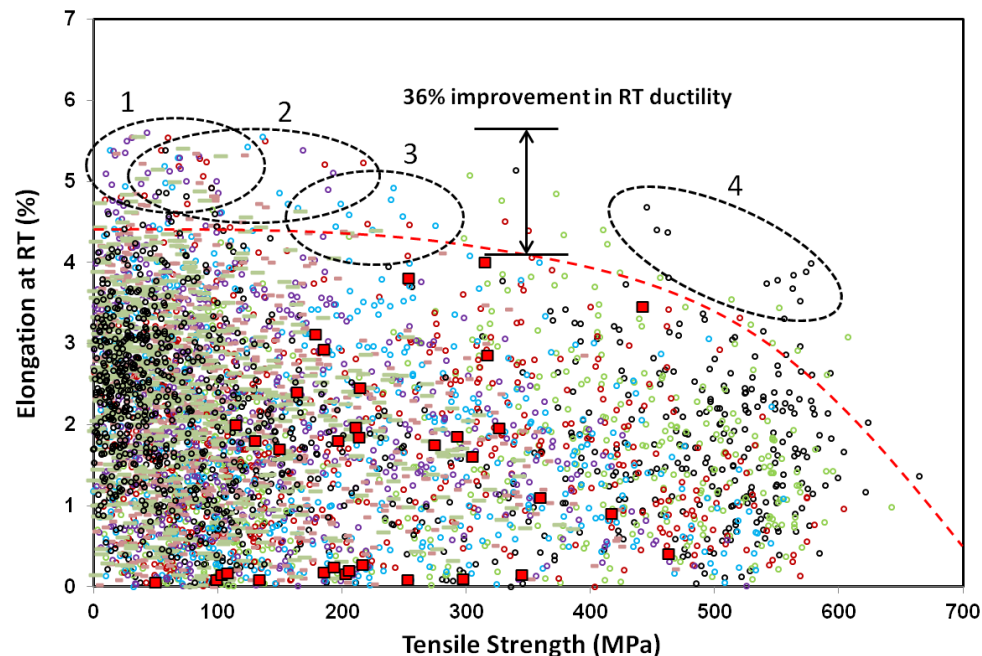
➤ Ordering of improvements with additions to Ti-Al, for the case of DFT calculations at 0K without ZPE (zero point energy), at 0K with ZPE, and at 300K through inclusion of quasi-harmonic effects.

- Identified alloying additions which improve the room temperature ductility, with our identified additions agreeing with experimental engineering data.
- The largest differences, highlighted in blue for the improvements and in orange for the worsening of properties, are seen for V, Nb, and Ta as Ti occupying additions.

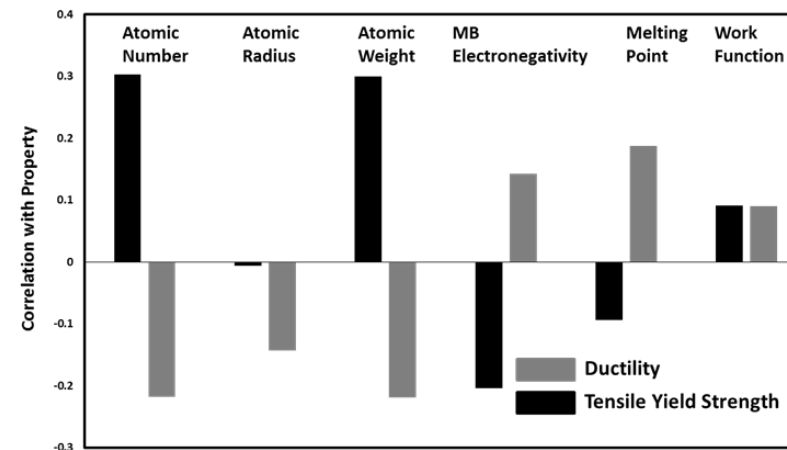


Screening the Design Space

- Developed quantitative structure-property relationships (QSPR) linking ductility with chemistry and linking strength with chemistry.
- From the QSPRs, spanned the chemical search space ($\sim 6,500$ chemistries) and plotted the outputs of the QSPRs versus each other.
- Screened the QSPR map and identified 4 compositional ranges which have improved ductility without sacrificing strength.

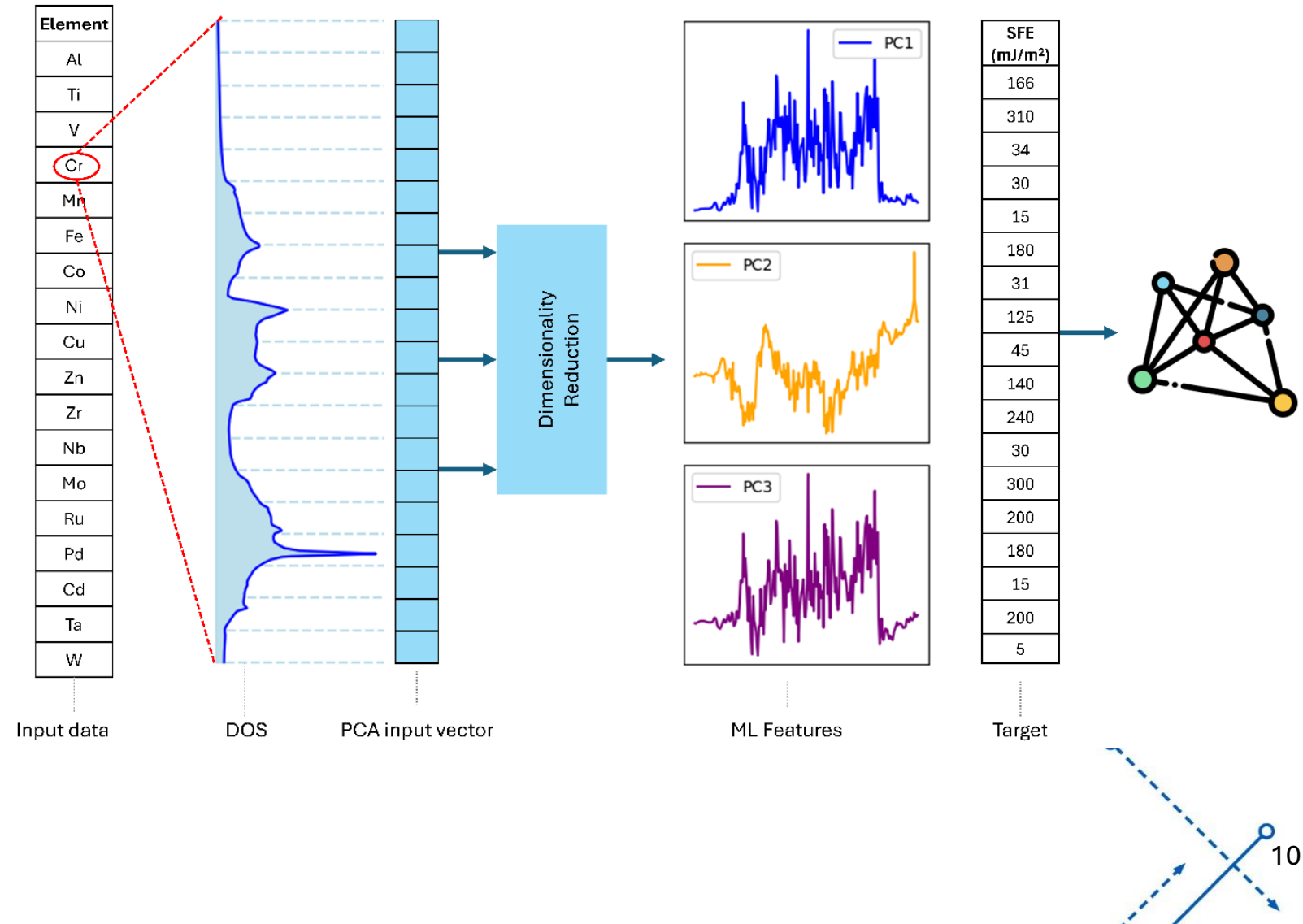


1: (46.1-47.0)Ti-(46.3-46.7)Al-(1.7-2.1)Cr-(1.0-1.1)Mn-(2.4-2.7)Ta-(0.1)B
 2: (46.8-48.0)Ti-(46.8-47.2)Al-(1.45-1.75)Cr-(1.3-1.45)Mn-(2.4-2.7)Ta-(0.1-0.15)B
 3: (47.9-48.7)Ti-(47.4-47.7)Al-(1.0-1.1)Mn-(0.8-1.1)Mo-(2.1-2.25)Ta
 4: (46.2-47.4)Ti-(47.3-47.6)Al-(1.35-1.55)Mn-(1.0-1.2)Mo-(2.8-3.2)Ta-(0.2-0.3)Si

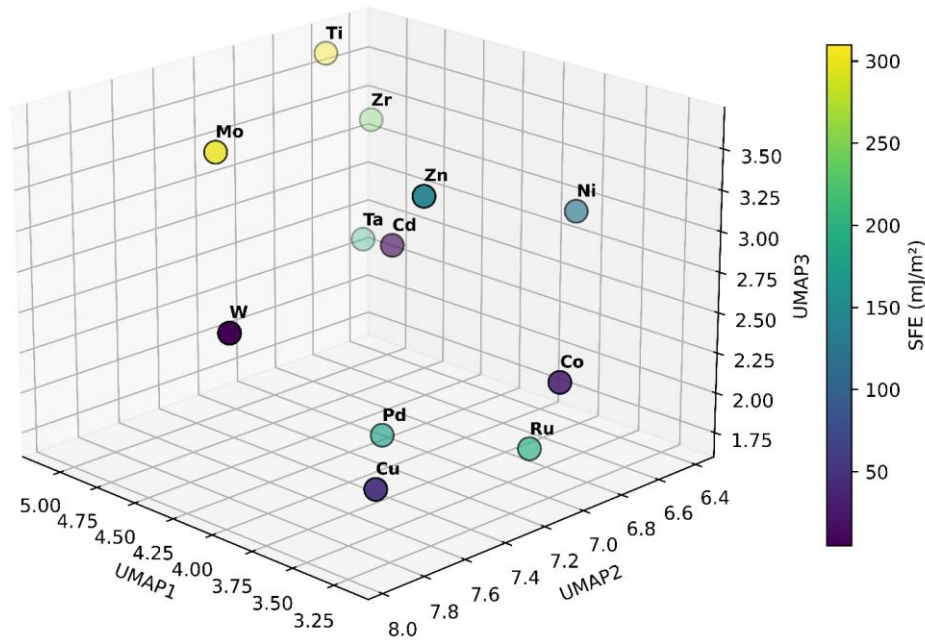


One More Example: Electronic Fingerprints to Mechanical Property

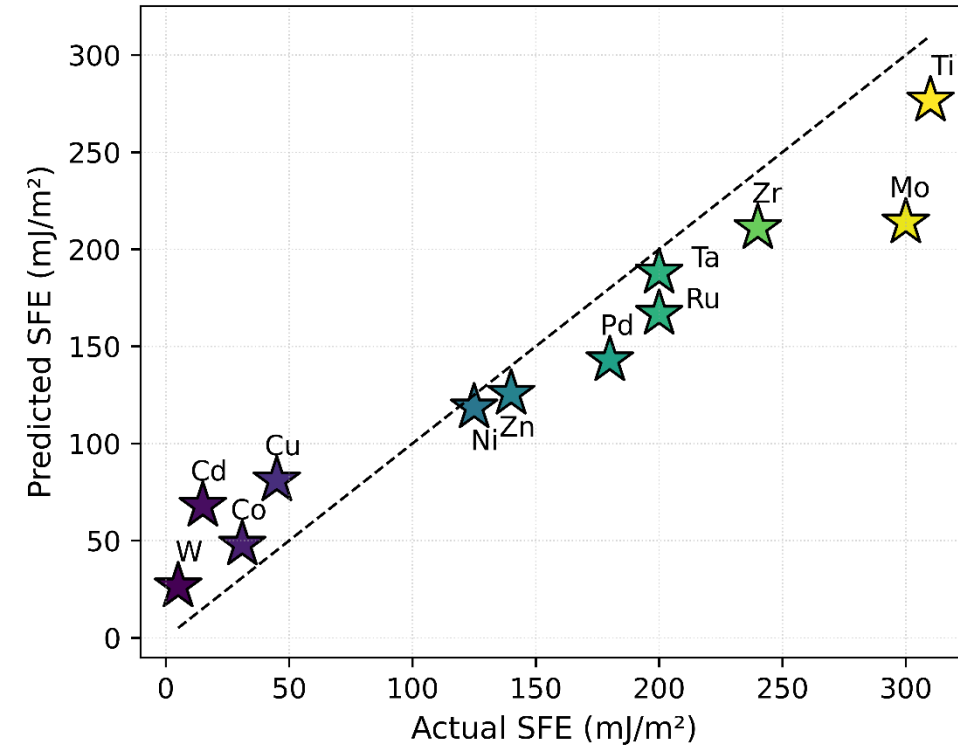
- Add in electronic structure (Density of States spectra) ... does this provide any added level of information in alloy design?
- Capture electronic fingerprints governing mechanical properties.
- Data input: Electronic Density of States (DOS) for single elements and binary chemistries



Electronic Fingerprints to Mechanical Property



- Position of elements in the reduced dimension representing similarities and dissimilarity.
- Provides guidelines on design of materials with desired property (SFE).

