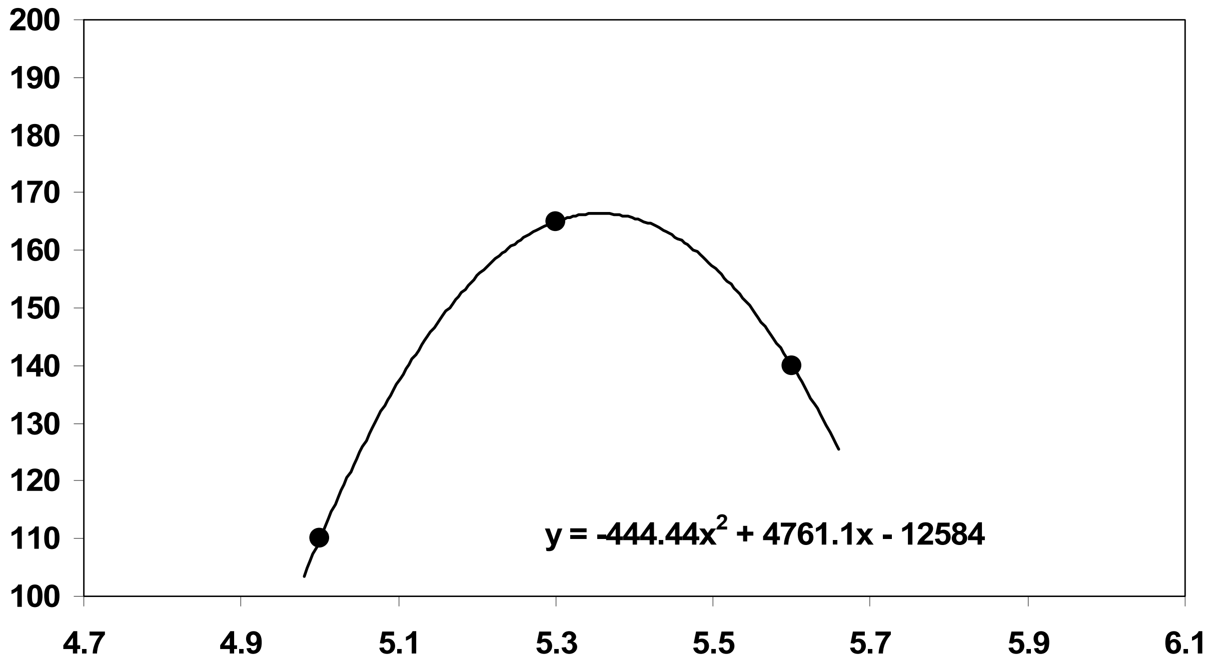
Machine Learning Methods

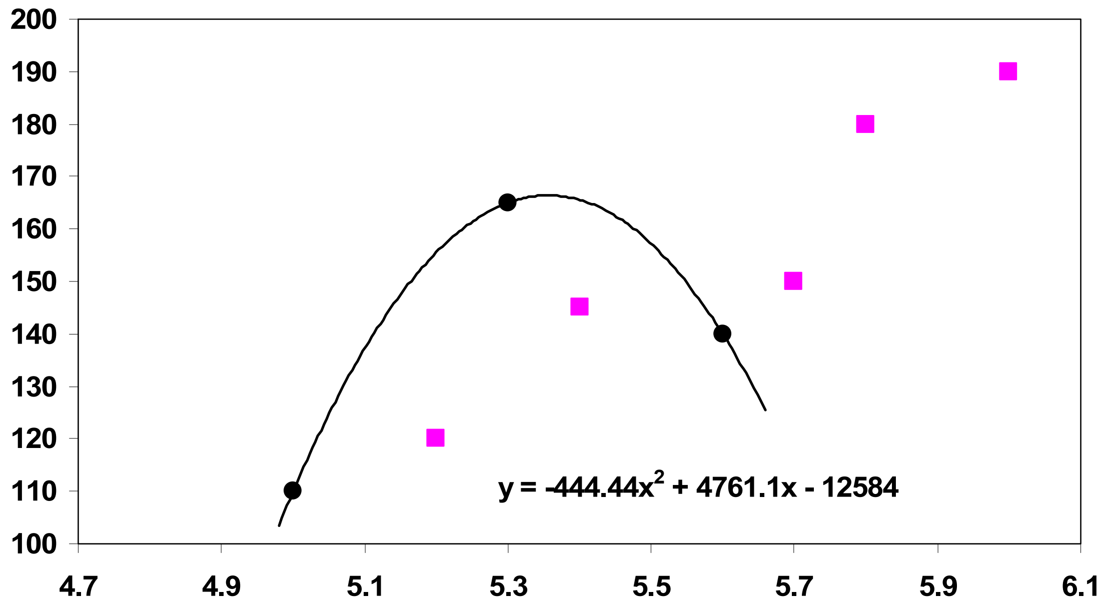
Majid Masso, PhD
Bioinformatics and Computational Biology
George Mason University

Introductory Example

- Attributes X and Y measured for each person (example or instance) in a training set of three individuals
- (X,Y): (4.9,110), (5.3, 160), (5.6, 120)
- Fit a model to the data



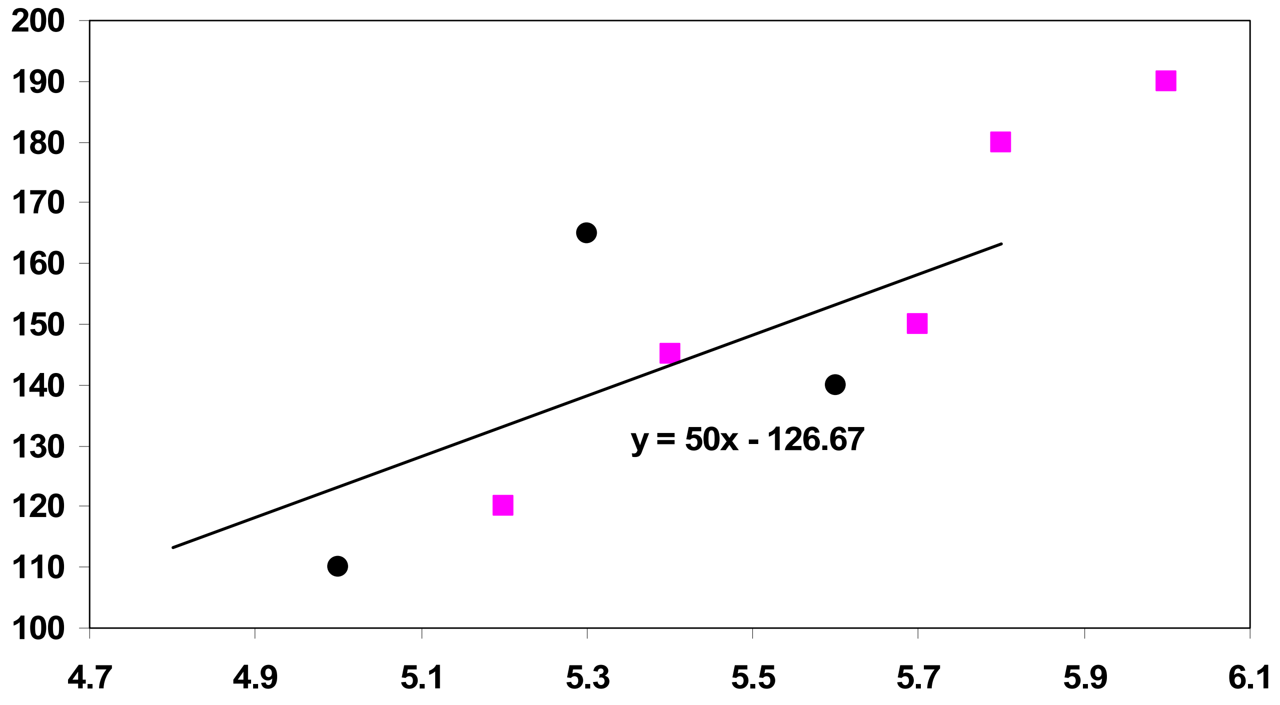
Introductory Example, Continued



- Can the model be used to accurately predict Y attribute of new people, given the value of their X attribute?
- No! By overfitting the model to the training data, we are prevented from using it to make reasonable predictions about new test data.

Introductory Example, Continued

- Perhaps fitting a linear function through the training set instances would allow for better prediction…or, is it too simple (underfitting)? Perhaps a logistic curve?



Example: To Play or Not to Play

- Given weather conditions (outlook, temperature, humidity, windy), should we schedule a game (yes, no)?

