# Lecture 19
# Elements of Machine Learning

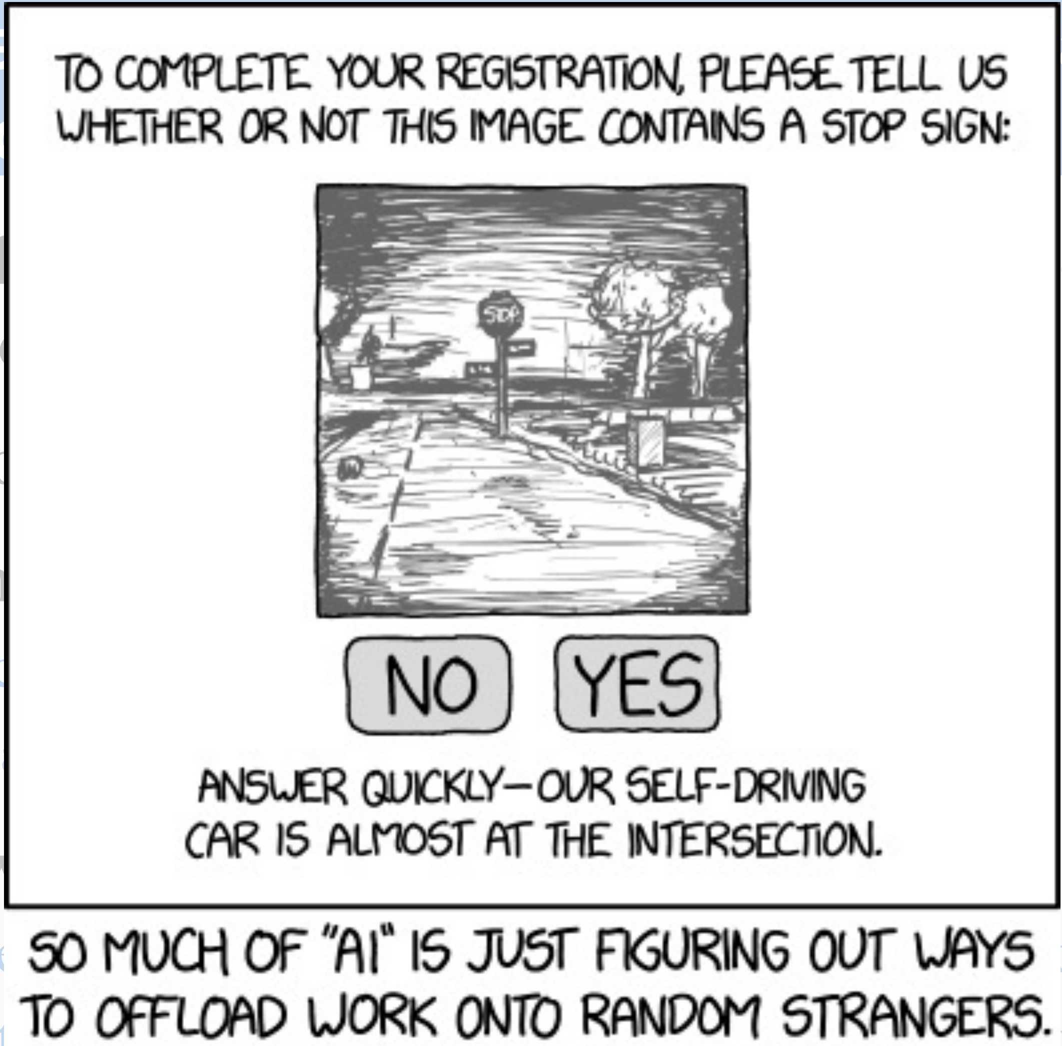
Basic terminology, concepts, and methods; linear models

*Prof. David A. Kofke*
*CE 500 – Modeling Potential-Energy Surfaces*
*Department of Chemical & Biological Engineering*
*University at Buffalo*

© 2024 David Kofke

# Machine … algorithms that aut… data

- New skill… d through observati…
- Most algo… s
- Core elen…
  - Statistic…
  - Artifici…
- What new…
  - The inte… s of data
  - Motivat…


TO COMPLETE YOUR REGISTRATION, PLEASE TELL US WHETHER OR NOT THIS IMAGE CONTAINS A STOP SIGN:
STOP
NO
YES
ANSWER QUICKLY—OUR SELF-DRIVING CAR IS ALMOST AT THE INTERSECTION.
SO MUCH OF "AI" IS JUST FIGURING OUT WAYS TO OFFLOAD WORK ONTO RANDOM STRANGERS.