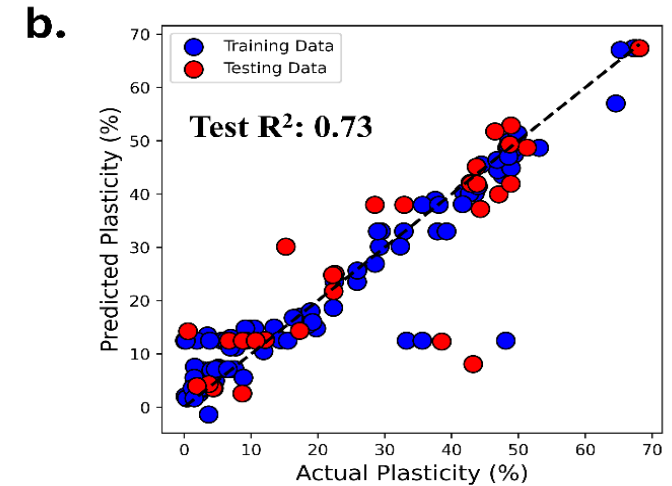
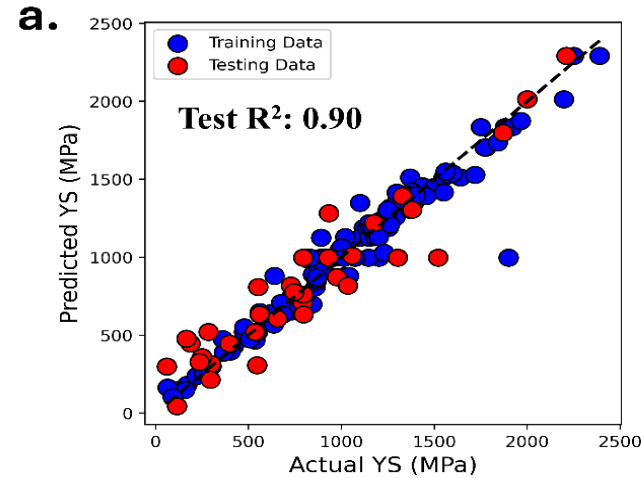


- Microstructural property being predicted as function of electronic DOS connecting two different length scales.

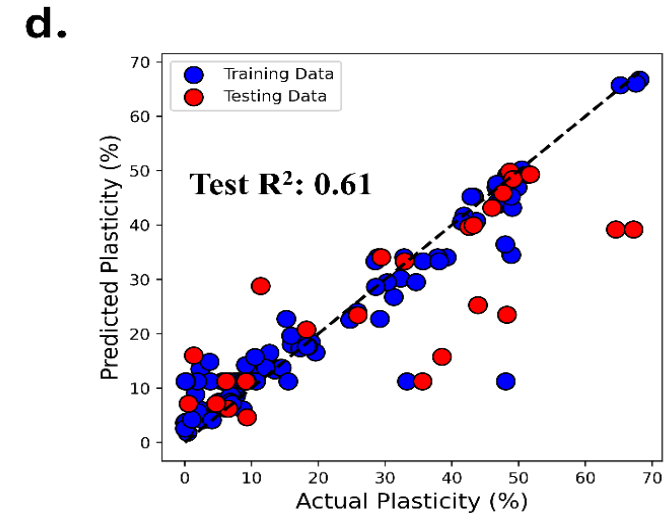
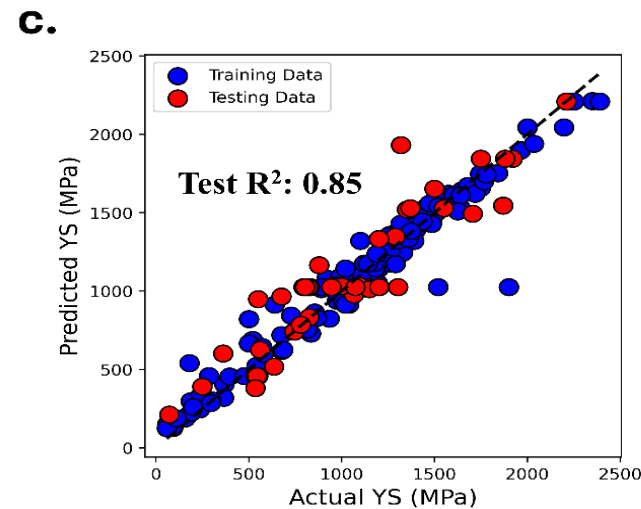
Enhancement of Model Accuracy

- DOS is included to prior models for prediction of mechanical properties (yield strength and plasticity)
- Models which include DOS perform significantly better.
- Capturing the energy-dependent DOS features helps describe atomic bonding strength and deformation mechanisms.

Input: Chemistry + DOS

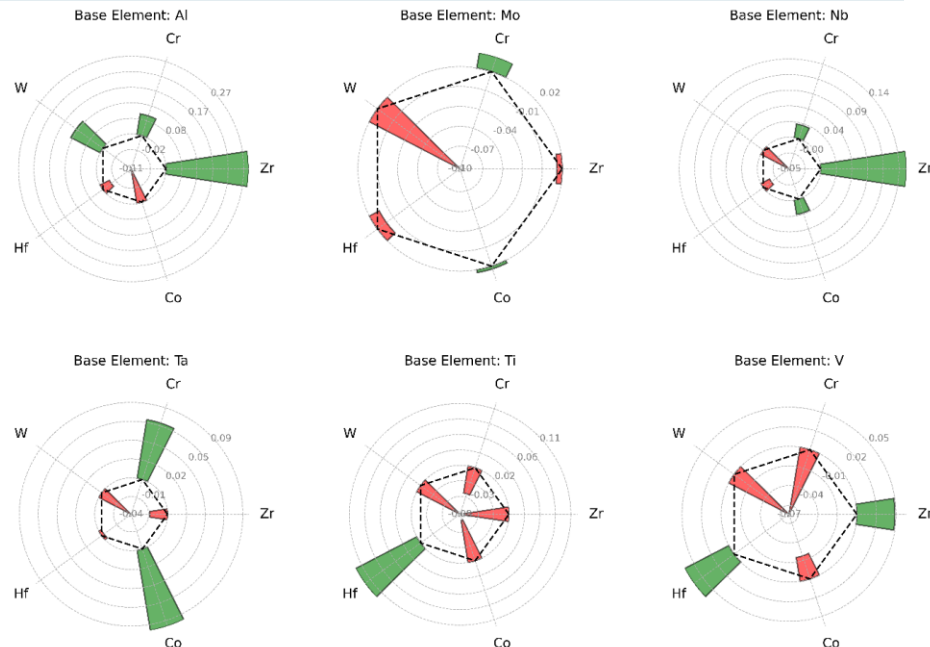


Input: Chemistry Only

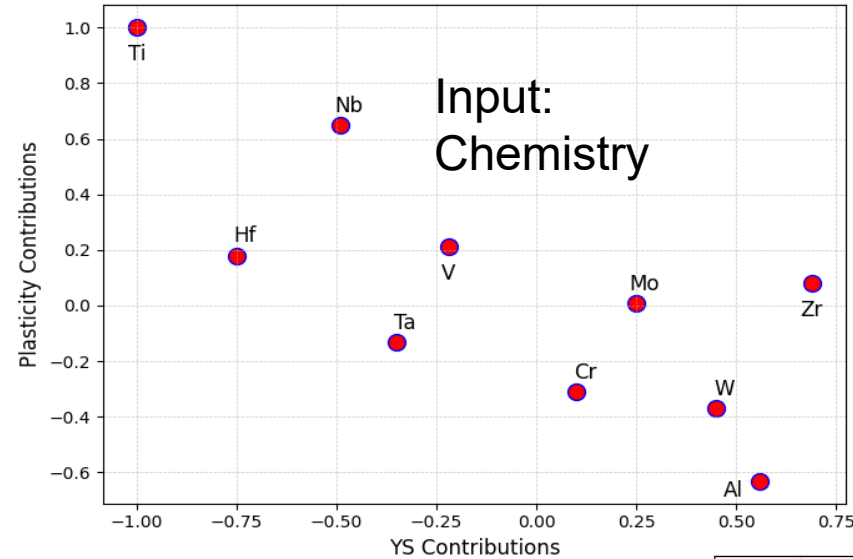


New Relationships from DFT Based Inputs

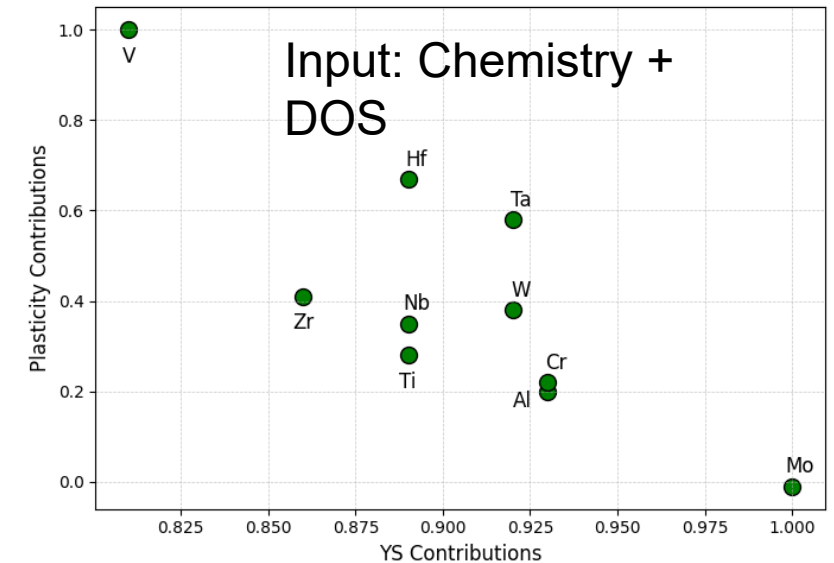
Development of new DFT based design maps for HEA from DFT inputs across different length scales ... multiscale design of alloy chemistries