- Input attributes: x1 = outlook (sunny, overcast, rainy), x2 = temperature (real #), x3 = humidity (real #), x4 = windy (yes, no); **X** = (x1, x2, x3, x4)

- Output attribute: Y = play game (yes, no)

- Training set: a collection of vectors **V** = (**X**,Y) covering a variety of conditions with known game playing decisions, from which a model can be learned and used to make decisions based on new sets of conditions

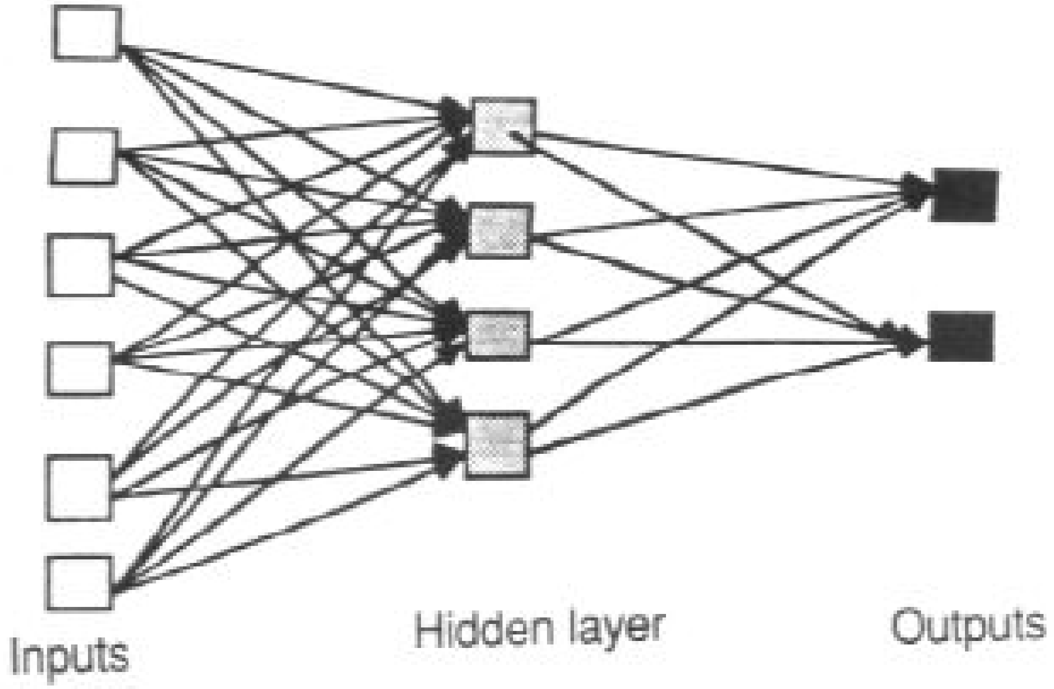
Data Format in Weka (.arff file)

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

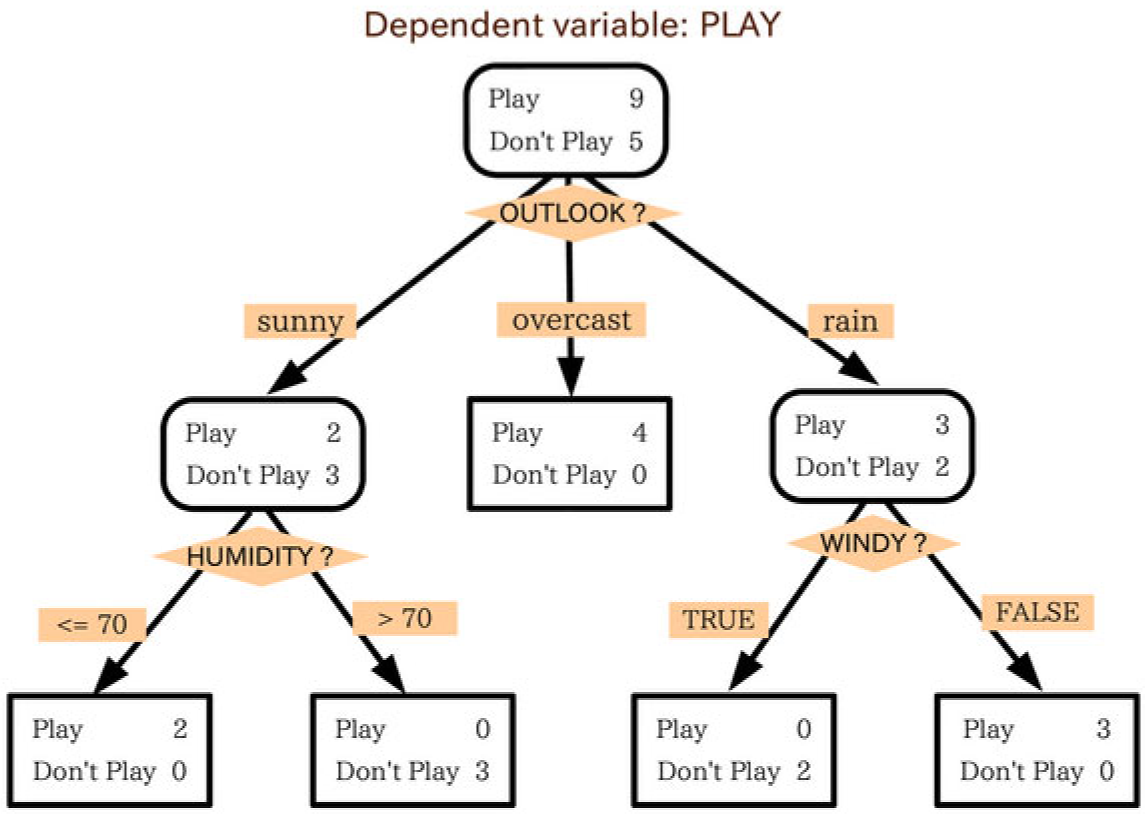
Supervised Classification

- Machine learning algorithms: Neural Network (NN), Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF)
- Training set: collection of instances that an algorithm uses to learn a model; each instance is provided as a feature vector V = (**X**,Y), where **X** = vector of input attributes (independent variables) and Y = the class of the instance (dependent variable, output attribute)
- Common approach among algorithms: learn a model (complex nonlinear function) using the training set that can accurately classify new instances, based on their input attributes
- Learned model: a consistent set of relationships or rules between the attributes of the instances and the class, used for predicting the class memberships of new, unstudied instances