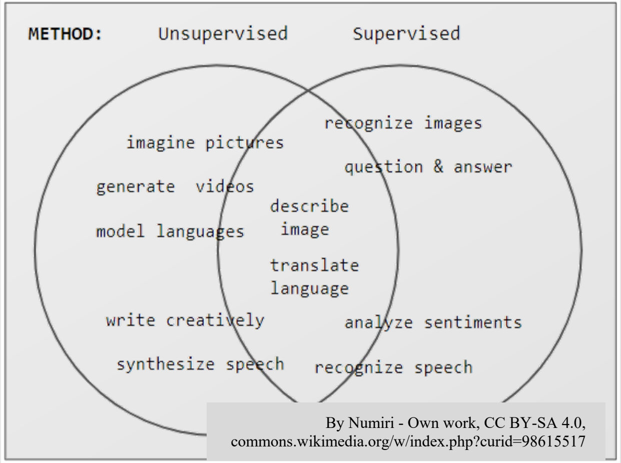
Randall Munroe
xkcd.com/1897/

# *Machine learning* (ML) is the study of algorithms that auto-improve via experience and data

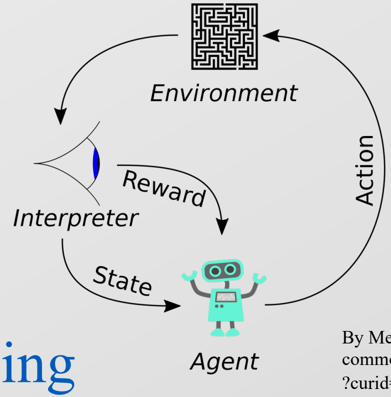
- New skills and/or better performance is acquired through observation and trial & error
- Most algorithms are based in statistical concepts
- Core elements have been around for decades
  - Statistical inference methods
  - Artificial intelligence
- What new?  Data!
  - The internet and social media produces huge amounts of data
  - Motivates development of algorithms to harness for useful purposes

# ML approaches are classified into supervised, unsupervised, and reinforcement learning

- Supervised
  - {input, output} data pairs are provided, and goal is to provide correct output values for new input data
- Unsupervised
  - No output labels are provided with data; rather algorithm seeks to find patterns in it
- Reinforcement
  - Agent explores actions guided by rewards
  - E.g., games, robot control, autonomous driving



By Numiri - Own work, CC BY-SA 4.0, commons.wikimedia.org/w/index.php?curid=98615517