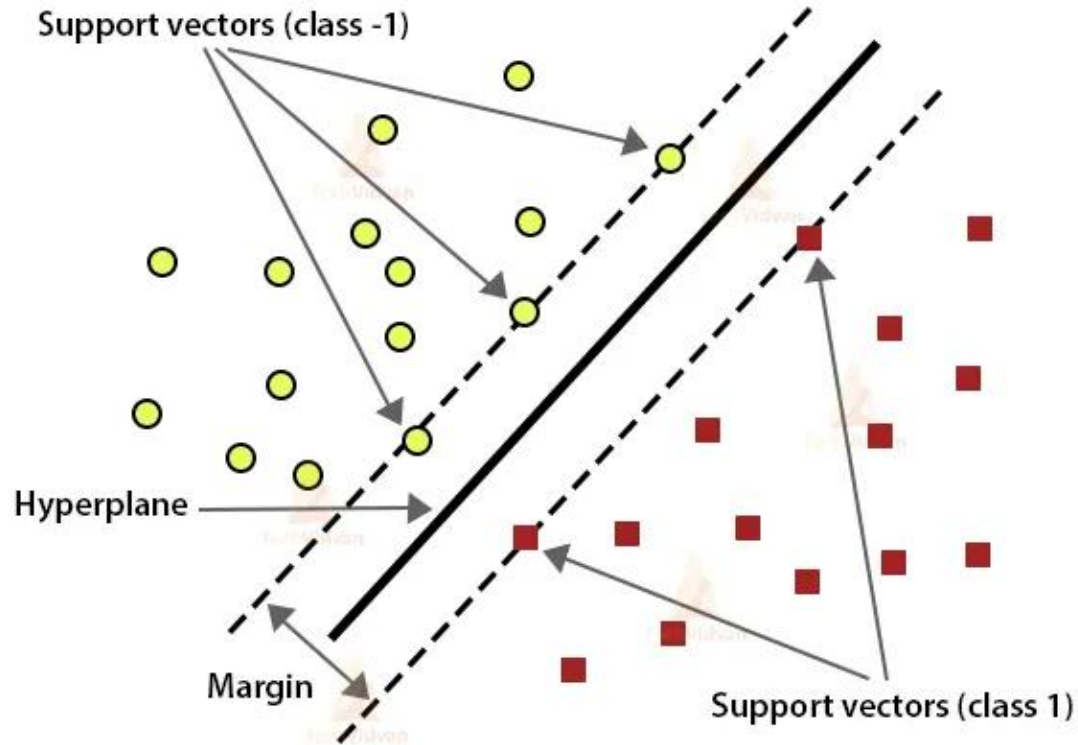


Chemistry – Phase Correlations

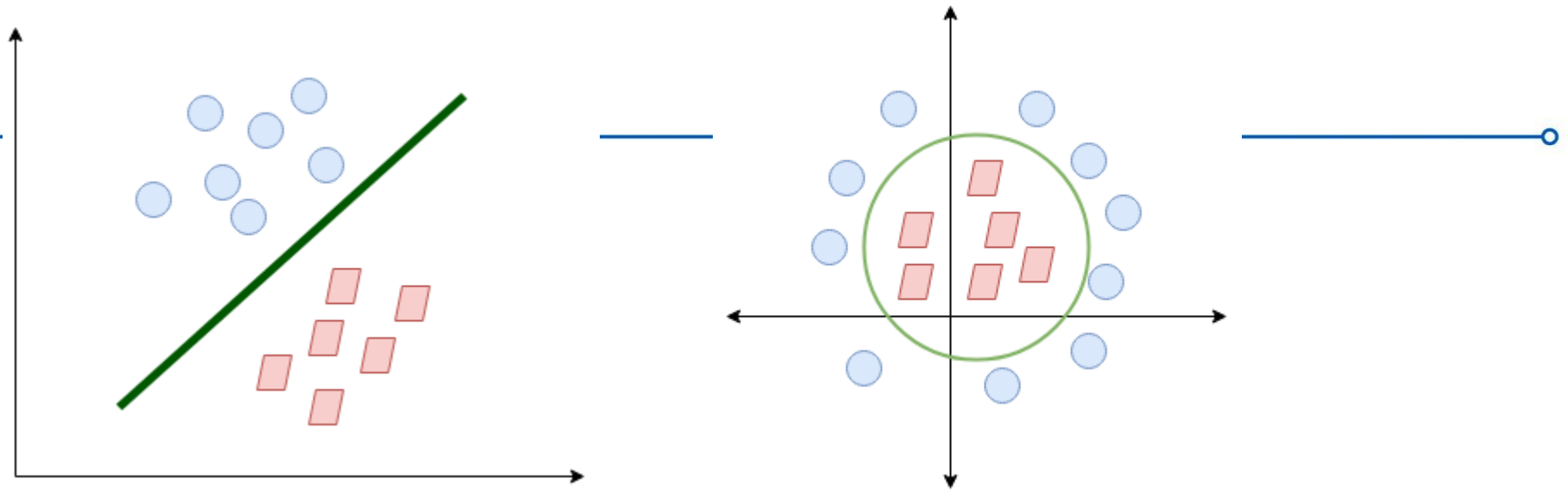


Chemistry – Mechanical Property Correlations



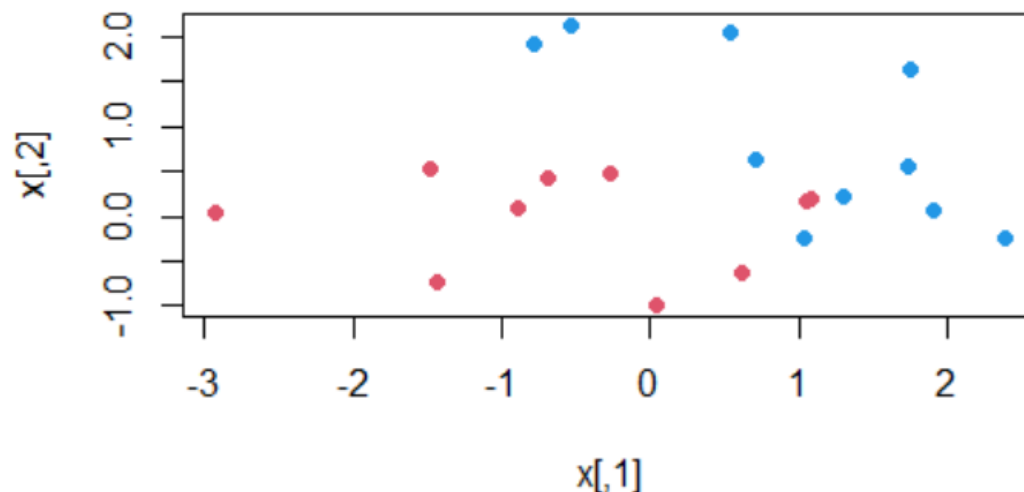


- Support vectors : The points closest to the decision boundary : ie. those that define the decision boundary.
- The goal of the maximal margin classifier is to identify the linear boundary that maximizes the total distance between the line and the closest point in each class.



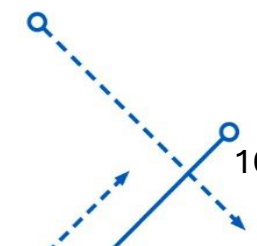
- In left figure, it is easy to understand the definition of the hyperplane, as it is just a line separating the two classes.
- In the figures on the right, it is a bit more complicated, as top we don't have a line we can draw However, if we look across n-dimensions, we can find a plane which maximizes the separation So the separation doesn't appear linear, but it is in some projection
- The use of SVM is more obvious when we have high dimensional data That is, where we are not able to just easily visualize the data. For the examples we'll talk about, it will be mostly in 2D for purpose of visualization, but keep in mind that we could be considering as high dimensional but projected in 2D.

I've randomly generated a 2D data set of 20 points, with 10 points belonging to one class (eg. -1) and the other 10 points to the other class (eg. 1)



- In this data, we can roughly see the separation of the data, but will also have some misclassification.

We have controllable parameters --- the kernel (in this case linear) and the cost The cost defines the penalty for points not falling beyond margin. That is, how much do we penalize the model for incorrect classifications or “fuzzy” classifications We can define how much to penalize for that and it will impact the result.



Parameters:

SVM-Type: C-classification

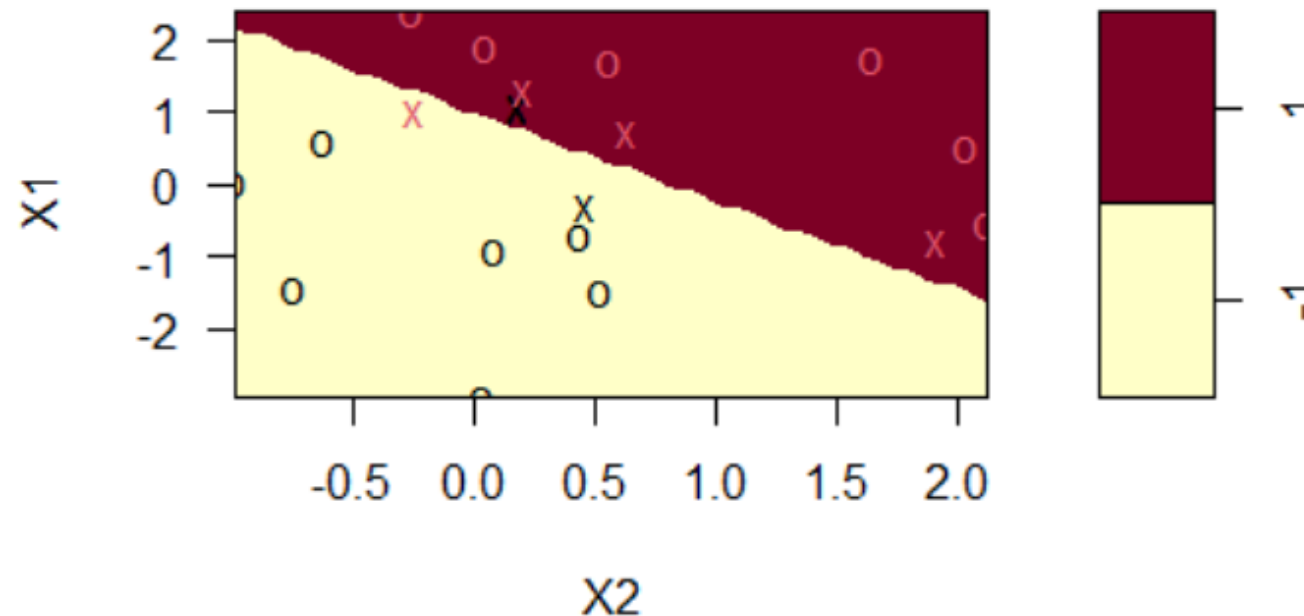
SVM-Kernel: linear

cost: 10

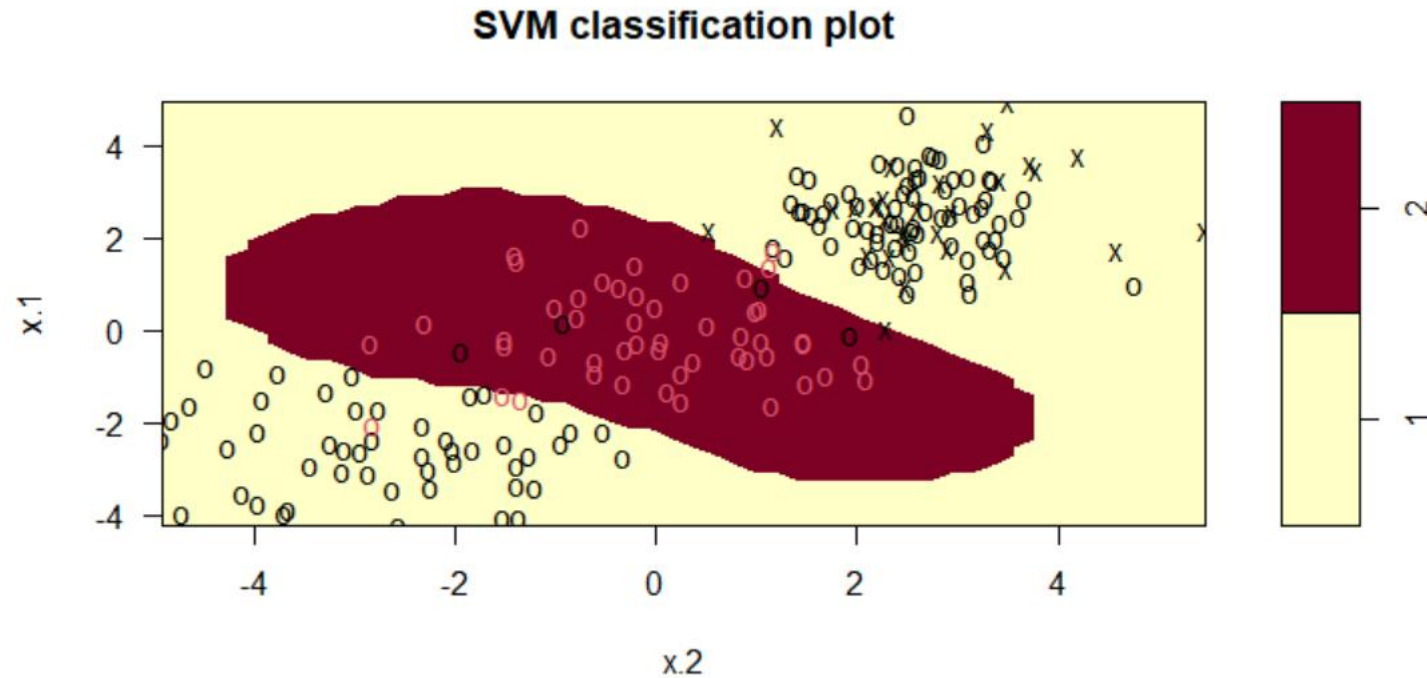
Number of Support Vectors: 7

Points which are
misclassified or close to
boundary (shown as x's in
the below plot)

SVM classification plot



Radial Kernel

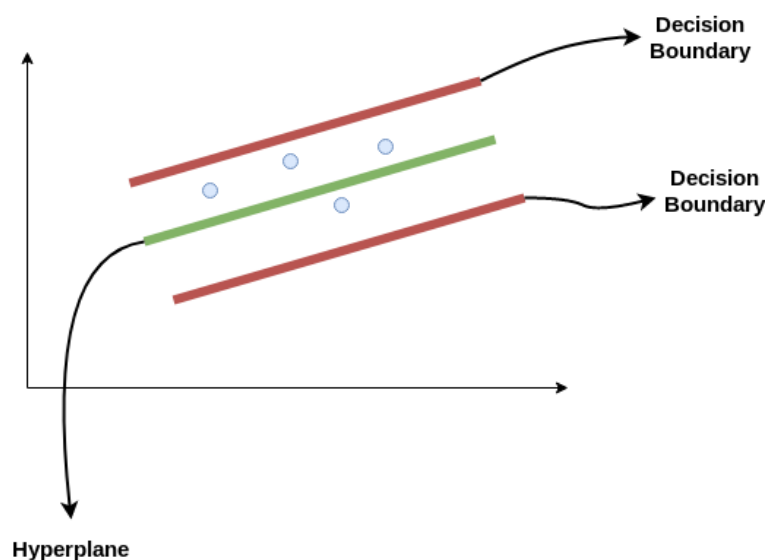


cost: 1
gamma: 0.5

*Scaling parameter for
non-linear boundaries*

Results in 67%
classification
accuracy

SVR uses the same basic idea as Support Vector Machine (SVM), a classification algorithm, but applies it to predict real values rather than a class. SVR acknowledges the presence of non-linearity in the data and provides a proficient prediction model.