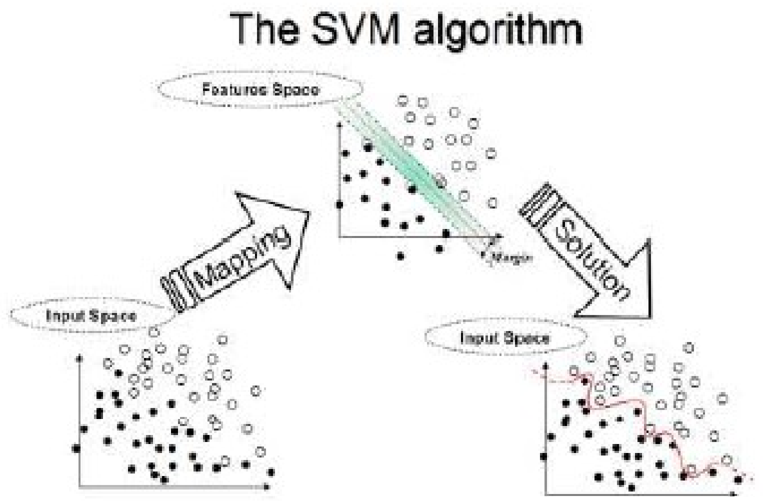
Neural Network

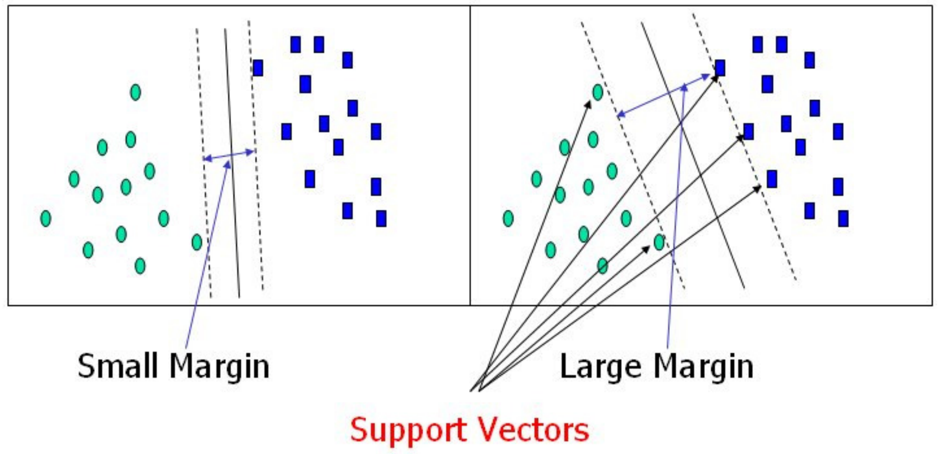


Decision Tree

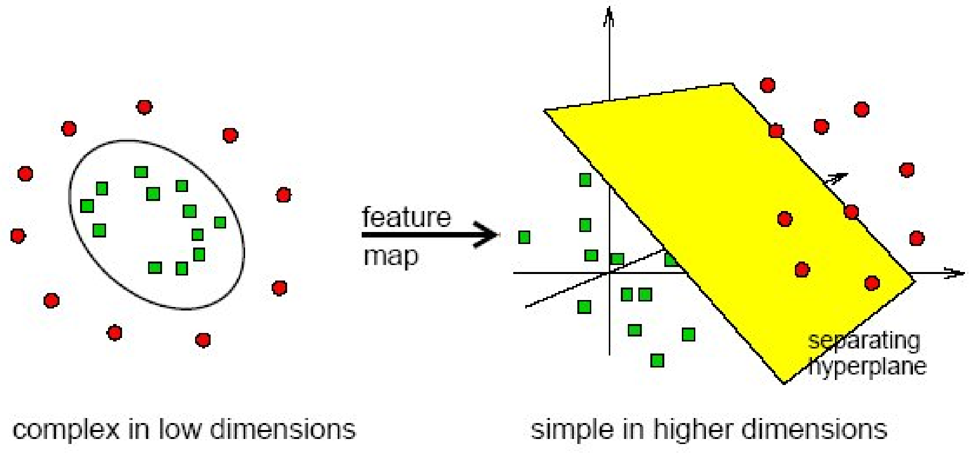


Support Vector Machine





Separation may be easier in higher dimensions



Random Forest

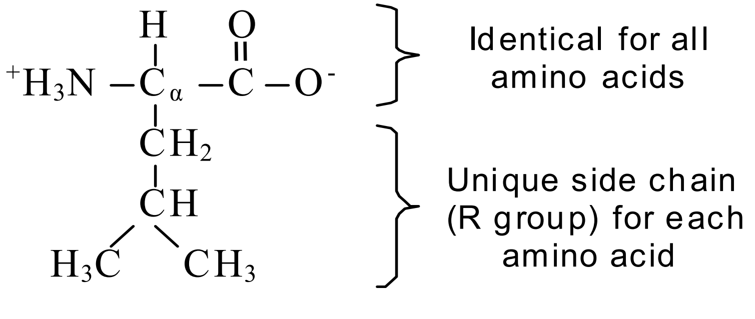
- Let t = total # of trees, n = total # of instances in dataset, and M = total # of input attributes
- For each decision tree, the training set is obtained by selecting n instances *with replacement* (bootstrapping)
- A fixed number m<<M is chosen, and for each node in every tree, the best split on a random subset of m attributes is used to split the node
- No pruning – trees are grown as large as possible
- Classification: majority vote (aggregating) among trees
- Bootstrapping + aggregating = Bagging

Regression

- Suppose output attribute is numerical rather than categorical (such as Y value of introductory example)
- Example: suppose “play game” (yes or no) is replaced with a probability (chance) of playing given weather
- Machine learning algorithms: tree regression, support vector regression
- Models similar to those of supervised classification, except here we predict output values instead of classes

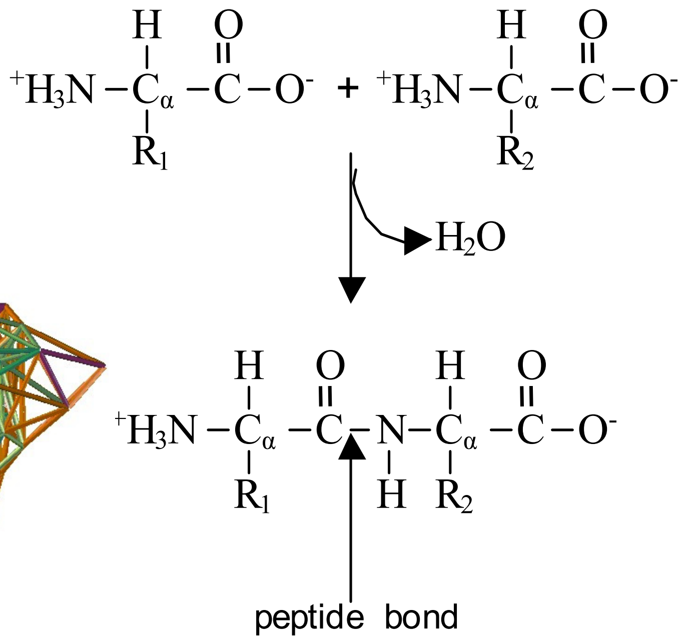
Proteins 101: Crash Course

- Building blocks: amino acids (aa)
  - 20 distinct types in nature (A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y)
  - Can be clustered based on similarities in physico-chemical properties
  - ~200 aa's/protein (widely variable) successively linked by peptide bonds
- Protein structure: primary, secondary, tertiary, quaternary

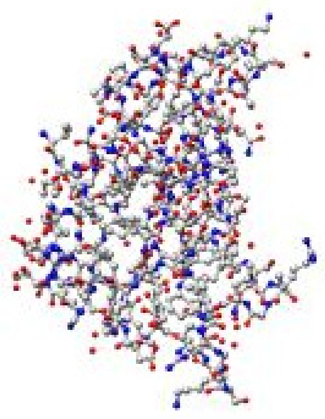
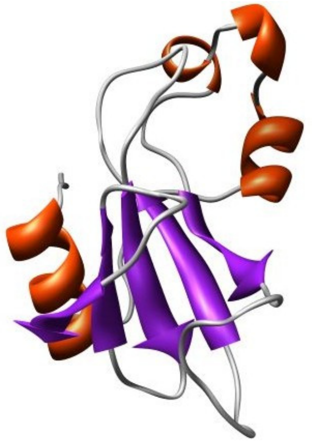
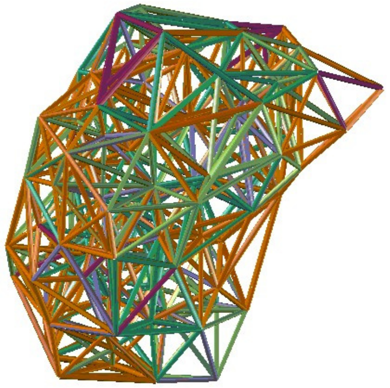


H
O
$^{+}H_3N - C_\alpha - C - O^-$
CH_2
CH
H_3C
CH_3
Identical for all amino acids
Unique side chain (R group) for each amino acid

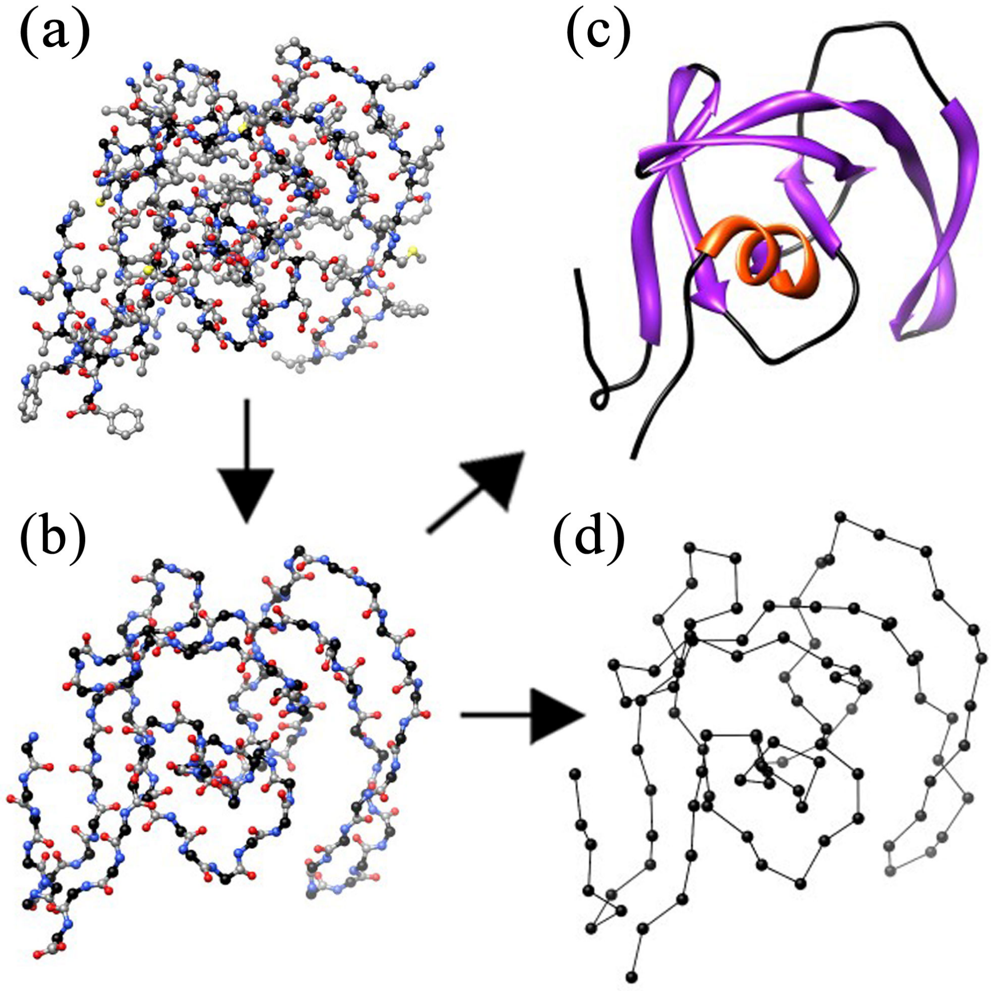
Leucine (Leu or L)