The difference with the other regressions we've discussed is that we give it a range of fluctuation : so we are building in more flexibility in the model.... So in the example we are predicting within the decision boundaries and not just a single hyperplane like we were in linear regression.

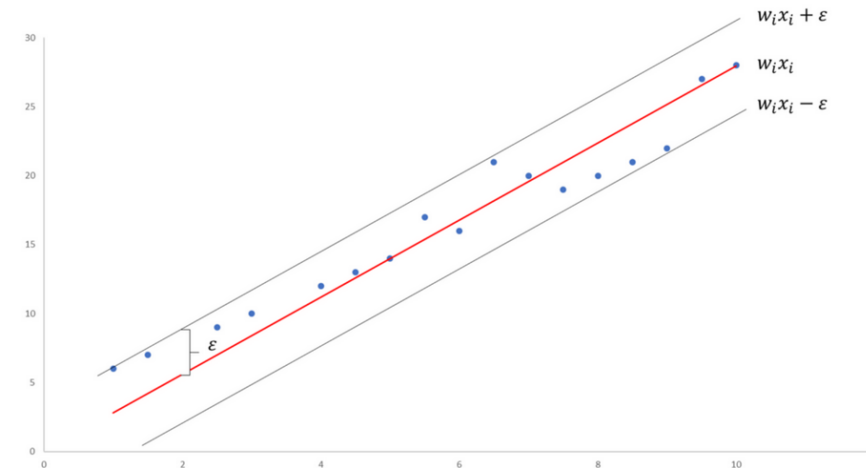
- In contrast to the other regression approaches we've discussed, the objective function of SVR is to minimize the coefficients and not the squared error.
- The error term is instead handled in the constraints, where we set the absolute error less than or equal to a specified margin, called the maximum error, ϵ (epsilon). We can tune epsilon to gain the desired accuracy of our model.
- Our new objective function and constraints are as follows:

Minimize:

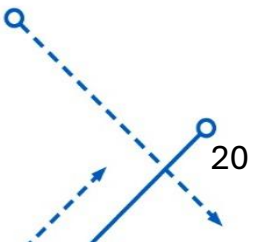
$$\text{MIN } \frac{1}{2} ||\mathbf{w}||^2$$

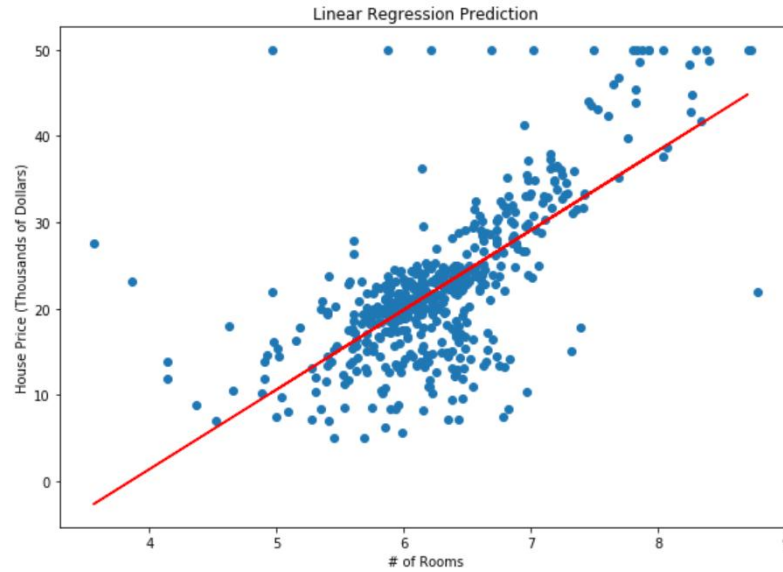
Constraints:

$$|y_i - w_i x_i| \leq \epsilon$$

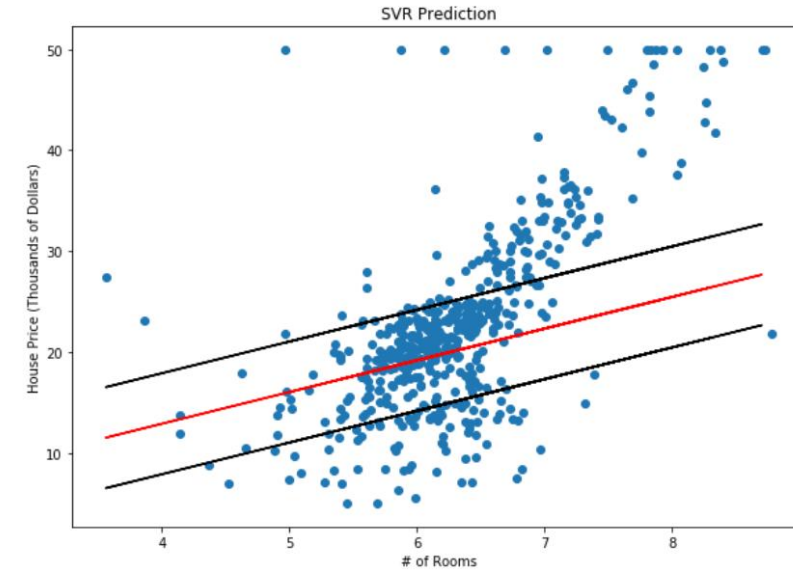


Illustrative Example of Simple SVR

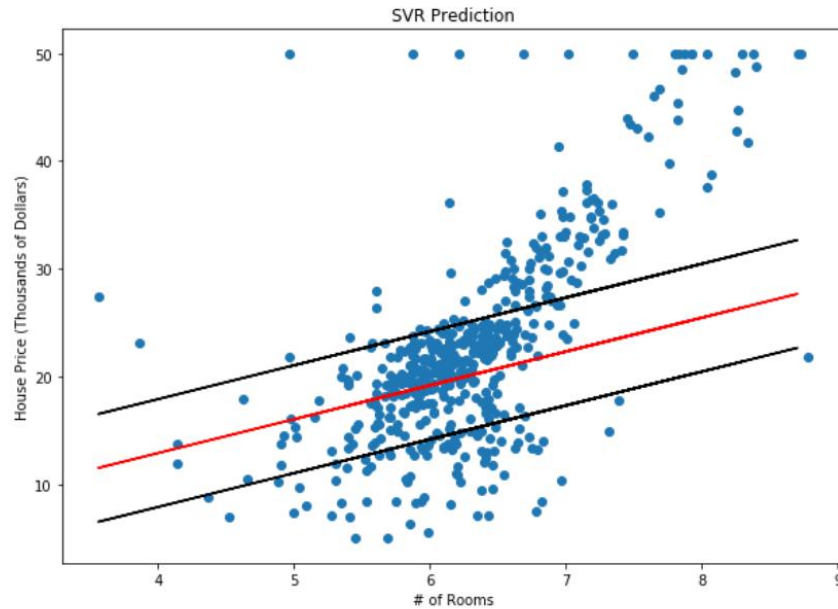




OLS Prediction of Boston Housing Prices

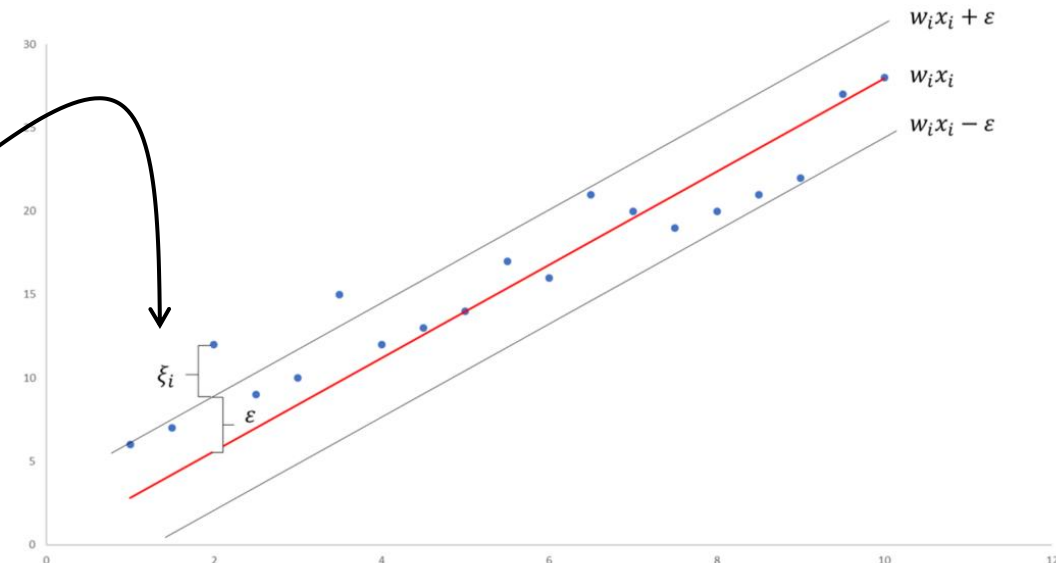

SVR Prediction of Boston Housing Prices with $\epsilon=5$

We see the change between linear regression model, which we try to maximize fit between predictor and predicted (left). The SVR result with $\epsilon=5$ (right), with the objective to minimize w (can think of it as the slope of the predictor line). In this case, we see that the fit is not good.... Looks worse than in the linear regression case Need to adjust parameters.

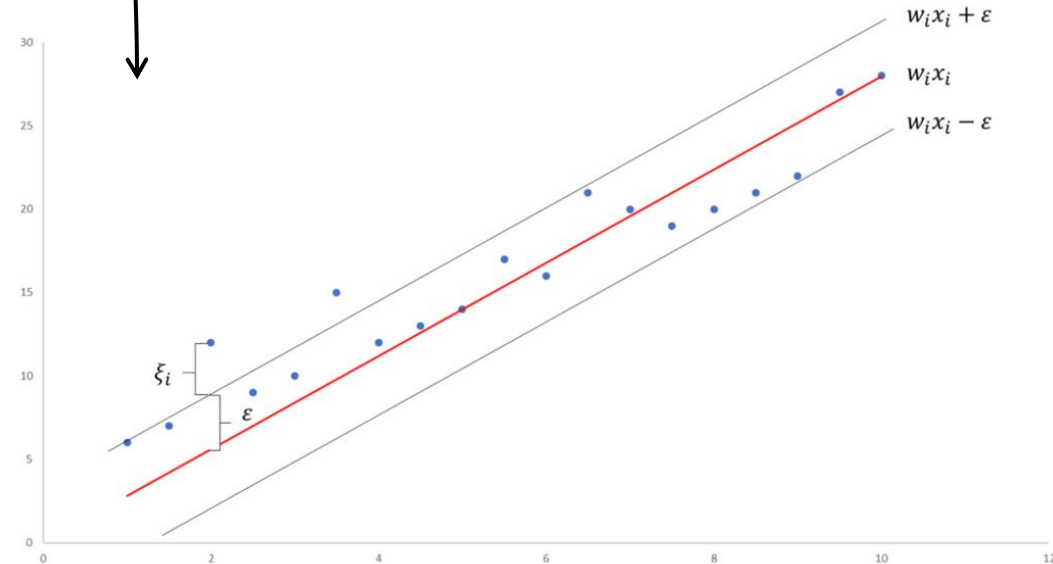

SVR Prediction of Boston Housing Prices with $\epsilon=5$

Also, we need to account for the possibility of errors that are larger than ϵ . We can do this with slack variables

$$\text{MIN } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n |\xi_i|$$



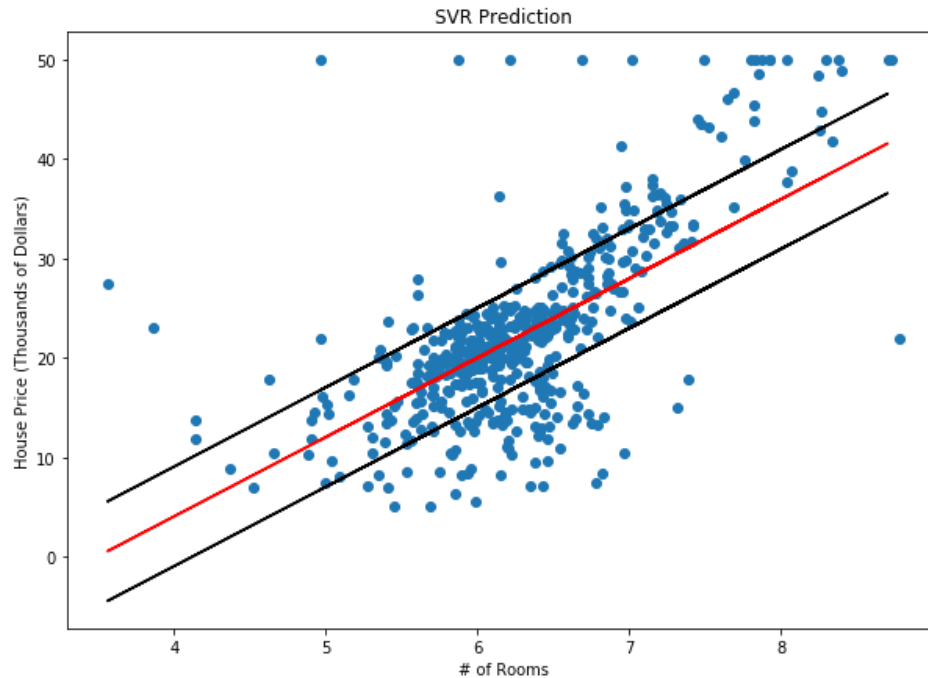
$$\text{MIN } \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^n |\xi_i|$$