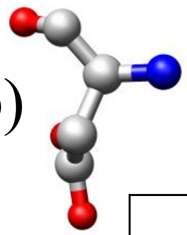
$^{+}H_3N - C_\alpha(H)(R_1) - C(=O) - O^-$ + $^{+}H_3N - C_\alpha(H)(R_2) - C(=O) - O^-$
H_2O
$^{+}H_3N - C_\alpha(H)(R_1) - C(=O) - N(H) - C_\alpha(H)(R_2) - C(=O) - O^-$
peptide bond

Example:HIV-1 Protease (3phv)

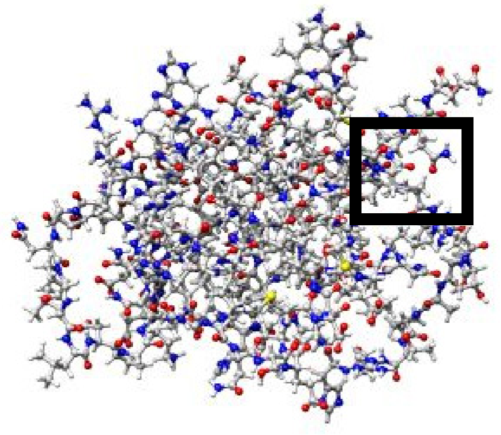


Delaunay Tessellation of Protein Structure

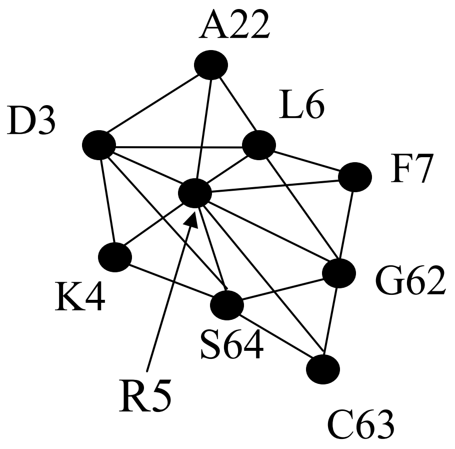
D (Asp) 

$\longrightarrow$ ● C_α or center of mass

Abstract each amino acid to a point
Atomic coordinates – Protein Data Bank (PDB)



$\longrightarrow$



Delaunay tessellation: 3D "tiling" of space into non-overlapping, irregular tetrahedral simplices. Each simplex objectively defines a quadruplet of nearest-neighbor amino acids at its vertices.

Counting Amino Acid Quadruplets

All quadruplets (including permutations): $20^4 = 160{,}000$

Permutation-independent quadruplets (our approach):

$$\underbrace{\mathrm{C}}\ \underbrace{\mathrm{D}}\ \underbrace{\mathrm{E}}\ \underbrace{\mathrm{F}} \qquad \binom{20}{4}$$

$$\mathrm{C}\ \mathrm{C}\ \underbrace{\mathrm{D}}\ \underbrace{\mathrm{E}} \qquad 20\cdot\binom{19}{2}$$

$$\underbrace{\mathrm{C}\ \mathrm{C}}\ \underbrace{\mathrm{D}\ \mathrm{D}} \qquad \binom{20}{2}$$

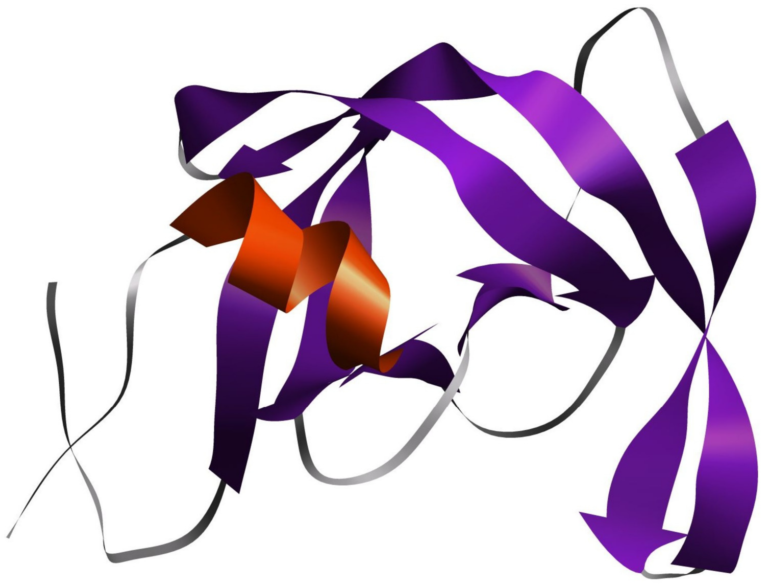
$$\mathrm{C}\ \mathrm{C}\ \mathrm{C}\ \mathrm{D} \qquad 20\cdot 19$$

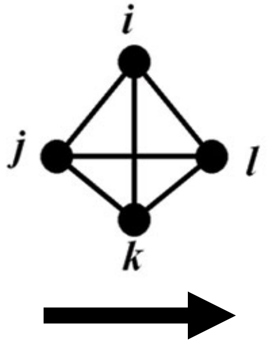
$$\mathrm{C}\ \mathrm{C}\ \mathrm{C}\ \mathrm{C} \qquad 20$$

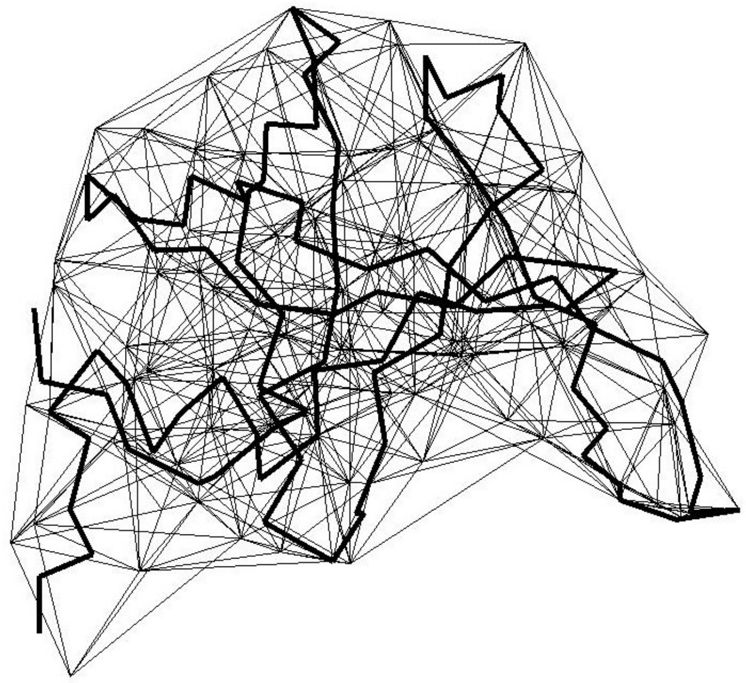
Total: 8,855 distinct quadruplets (no permutations)

Example: HIV-1 Protease (3phv)