So, what we have are the following to consider :

- Epsilon : what are the acceptable predicted values --- we don't care how close it is to the red line, as long as it is within epsilon of red line
- Each point falling outside of the accepted bounds is a support vector
- Cost function (C): how much do we penalize for falling outside lines and distance from lines This is basically what we are using to assess the model

Tuning epsilon and C to different amounts will determine different models

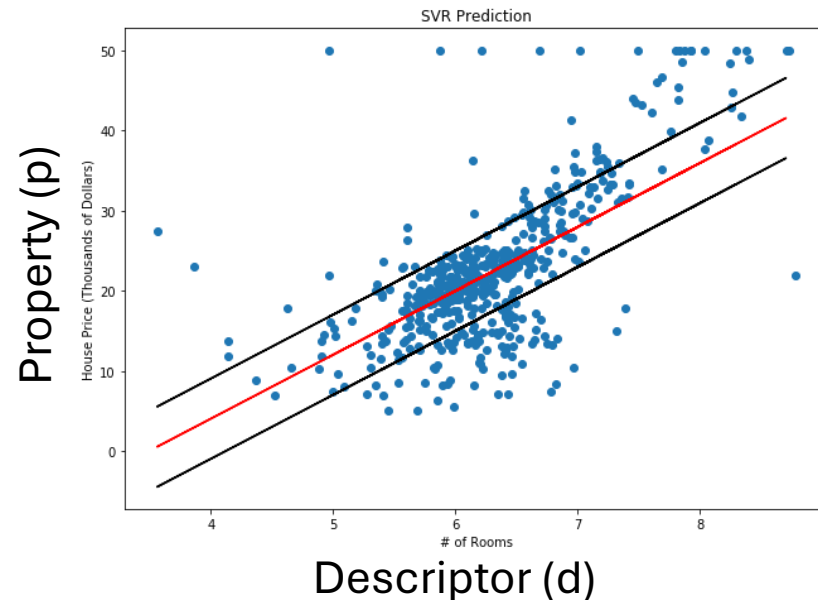


Result, with the best parameters found to be $\epsilon=5$, $C=6.13$.

Note, in this case, it still isn't very good. This highlights an important point that more involved regression approaches don't necessarily lead to improvement.

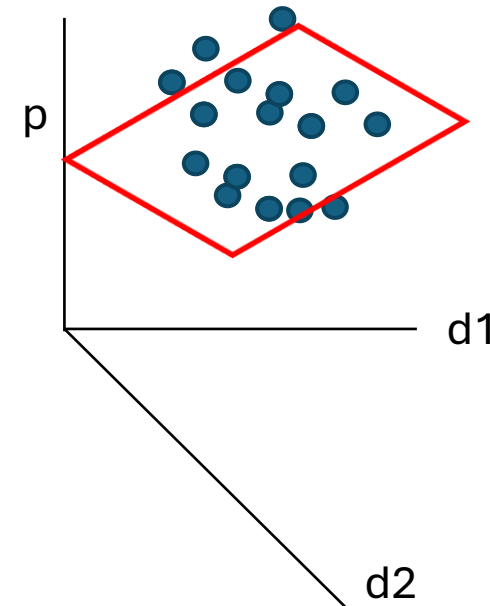
My general suggestion is to use the simplest method which gives a model of relatively equivalent accuracy to others Simpler, less computational costs, and most importantly more interpretability.

In SVR, we are attempting to predict a value of a property as a function of the descriptors or features. We define a hyperplane which gives the best predictive accuracy for the properties, given our other inputs.



Hyperplane (actually line in 2D case)

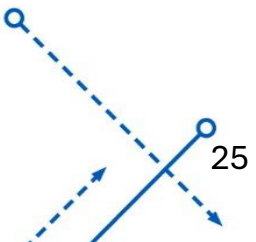
$$p = f(d)$$



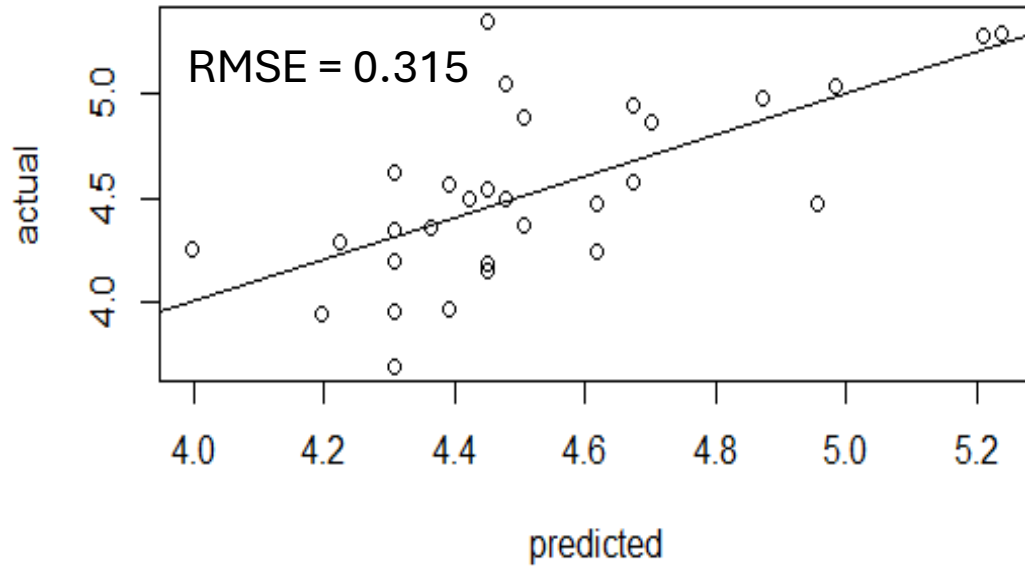
Now define a plane with a second descriptor :

$p = f(d1, d2)$
... Defined by weight matrix
... in this case $w1$ and $w2$

For n descriptors, we have : $p = f(d1, d2, \dots, dn)$, and we have a weight matrix $w1, w2, \dots, wn$: the weights are dependent on the user input parameters of cost and epsilon



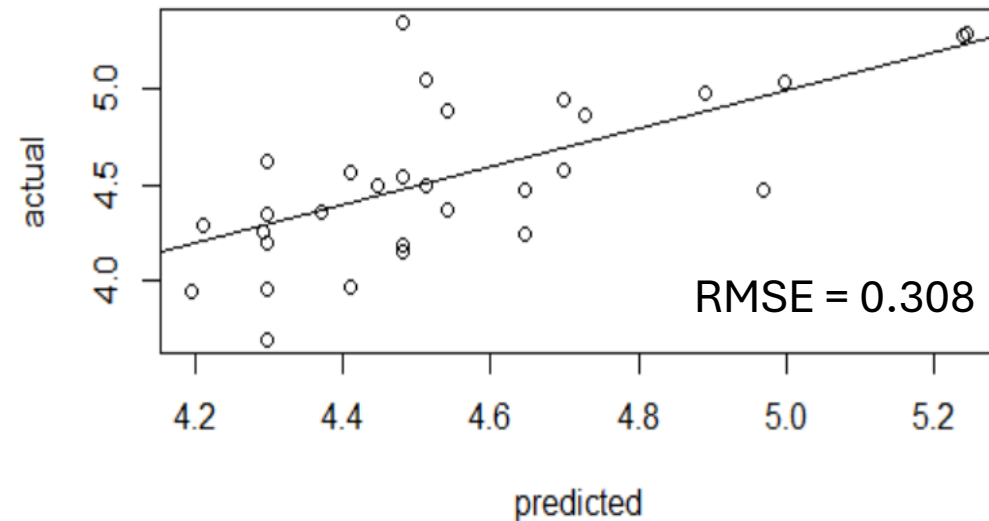
We'll just use batting average to predict runs per game



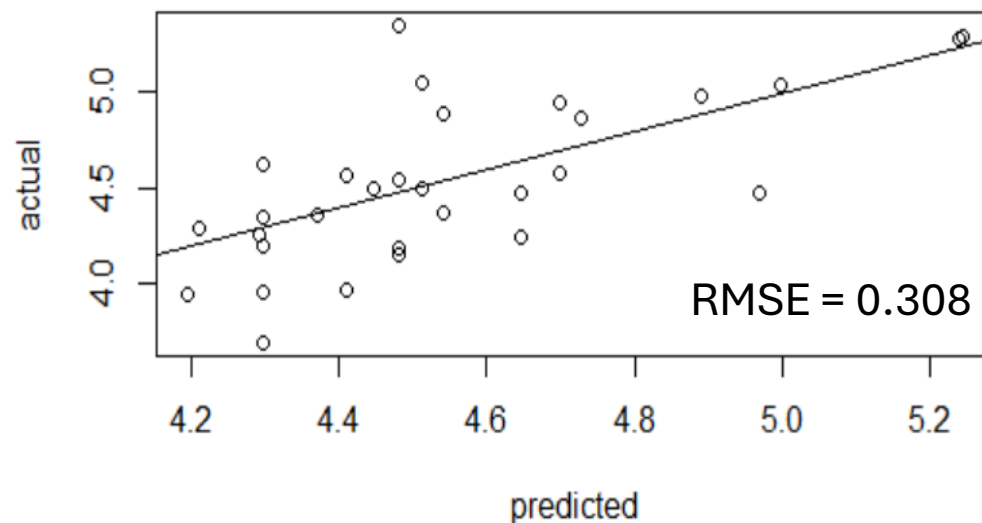
Linear regression
(like we had before)

SVR (using parameters
selected in R with svr
function)

Result looks almost same ---
slight improvement with SVR
.... ie. RMSE went down



Not tuning parameters :

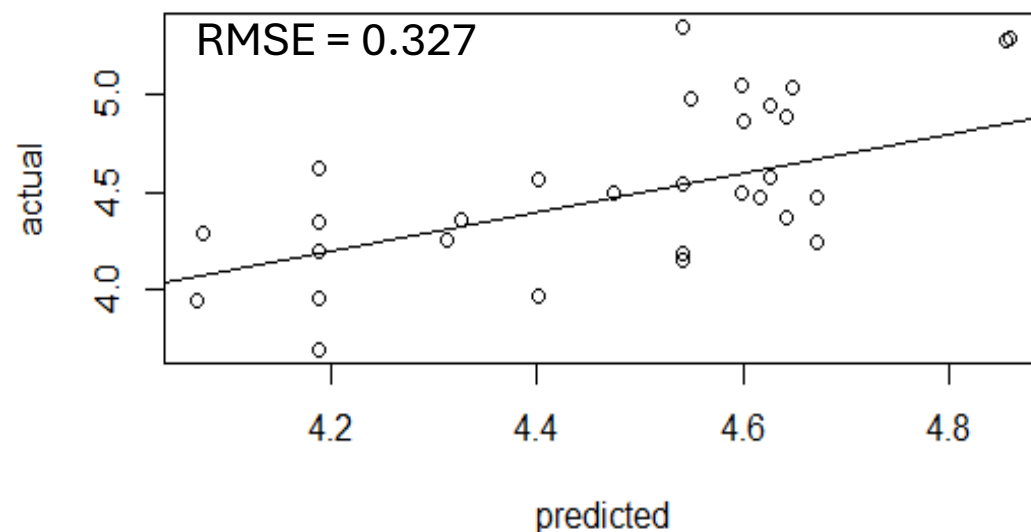


Parameters:

SVM-Type: eps-regression
SVM-Kernel: radial
cost: 1
gamma: 1
epsilon: 0.1

Number of Support Vectors: 28

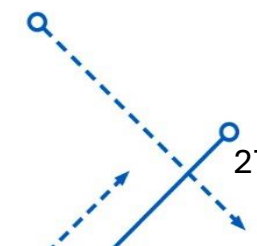
With tuned parameters :

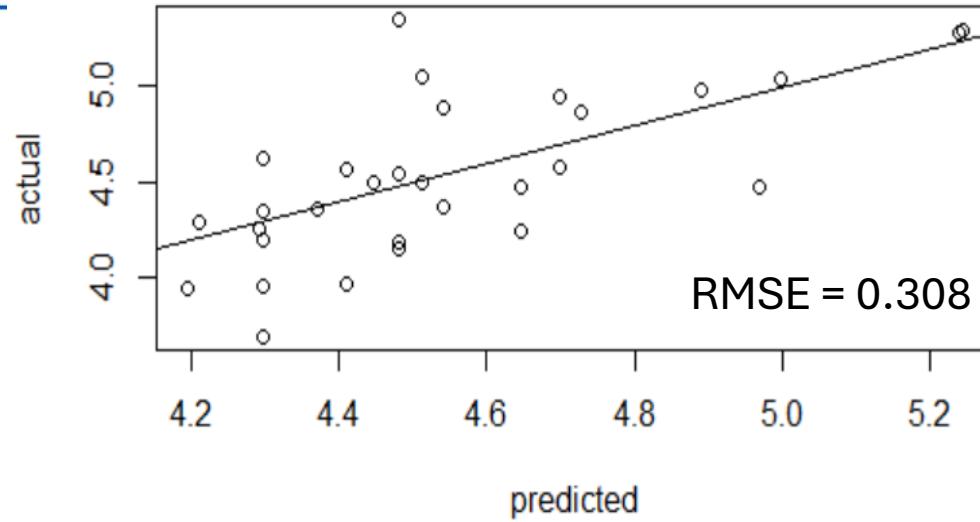


Parameters:

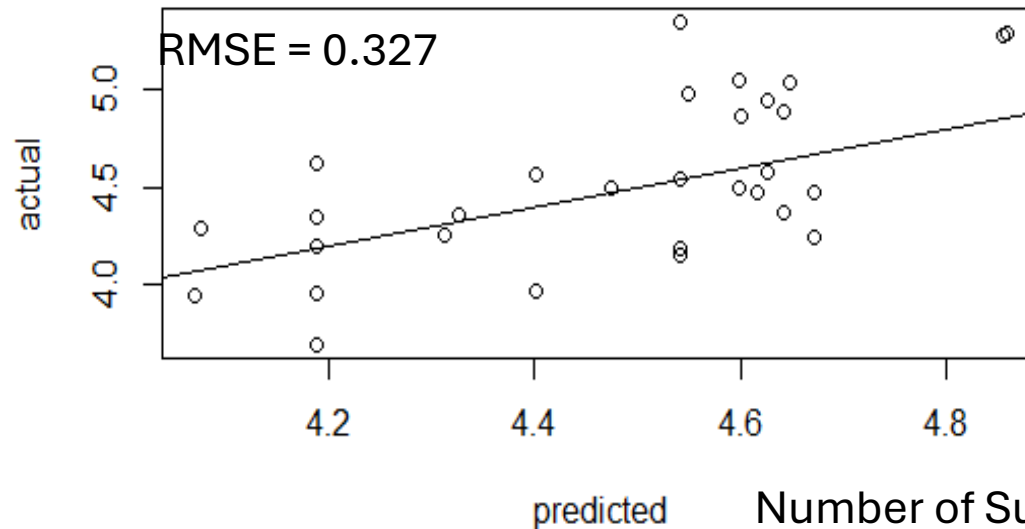
SVM-Type: eps-regression
SVM-Kernel: radial
cost: 2
gamma: 1
epsilon: 1

Number of Support Vectors:
8





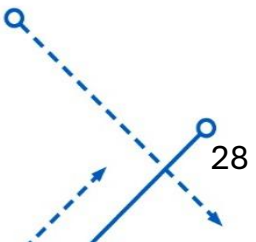
Number of Support Vectors: 28



Number of Support Vectors: 8

When tuning parameters, we are finding the best combination of parameters for the model, so we would always expect an improvement in accuracy.....

That is not seen here ... why?



- Tuning uses 10-fold cross validation as a default
- What this means is that the data is divided into 10 equal sized groups (or folds) and each time we run the model, 9 folds are used for training and 1 fold for testing. Then it is run 10 times, with each fold used once for training.
- Can change the fold... so for example cross validation=5 would instead perform 5 fold cross validation (giving us 80%/20% training/test split).

Recap of SVM and SVR

In SVM, we want to separate classes ... so we define the hyperplane to :

- Maximize classification accuracy of support vectors (ie. correctly predicting some condition is one class or another)
- Maximize the distance of support vectors from the hyperplane.

The balance of these two is based on the cost value.

The support vectors are those points closest to the boundary (ie. changing their feature values a little will result in different hyperplane.

In SVR, we want to predict properties ... so we define the hyperplane to :

- Maximize prediction accuracy
- Minimize the weights

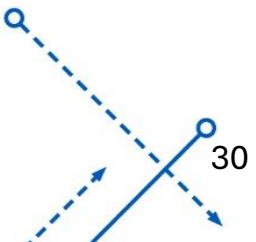
The balance of these two is based on the cost value.

The support vectors are those points outside decision boundary The decision boundary is based on how we define epsilon Too large epsilon, and model is too general, too small epsilon and the model is potentially over-fitting.

Increasing the cost value, will put higher penalty in errors in the predictions (given a value of epsilon), while decreasing the cost value will put higher penalty on larger weight values.

These parameters are tuned to find which combination of epsilon and cost give the 'best' model (somewhat subjective)..... Important to consider cross validation.

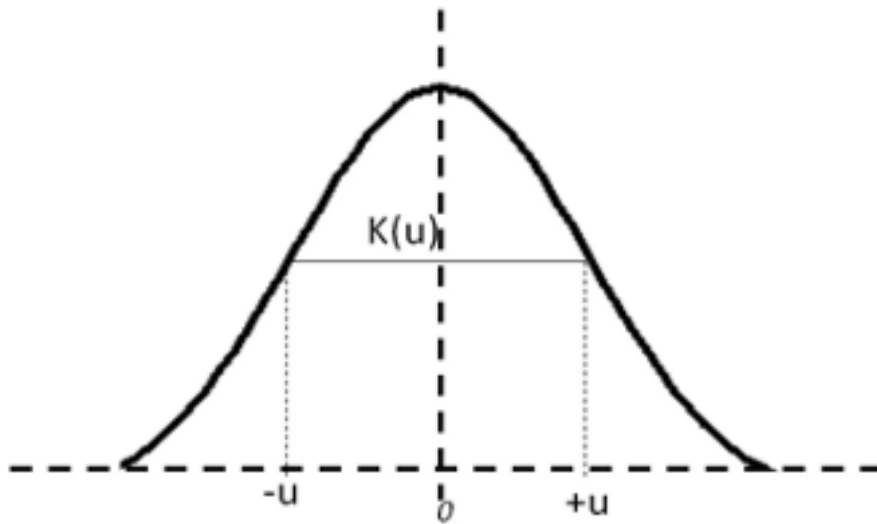
In both cases, we can select different types of kernels, which also results in consideration b/w accuracy and robustness



A Gaussian process is a probability distribution over possible functions that fit a set of points.

Because we have the probability distribution over all possible functions, we can calculate the means as the function, and calculate the variance to show how confident when we make predictions using the function.

- The functions update with new observations.
- The mean calculated by the posterior distribution of the possible functions is the function used for regression.



Let's now look at kernels in an alternate way (we'll focus on Gaussian kernels for now)

A kernel is a weighting function with the following constraints:

- It must be symmetric
- The max value of $K(u)$ falls in the middle
- The area under the curve is 1.
- $K(U)$ is never negative

$$K(x) = \frac{1}{h\sqrt{2\pi}} e^{-0.5\left(\frac{x-x_i}{h}\right)^2}$$

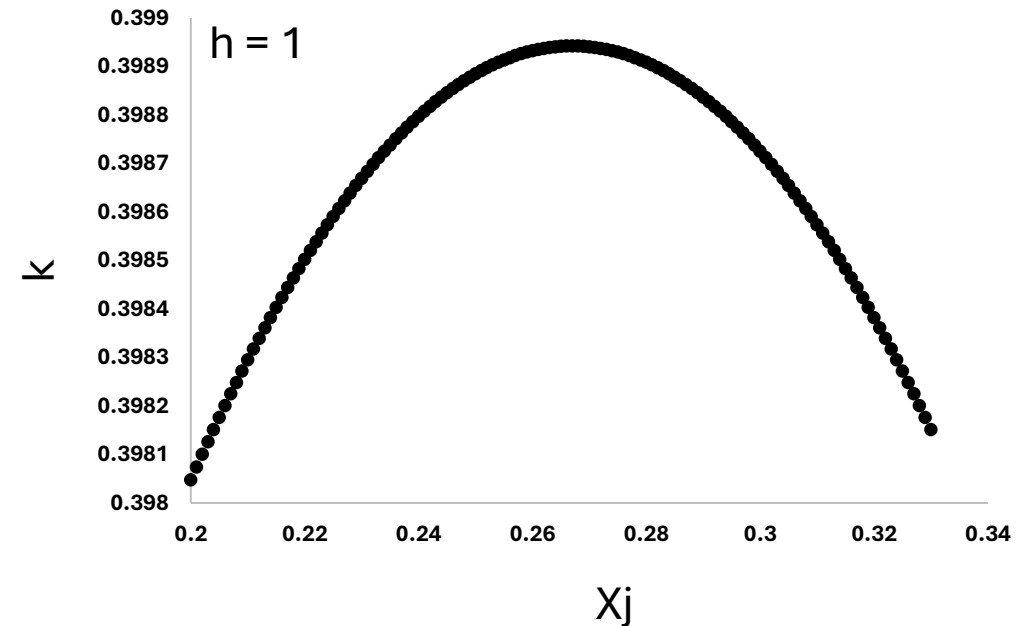
Where x_i is the observed value, x is the value where kernel is computed, and h is the bandwidth (also known as smoothing parameter)

The bandwidth is an adjustable parameter which will impact our model.

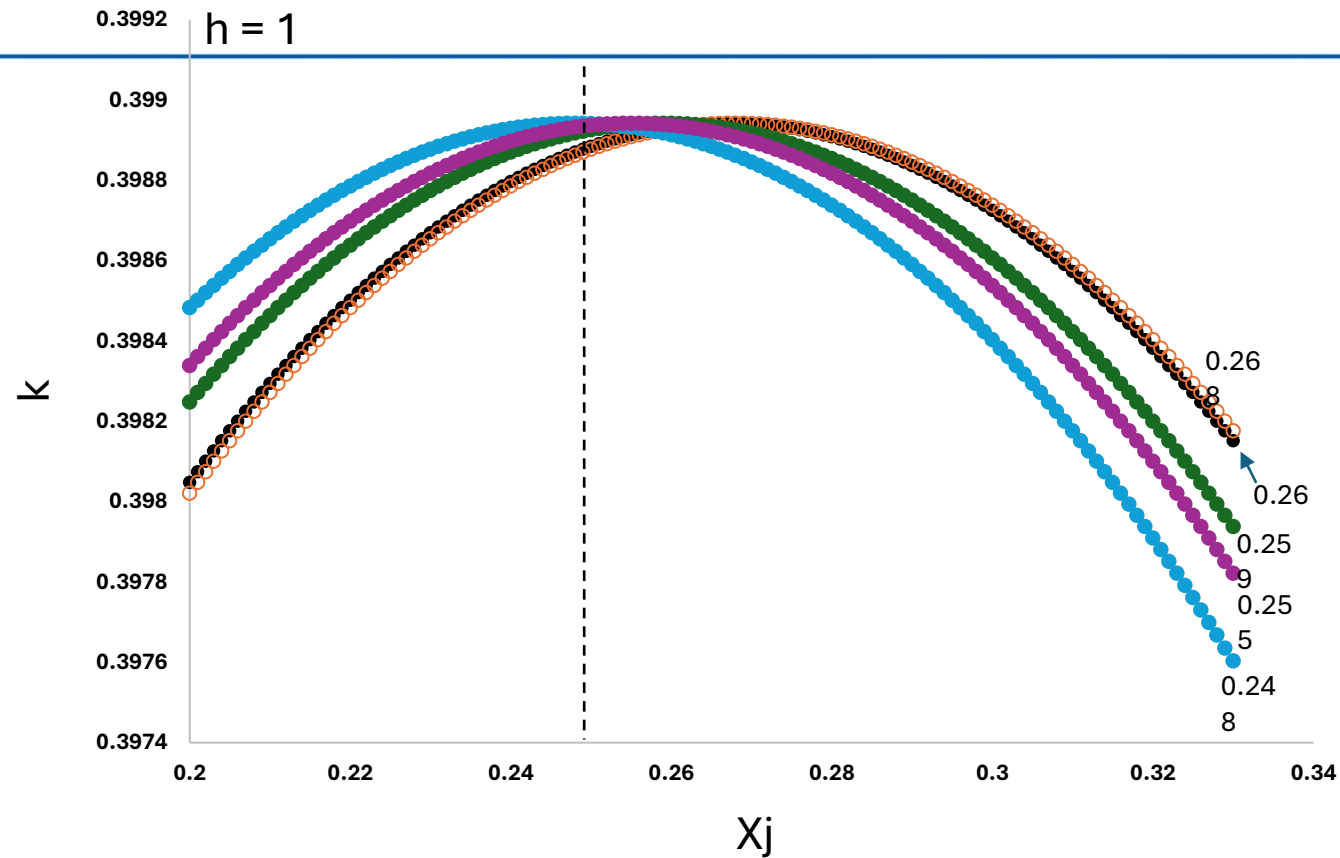
Xj	xi	h	A= 1/(h*sqrt(2pi))	B = -0.5*((Xj-xi)/h)^2	K = A*exp(B)
0.2	0.267	1	0.39894228	-0.0022445	0.398047859
0.201	0.267	1	0.39894228	-0.002178	0.39807433
0.202	0.267	1	0.39894228	-0.0021125	0.398100404
0.203	0.267	1	0.39894228	-0.002048	0.398126083
0.204	0.267	1	0.39894228	-0.0019845	0.398151364
0.205	0.267	1	0.39894228	-0.001922	0.39817625
0.206	0.267	1	0.39894228	-0.0018605	0.398200738
0.207	0.267	1	0.39894228	-0.0018	0.39822483
0.208	0.267	1	0.39894228	-0.0017405	0.398248525
0.209	0.267	1	0.39894228	-0.001682	0.398271823
0.21	0.267	1	0.39894228	-0.0016245	0.398294725
0.211	0.267	1	0.39894228	-0.001568	0.398317229
0.212	0.267	1	0.39894228	-0.0015125	0.398339336
0.213	0.267	1	0.39894228	-0.001458	0.398361046
0.214	0.267	1	0.39894228	-0.0014045	0.398382359
0.215	0.267	1	0.39894228	-0.001352	0.398403275
0.216	0.267	1	0.39894228	-0.0013005	0.398423793
0.217	0.267	1	0.39894228	-0.00125	0.398443914
0.218	0.267	1	0.39894228	-0.0012005	0.398463638

$$K(x) = \frac{1}{h\sqrt{2\pi}} e^{-0.5\left(\frac{x-x_i}{h}\right)^2}$$

Calculating one kernel for 0.267 (using our baseball data, with batting average)



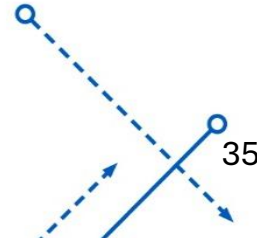
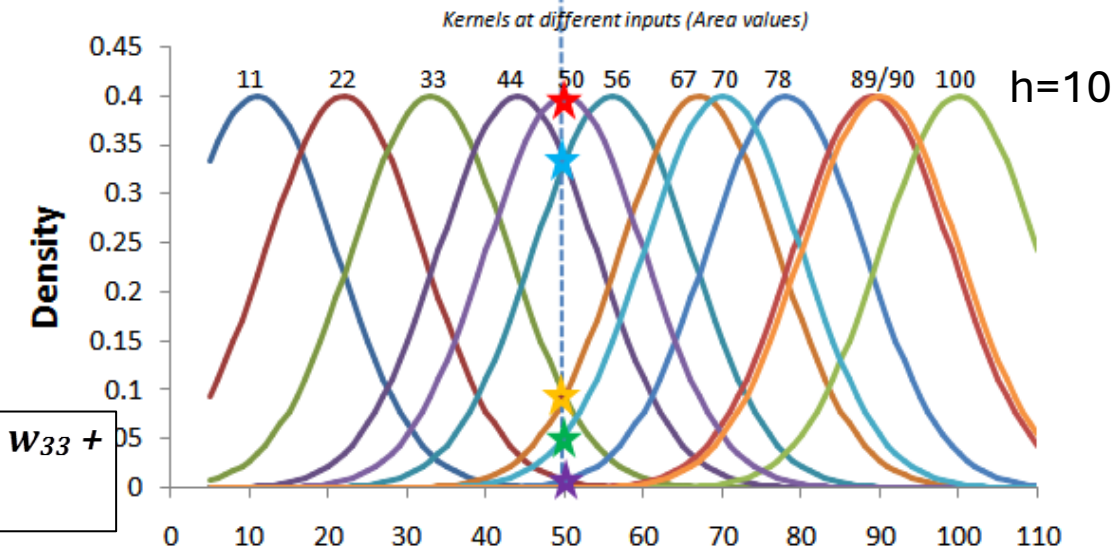
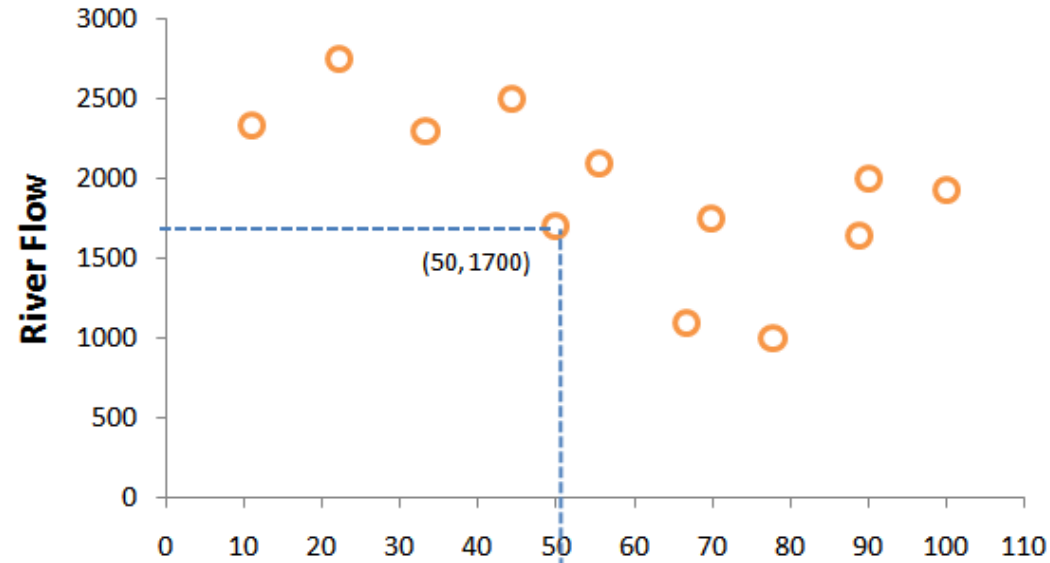
Expanding out to more kernels for batting averages



- For a given h , the model is based on the sum of the y -axis values for all kernels.
- For example, .248 has five kernels in this example, with each contribute a weight.
- This takes into account then the probability distribution, through having the weighting function of all possible observations

Area	River Flow
11	2337
22	2750
33	2301
44	2500
56	2100
67	1100
78	1000
89	1642
100	1932
50	1700
70	1750
90	2000

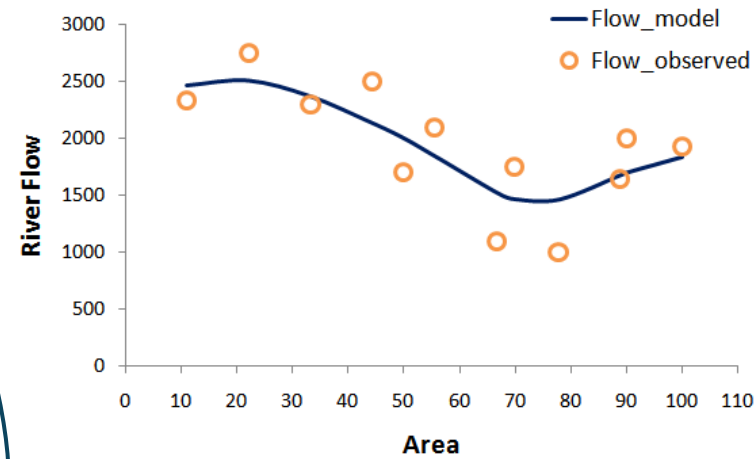
$$Y_{50} = y_{11} \times w_{11} + y_{22} \times w_{22} + y_{33} \times w_{33} + \dots + y_{100} \times w_{100}$$



Calculation of y for $x = 50$					
A	B	C	D	E	F
Kernels	Density	Col-B/Sum of Col-B = weight	River flow_Data	Col-D $\times$ Col-C	Sum of Col-F = Ymodel
K(11)	0.0001987	0.0001501	2337	0.35	2006.40
K(22)	0.0079155	0.0059791	2750	16.44	
K(33)	0.0940491	0.0710424	2301	163.49	
K(44)	0.3332246	0.2517098	2500	629.27	
K(56)	0.3332246	0.2517098	2100	528.59	
K(67)	0.0940491	0.0710424	1100	78.15	
K(78)	0.0079155	0.0059791	1000	5.98	
K(89)	0.0001987	0.0001501	1642	0.25	
K(100)	0.0000015	0.0000011	1932	0.00	
K(50)	0.3989423	0.3013514	1700	512.30	
K(70)	0.0539910	0.0407835	1750	71.37	
K(90)	0.0001338	0.0001011	2000	0.20	

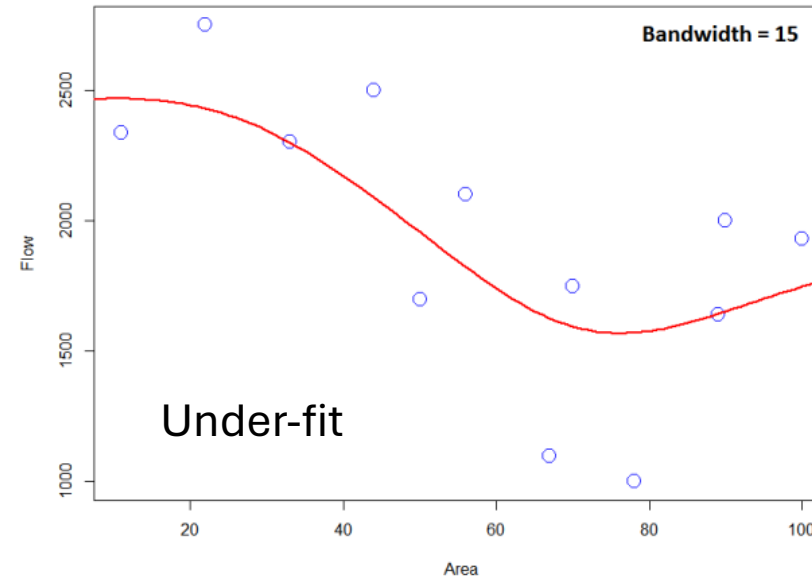
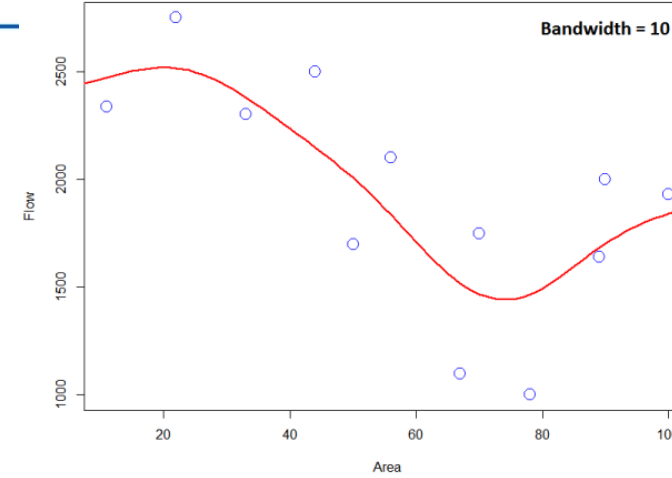
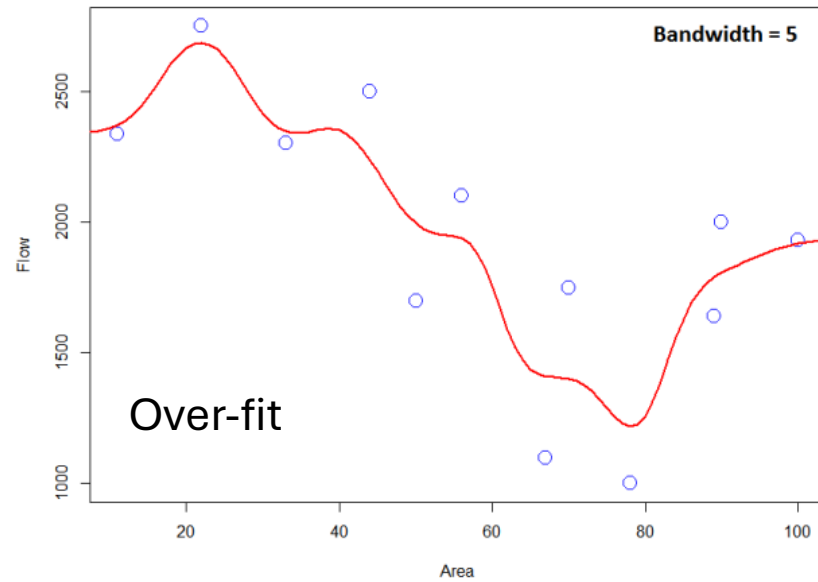
$$Y_{50} = y_{11} \times w_{11} + y_{22} \times w_{22} + y_{33} \times w_{33} + \dots + y_{100} \times w_{100}$$