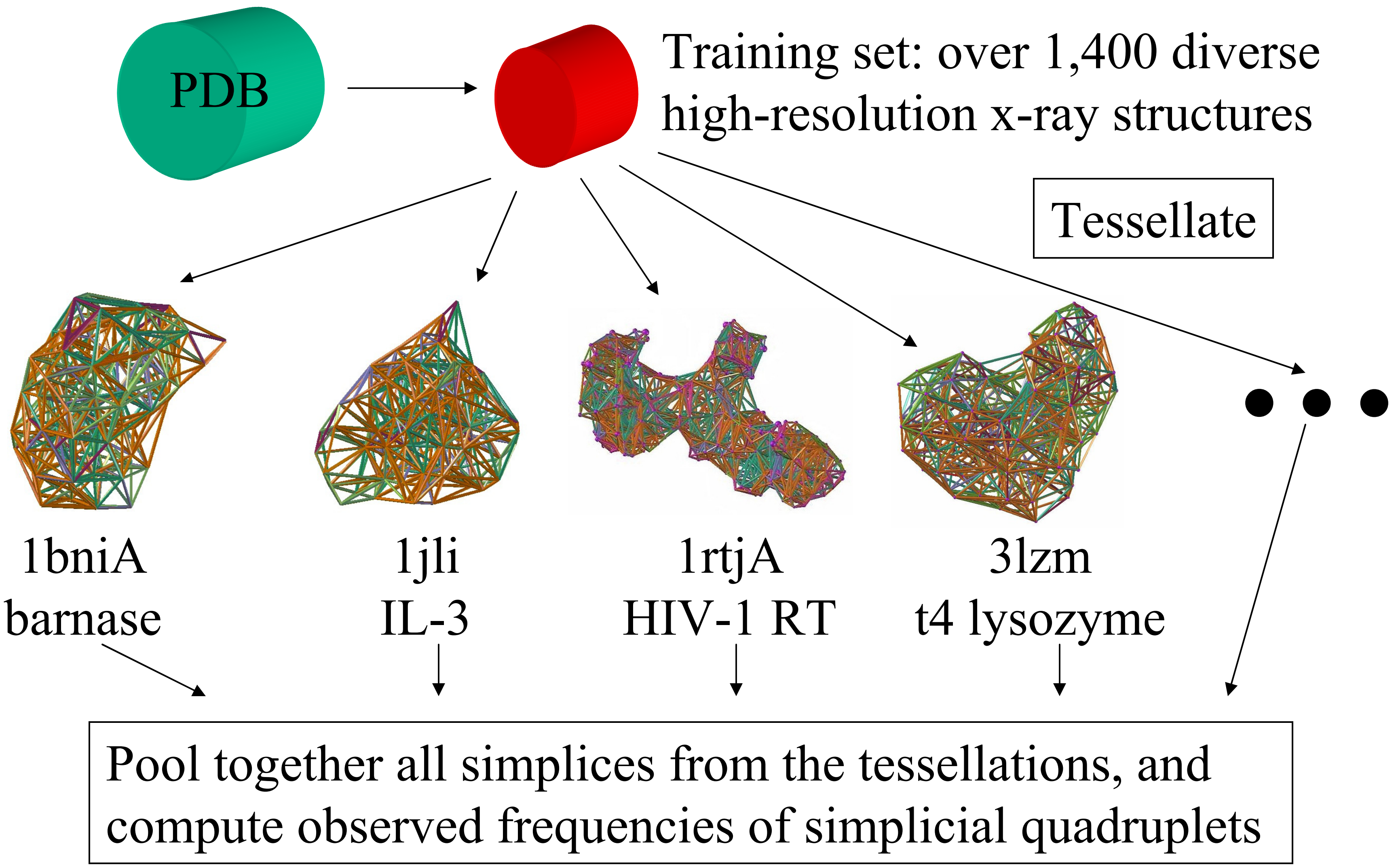
Vertices: Weighted side chain center of mass (CM) points for 99 aa's
Dark line: C-alpha backbone trace (coincides with a vertex for Gly)
Left: ribbon diagram; Right: tessellation (12A filter), "true" neighbors







Four-Body Statistical Potential



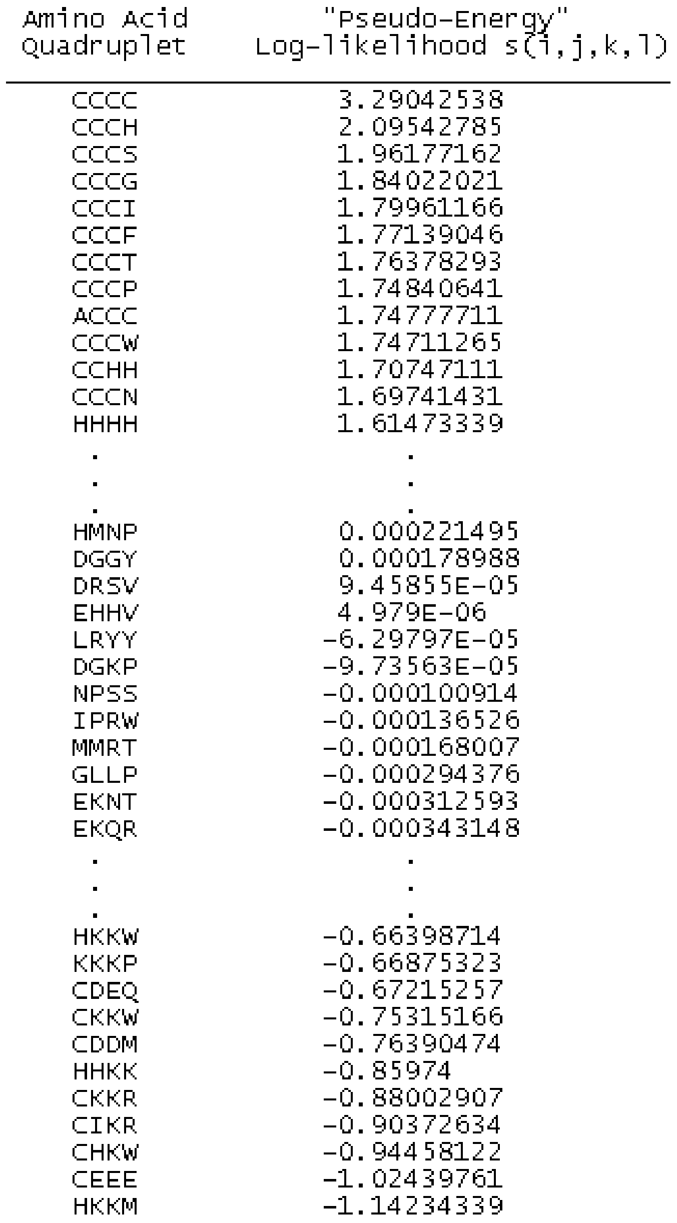
Training set: over 1,400 diverse high-resolution x-ray structures

Pool together all simplices from the tessellations, and compute observed frequencies of simplicial quadruplets

Four-Body Statistical Potential

- Knowledge-based, modeled after inverse Boltzmann principle
- For amino acid quadruplet (i,j,k,l), a log-likelihood score (energy of interaction) is given by $s(i,j,k,l) = \log(f_{ijkl} / p_{ijkl})$
- f_{ijkl} = observed proportion of tetrahedra in the training set tessellations whose four vertex residues are i,j,k,l
- p_{ijkl} = rate expected by chance (multinomial distribution, based on the proportion of all residues in the training set proteins that are of the types i,j,k,l)
- Four-body statistical potential: the collection of 8855 quadruplet types and their respective log-likelihood scores