Area	Flow_observed	Flow_model
11	2337	2473
22	2750	2516
33	2301	2380
44	2500	2148
50	1700	2006
56	2100	1836
67	1100	1518
70	1750	1468
78	1000	1467
89	1642	1683
90	2000	1702
100	1932	1841



Calculated in prior slide.
The process is then repeated for every x_i .

Kernal regression – impact of bandwidth

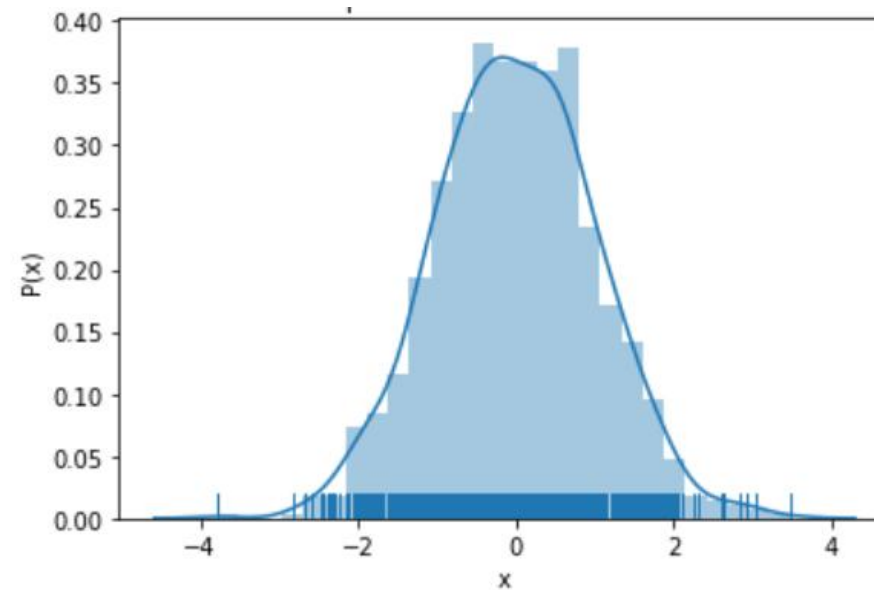


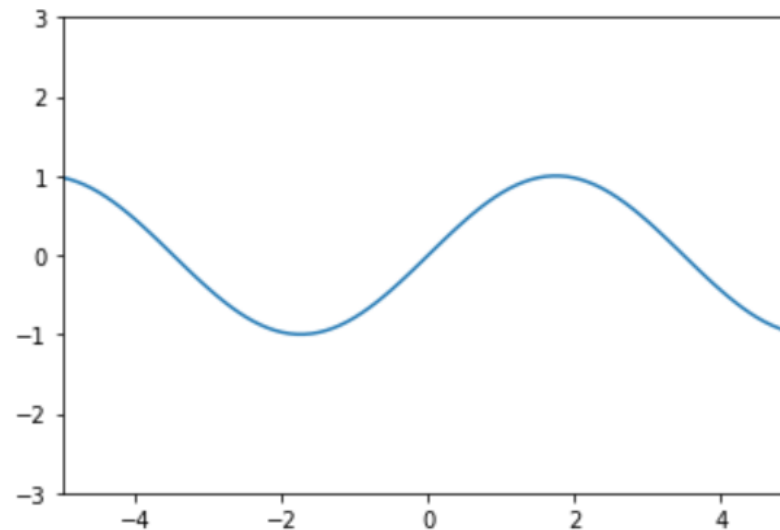
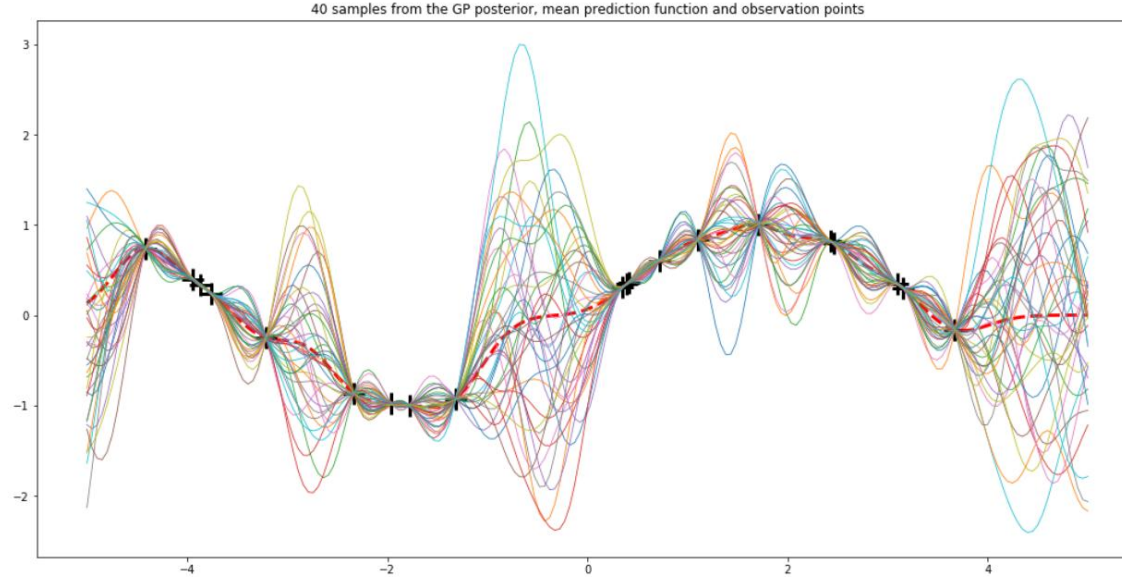
Gaussian Process Regression

A random variable X is said to be normally distributed with mean μ and variance σ^2 if its probability density function (PDF) is

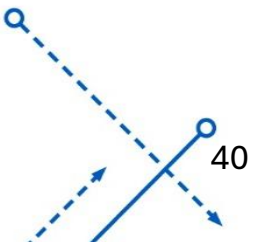
$$P_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

Here, X represents random variables and x is the real argument. The Gaussian or Normal distribution of X is usually represented by $P(x) \sim \mathcal{N}(\mu, \sigma^2)$.

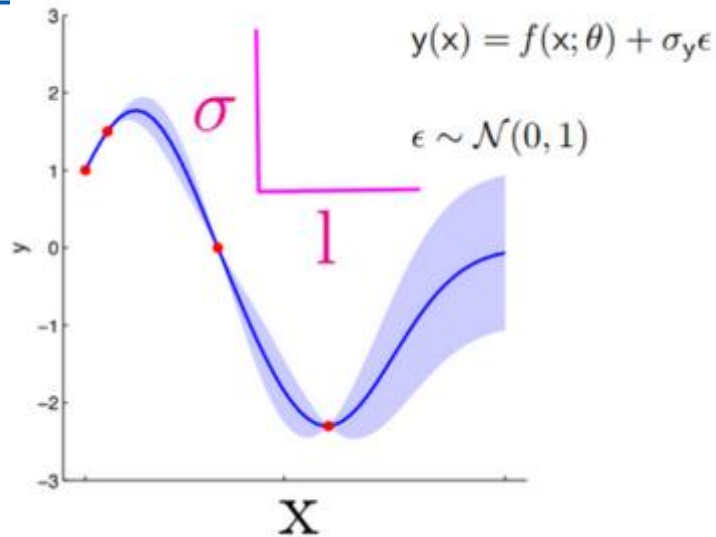




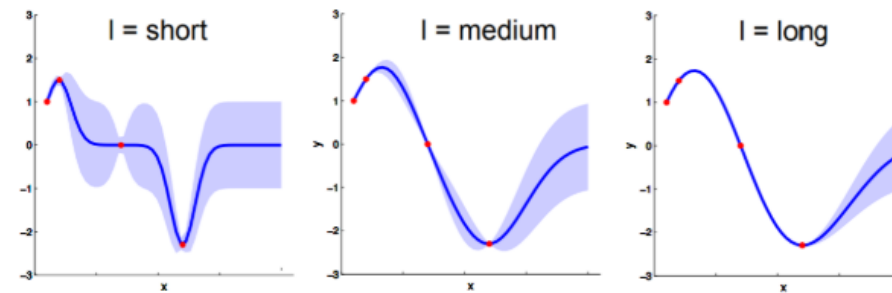
- We can then end up with something like this after the calculations.
- Note, we have a general trend line, which is our model (ie. we would use the below blue line for predicting the values), but it also gives us an idea of the range of values
- That is a big benefit of GPR is that we can get a confidence interval in the analysis.
- Note, the model is accurate around known points but varies in between.



Gaussian Process Regression – Some Parameters



$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)\right)$$



The hyperparameter p (shown instead as l here) is the *length-scale*, and corresponds to the frequency of the functions represented by the Gaussian process prior with respect to the domain. Values of p closer to zero lead the GP to represent high-frequency functions, whereas larger values of p lead to low-frequency functions.

The hyperparameter α (shown here as σ) is the *marginal standard deviation*. It controls the magnitude of the range of the function represented by the GP. If you were to take the standard deviation of many draws from the GP prior at a single input x conditional on one value of α , we would recover α .

- Gaussian process regression is nonparametric (i.e. not limited by a functional form), so rather than calculating the probability distribution of parameters of a specific function, GPR calculates the probability distribution over all admissible functions that fit the data.
- GPR has several benefits, working well on small datasets and having the ability to provide uncertainty measurements on the predictions.
- Unlike many popular supervised machine learning algorithms that learn exact values for every parameter in a function, the Bayesian approach infers a probability distribution over all possible values.

$$p(w|y, X) = \frac{p(y|X, w)p(w)}{p(y|X)}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

$$p(w|y, X) = \frac{p(y|X, w)p(w)}{p(y|X)}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

- How the Bayesian approach works is by specifying a prior distribution, $p(w)$, on the parameter, w , and relocating probabilities based on evidence (i.e. observed data) using Bayes' Rule.
- The updated distribution $p(w|y, X)$, called the posterior distribution, thus incorporates information from both the prior distribution and the dataset.

$$p(f^*|x^*, y, X) = \int_w p(f^*|x^*, w)p(w|y, X)dw$$

- To get predictions at unseen points of interest, x^* , the predictive distribution can be calculated by weighting all possible predictions by their calculated posterior distribution.
- The prior and likelihood is usually assumed to be Gaussian for the integration to be tractable. Using that assumption and solving for the predictive distribution, we get a Gaussian distribution, from which we can obtain a point prediction using its mean and an uncertainty quantification using its variance.

GPR vs. SVR

- SVR offers excellent generalization ability and robustness to outliers. On the other hand, GPR approaches nonlinear regression from a Bayesian perspective, which produces both the optimal estimated function and the covariance associated with the estimation. GPR is a nonparametric model that places a prior on the nonlinear function directly without parametrization.
- Both SVR and GPR are memory-based methods that store a part of or the entire training data for testing. Therefore, they are generally fast to train and are able to improve the efficiency of the massive-training methodology.
- As an oversimplification, SVR defines the kernel and then puts in a margin of variance off of that for prediction, with the accuracy determined by how far off from that margin. GPR instead defines kernels for describing covariance between points, and taking the various probabilities into account.