Four-Body Statistical Potential

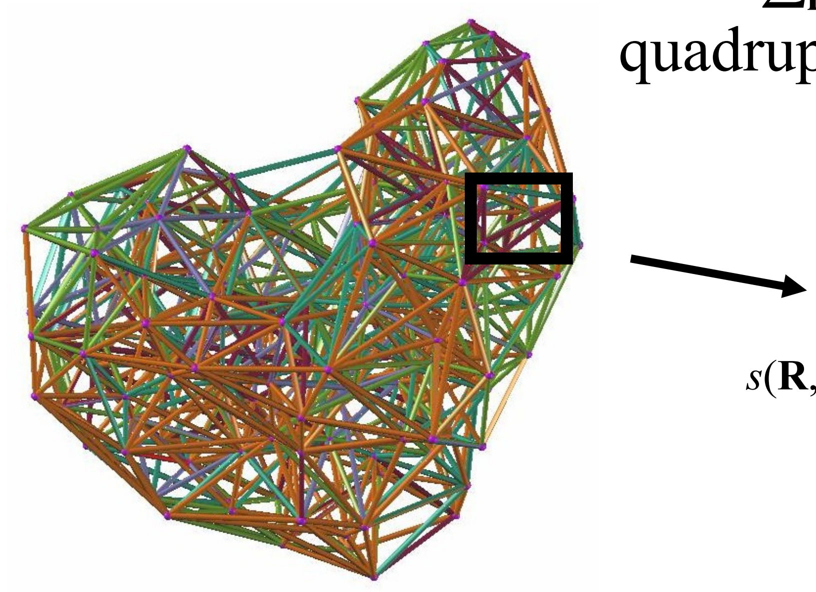


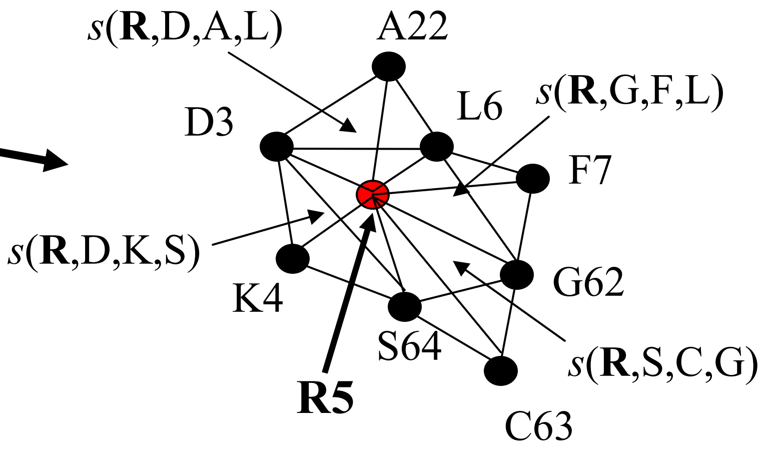
Amino Acid Quadruplet	"Pseudo-Energy" Log-likelihood s(i,j,k,l)
CCCC	3.29042538
CCCH	2.09542785
CCCS	1.96177162
CCCG	1.84022021
CCCI	1.79961166
CCCF	1.77139046
CCCT	1.76378293
CCCP	1.74840641
ACCC	1.74777711
CCCW	1.74711265
CCHH	1.70747111
CCCN	1.69741431
HHHH	1.61473339
.	.
.	.
.	.
HMNP	0.000221495
DGGY	0.000178988
DRSV	9.45855E-05
EHHV	4.979E-06
LRYY	-6.29797E-05
DGKP	-9.73563E-05
NPSS	-0.000100914
IPRW	-0.000136526
MMRT	-0.000168007
GLLP	-0.000294376
EKNT	-0.000312593
EKQR	-0.000343148
.	.
.	.
.	.
HKKW	-0.66398714
KKKP	-0.66875323
CDEQ	-0.67215257
CKKW	-0.75315166
CDDM	-0.76390474
HHKK	-0.85974
CKKR	-0.88002907
CIKR	-0.90372634
CHKW	-0.94458122
CEEE	-1.02439761
HKKM	-1.14234339

Application 1: Protein Total Potential (TP)

- Obtained by summing the log-likelihood scores of **all** simplicial quadruplets defined by the protein tessellation
- Global measure of protein sequence-structure compatibility

TP = $\sum_{\hat{\imath}} s(\hat{\imath})$, sum taken over **all** simplex quadruplets $\hat{\imath}$ in the entire tessellation.

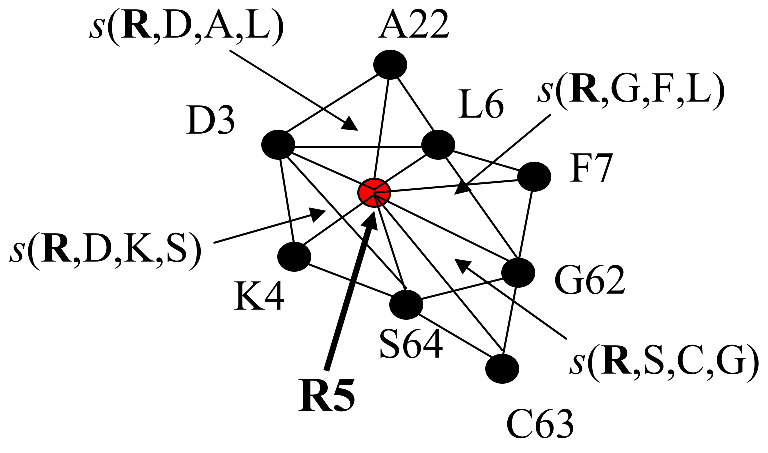




Close-up view of **only** the four simplices that use **R** at position **5** as a vertex

Application 2: Residue Environment Scores

- For each amino acid position, **locally** sum the scores $s(i,j,k,l)$ of **only** tetrahedra that use the position as a vertex

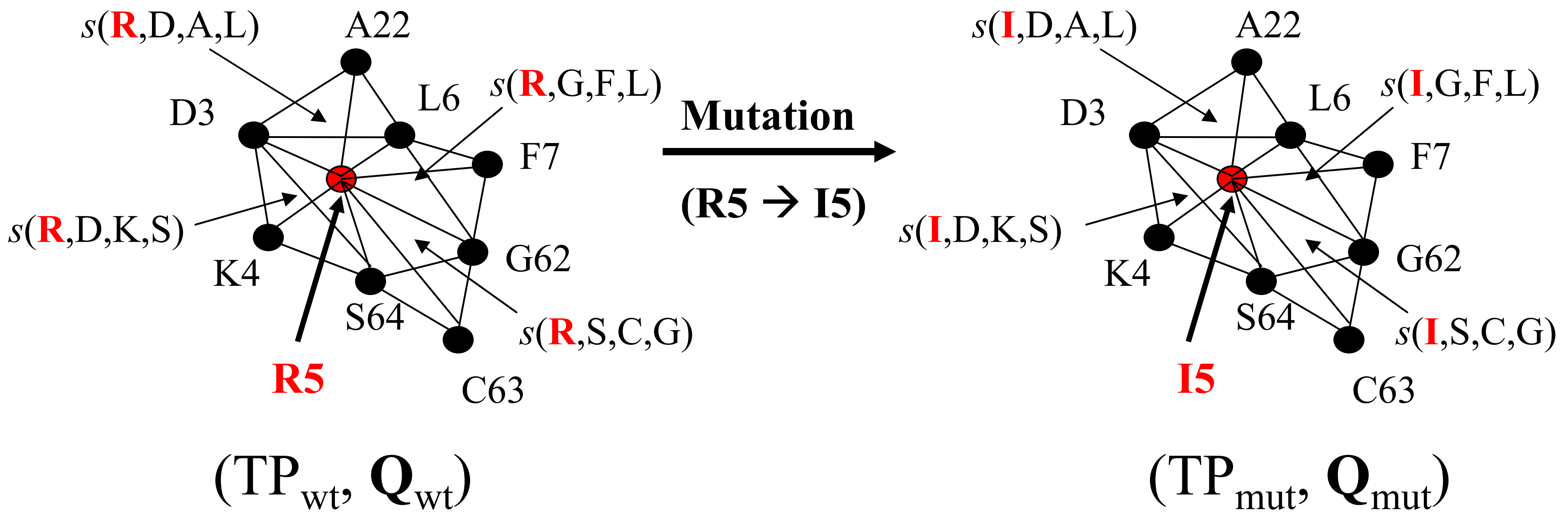


Example: $q_5 = q(\mathrm{R5}) = \sum_{(i,j,k,l)} s(i,j,k,l)$, sum is taken **only** over all quadruplets (i,j,k,l) that use R5

- The scores of all the amino acid positions in the protein structure form a **Potential Profile** vector $\mathbf{Q} = < q_1, q_2, q_3, \ldots, q_N >$
  (N = length of primary sequence in the solved structure)

Computational Mutagenesis Methodology

- Observations:
  - Few solved mutant structures to compare with solved wild type (wt) structure
  - Mutant and wt protein structure tessellations are very similar or identical
- Approach:
  - Obtain total potental (TP_{mut}) and potential profile vector ($\mathbf{Q}_{mut}$) for any single residue mutant by using the wt structure tessellation as a template
  - Simply change the residue label at a given point and re-compute



Computational Mutagenesis Methodology

- **Scalar "Residual Score"** of a mutant:

  (mutant – wt) total potential difference = $TP_{mut} - TP_{wt}$ (relative change in sequence-structure compatibility upon mutation)

- **Vector "Residual Profile"** of a mutant:

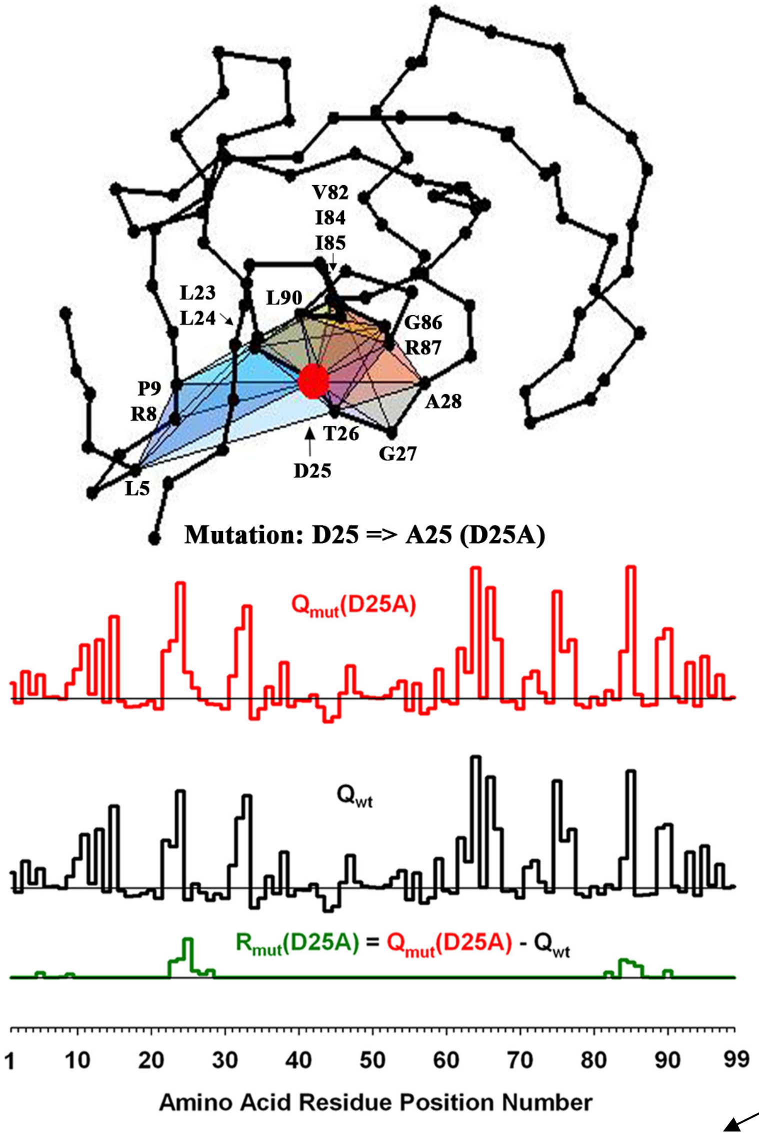
  $\mathbf{R} = \mathbf{Q}_{mut} - \mathbf{Q}_{wt}$ = (mutant – wt) potential profile vector difference (environmental perturbation score for every position in structure)

- Denote $\mathbf{R} = < EP_1, EP_2, EP_3, \ldots, EP_N >$

  $EP_i = q_{i,mut} - q_{i,wt}$ = environmental perturbation at position i

- Geometric property: mutation at position i => EP_i = residual score

Example: HIV-1 Protease (PR)



(A) Mutant 3D-1D

(B) Native 3D-1D

Residual Profile **R**
(A – B)

<u>Note:</u> "EC" is now "EP"

EC components of **R**

Residual Score: EC_{25}
(mutation "epicenter")

EC_5 = 0.43, EC_8 = 0.01, EC_9 = 0.31, EC_{23} = 1.63, EC_{24} = 1.87, EC_{25} = 3.83, EC_{26} = 0.67, EC_{27} = 0.34, EC_{28} = 0.67, EC_{82} = 0.49, EC_{84} = 1.76, EC_{85} = 1.52, EC_{86} = 1.14, EC_{87} = 4.60×10^{-5}, EC_{90} = 0.63, EC_i = 0 for all other positions *i*

HIV-1 Protease Dataset Example: Residual Profiles for 536 Experimental Mutants

WT	POSITION	MUTANT	P1	Q2	I3	T4	L5	W6	Q7	R8	P9	L10	V11	T12		N98	F99	ACTIVITY
PRO	1	HIS	1.89369	0.12473	0.2462	-0.01137	0	0	0	0	0	0	0	0.15478		0.2482	0	pos
PRO	1	LEU	1.61399	-0.21225	1.51021	0.14456	0	0	0	0	0	0	0	-0.05708		-0.7566	0	pos
PRO	1	SER	0.80073	0.19565	0.14197	0.15969	0	0	0	0	0	0	0	0.1124		0.30934	0	int
GLN	2	GLU	-0.6395	-1.55273	-0.24116	-1.33969	-0.4477	-0.41718	0	0	0	0	0	-0.47309		-0.29306	-0.31513	pos
ILE	3	ASN	-0.32949	0.76726	-2.46203	0.5757	-1.49592	0	0.31665	0	-0.93573	-0.49091	-1.47315	0		0.46809	0	pos
ILE	3	LEU	0.35974	0.41178	1.5984	0.10011	0.37716	0	0.2498	0	0.42616	0.2479	0.19533	0		0.50297	0	pos
ILE	3	SER	0.35207	0.88747	-1.14271	0.53599	-1.30293	0	0.40746	0	-0.52978	-0.29686	-1.07501	0		0.38893	0	neg
ILE	3	THR	0.28471	0.89302	-0.3196	0.72597	-1.06583	0	0.60907	0	-0.17343	-0.1048	-0.43737	0		0.29873	0	int
THR	4	ARG	-0.36146	-0.33689	-0.18267	-0.34217	-0.43148	0.00263	0.25453	0	-0.16441	0	0	0.03462		-0.18464	-0.18971	int
THR	4	SER	0.03021	-0.26497	-0.21622	-0.33293	-0.23951	0.0838	-0.11714	0	-0.11618	0	0	-0.06209		-0.08467	0.06375	pos
LEU	5	HIS	0	0.06901	-1.55951	0.05785	-0.9789	0.1661	0.55983	0.86038	0.44361	0	0	0		-0.09357	-0.48623	neg
LEU	5	VAL	0	0.00037	-0.2512	0.07167	-0.33375	-0.05122	-0.07882	-0.14561	-0.02276	0	0	0		0.09464	-0.01646	neg
TRP	6	CYS	0	-0.24419	0	-0.521	-0.58979	-1.12732	-0.66335	-0.45596	0	0	0	0		0	-0.26395	pos
TRP	6	GLY	0	-0.18178	0	-0.63535	-0.90704	-1.28979	-0.33159	-0.17572	0	0	0	0		0	-0.62764	pos
TRP	6	LEU	0	-0.03694	0	-0.00334	0.26617	0.26431	-0.04368	0.14435	0	0	0	0	…	0	0.08937	pos
GLN	7	HIS	0	0	0.22456	0.14707	-0.05542	0.16744	0.24723	-0.08248	-0.0548	0.17104	0.14183	0.02147		0	0	pos
GLN	7	LEU	0	0	1.13621	0.28754	0.24948	0.54479	1.00782	-0.41464	0.37055	1.21177	0.94688	-0.13142		0	0	neg
GLN	7	PRO	0	0	0.20172	-0.12112	0.03098	-0.03136	0.00232	0.20147	0.33796	0.19486	0.06676	-0.14616		0	0	neg
ARG	8	ASN	0	0	0	0	-0.38913	0.18631	-0.63722	-2.26973	-0.61127	-0.75384	0	0		0	0	neg
ARG	8	ASP	0	0	0	0	-0.94424	-0.29427	-1.15565	-4.07861	-0.73567	-1.05439	0	0		0	0	neg
ARG	8	GLN	0	0	0	0	0.02021	0.48854	0.52975	-0.80067	0.15343	-0.06552	0	0		0	0	int
ARG	8	GLU	0	0	0	0	-0.95011	-0.35115	-0.5433	-3.12437	-0.62964	-0.65032	0	0		0	0	neg
ARG	8	GLY	0	0	0	0	-0.42784	6.00E-05	-1.3967	-3.00439	-0.60337	-0.61053	0	0		0	0	neg
ARG	8	HIS	0	0	0	0	0.18617	0.41218	-0.14344	-0.53493	0.01364	-0.13521	0	0		0	0	neg
ARG	8	LEU	0	0	0	0	0.69068	0.95149	-0.60797	0.0926	0.18717	0.90623	0	0		0	0	neg
ARG	8	LYS	0	0	0	0	-0.61972	-0.26158	-0.45997	-1.35066	-0.56148	-0.48045	0	0		0	0	int
ARG	8	TYR	0	0	0	0	0.46293	0.69359	-0.68478	-0.51269	0.08071	0.13992	0	0		0	0	neg
PRO	9	ARG	0	0	-0.53754	-0.11854	0.08246	0	0.06947	0.34747	0.05305	-0.37048	-0.40188	0		0	0	neg
PRO	9	HIS	0	0	-0.03502	0.01097	0.29562	0	0.07942	0.04235	0.37048	-0.05895	-0.01009	0		0	0	neg

⋮ ⋮

Data Format in Weka (.arff file)

```
@relation 536profiles

@attribute wt {ALA, CYS, ASP, GLU, PHE, GLY, HIS, ILE, LYS, LEU, MET, ASN, PRO, GLN, ARG,
SER, THR, VAL, TRP, TYR}
@attribute position real
@attribute sub {ALA, CYS, ASP, GLU, PHE, GLY, HIS, ILE, LYS, LEU, MET, ASN, PRO, GLN, ARG,
SER, THR, VAL, TRP, TYR}
@attribute P1 real
@attribute Q2 real
@attribute I3 real
@attribute T4 real
```

• • •

```
@attribute L97 real
@attribute N98 real
@attribute F99 real
@attribute activity {active, inactive}

@data
PRO,1,HIS,1.89369,0.12473,0.2462,-
0.01137,0,0,0,0,0,0,0,0.15478,0,0,0,0,0,0,0.03253,0,0,0,0,0.17239,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.87811,0.19179,1.11231,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.08714,0.38016,0.97826,0.08584,0,0.2482,0,active
PRO,1,LEU,1.61399,-0.21225,1.51021,0.14456,0,0,0,0,0,0,0,-
0.05708,0,0,0,0,0,0,0.03691,0,0,0,0,2.13064,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.28094,-0.1156,-
0.60914,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1.23294,-0.46368,1.06605,-
0.34583,0,-0.7566,0,active
PRO,1,SER,0.80073,0.19565,0.14197,0.15969,0,0,0,0,0,0,0,0.1124,0,0,0,0,0,0,-
0.09538,0,0,0,0,0.02238,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0.34489,-
0.005,0.15276,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.22211,0.16134,0.53626,0.1438
6,0,0.30934,0,active
```

• • •

Evaluating Algorithm Performance

- Overall goal: Develop model with known examples to accurately predict class (or value) of instances that have not yet been assayed experimentally (potentially great savings of time and money)
- Ideal situation: split large original dataset into 3 subsets
  - Training set (learn model)
  - Validation set (optimize model by tweaking model parameters)
  - Test set (evaluate model on new data not used to develop model)
  - Errors measured at each step (resubstitution, validation, generalization)
- Approaches: Tenfold cross-validation (10-fold CV);

  leave-one-out CV (i.e., N-fold CV, where N = dataset size);

  % split (e.g., use only 2/3 for training, 1/3 held out for testing)

Evaluating Algorithm Performance

- 10-fold CV
  - Randomly split the dataset instances into 10 equally-sized subsets
  - Hold-out subset 1; combine subsets 2-10 into one training set for learning a model; use trained model to predict classes of instances in subset 1
  - Repeat previous step 9 more times (e.g., hold-out subset 2, combine subsets 1 and 3-10 together to train a model, use model to predict subset 2, etc)
  - We end up with 10 models, each trained using 90% of the original dataset, and each used to predict the held-out 10% subset.
  - In the end, each instance has one class prediction – compare to actual class
- LOOCV (leave-one-out CV, or N-fold CV)
  - Similar to above, but each subset contains only 1 instance
  - Deterministic – no randomness to which instances are grouped as subsets
  - Overall prediction accuracy provides rough idea of how a model trained with the full dataset will perform
- % split (self-explanatory)

Evaluating Algorithm Performance

- Assume instances belong to two generic classes (Pos/Neg)
- Results of comparing predictions with actual classes based on the approaches described (10-fold CV, LOOCV, % split) can be summarized in a confusion matrix:

		Predicted as	
		Pos	Neg
Actual class	Pos	TP	FN
	Neg	FP	TN

- Classification performance measures:

accuracy = (TP+TN) / (TP+FP+TN+FN); sensitivity = TP / (TP+FN);

specificity = TN / (TN+FP); precision (PPV) = TP / (TP+FP);

BAR = $0.5 \times$ [sensitivity + specificity];

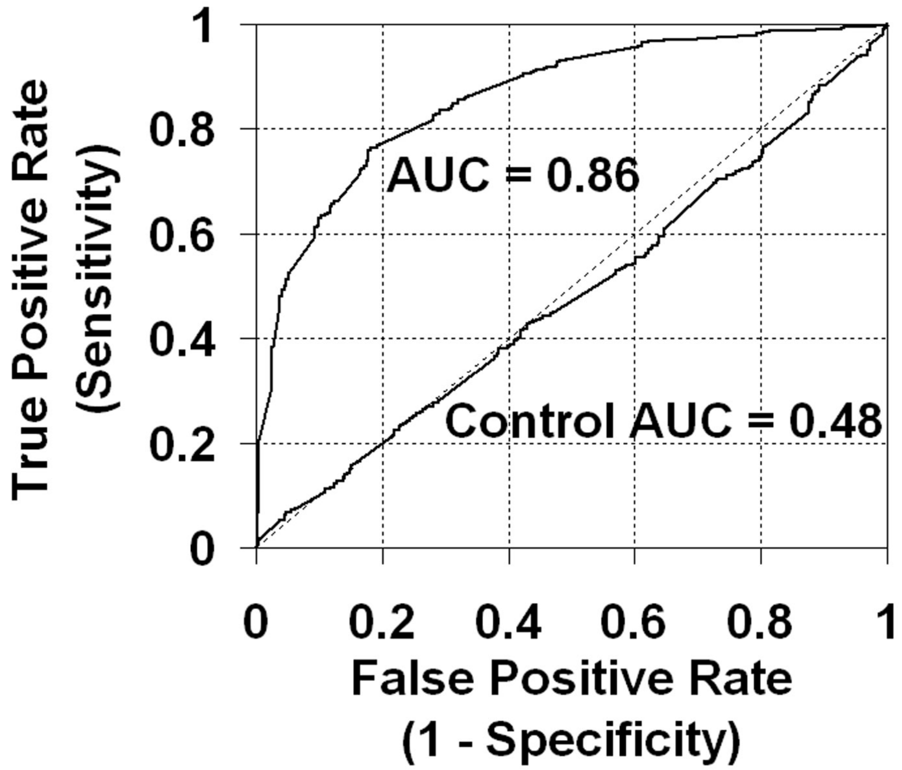
MCC = $(TP \times TN - FP \times FN) / [(TP+FN)(TP+FP)(TN+FN)(TN+FP)]^{1/2}$;

AUC = area under ROC curve (plot of sensitivity vs. 1 – specificity)

ROC Curve

- Plot of true positive rate (sensitivity) versus false positive rate (1 – specificity) in the unit square
- AUC = probability that classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one
- AUC ~ 0.5 (ROC close to diagonal line joining points (0,0) and (1,1)) suggests no signal in dataset and that trained model is not likely to perform any better than random guessing
- AUC = 1 (piecewise linear ROC joining (0,0) to (0,1) and (0,1) to (1,1)) indicates a perfect classifier

ROC Curve



True Positive Rate (Sensitivity)
False Positive Rate (1 - Specificity)
0
0.2
0.4
0.6
0.8
1
AUC = 0.86
Control AUC = 0.48

10-Fold CV Weka Output Example

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -I 100 -K 0 -S 1
Relation:     536profiles
Instances:    536
Attributes:   103
              [list of attributes omitted]
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Random forest of 100 trees, each constructed while considering 7 random features.
Out of bag error: 0.1866
```

10-Fold CV Weka Output Example, Continued

```
Time taken to build model: 17.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         443               82.6493 %
Incorrectly Classified Instances        93               17.3507 %
Kappa statistic                          0.6418
Mean absolute error                      0.2963
Root mean squared error                  0.3625
Relative absolute error                 60.8836 %
Root relative squared error             73.5037 %
Total Number of Instances              536

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.777     0.138      0.802     0.777     0.789      0.894    active
  0.862     0.223      0.843     0.862     0.853      0.894    inactive

=== Confusion Matrix ===

   a   b   <-- classified as
 174  50 |   a = active
  43 269 |   b = inactive
```