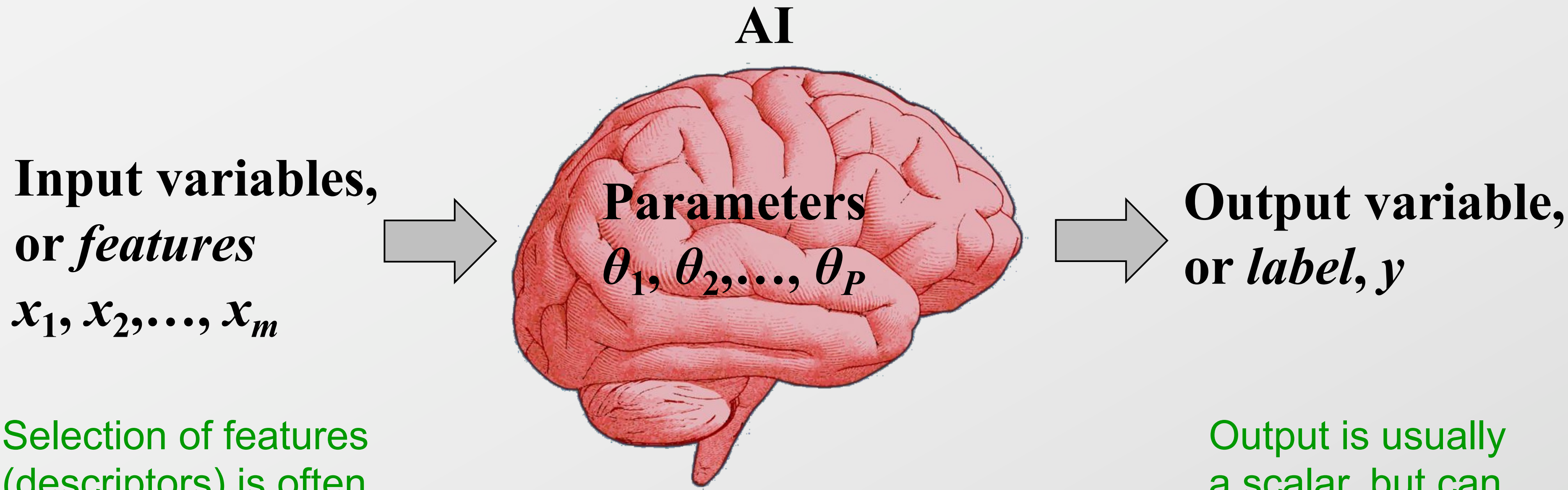


By Megajuice - Own work, CC0, commons.wikimedia.org/w/index.php ?curid=57895741

# Supervised learning separates into *classification* and *regression*, depending on the type of output

- Classification aims to identify the discrete category for new input data
  - E.g., whether a new molecule is toxic vs. non-toxic
- Regression aims to estimate the value of a continuous variable given new input data
  - E.g., $pK_a$, atomization energy, redox potential for a new molecule
  - Potential energy for a configuration of molecules

# The operation of a supervised ML algorithm is governed by a set of numeric parameters



**AI**

**Input variables, or *features***
$x_1, x_2, \ldots, x_m$

**Parameters**
$\theta_1, \theta_2, \ldots, \theta_P$

**Output variable, or *label*, $y$**

Selection of features (descriptors) is often an art that can be crucial to ML success

There may be many, many parameters

Output is usually a scalar, but can be multivariate

# *Training* is the process of establishing parameter values, by minimizing a *cost function*

- The training set is a collection of (***x***, *y*) pairs

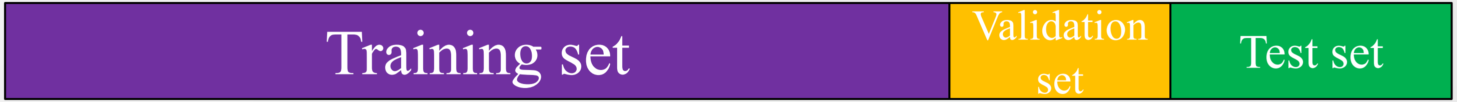
- A cost function (aka *loss*, *error*) characterizes the error in the ML estimate of the output values relative to the given ones
  - L1 norm: $\sum_{i=1}^{n} \left| \hat{y}(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right| = ||\hat{\mathbf{y}} - \mathbf{y}||_1$
  - L2 norm: $\sum_{i=1}^{n} \left( \hat{y}(\mathbf{x}^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right)^2 \equiv ||\hat{\mathbf{y}} - \mathbf{y}||_2^2$

- The training process attempts to minimize the cost function through manipulation of the parameters ***θ***

# Some data processing may be performed before starting the training and application

- Cleaning
  - *e.g.*, filling in missing values
- Standardization or other transformation
  - *e.g.*, scaling to zero mean and unit standard deviation
  - Often this makes fitting more generic and easier, without irreversibly changing the data

# Available data should be split into training, validation, and test sets



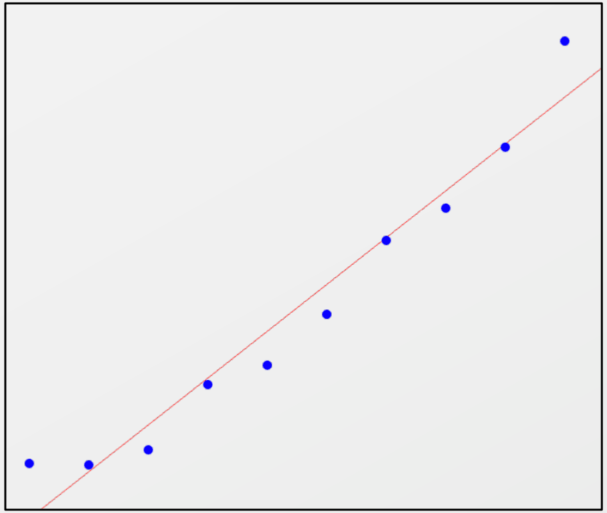
Training set

Validation set

Test set

- *Training set* to determine the ML model parameters
- *Validation set* to adjust hyperparameters and avoid overfitting
  - Hyperparameters define structure of ML model or guide training
  - Validation may be added to training once hyperparameters are set
- *Test set* to assess the ML model
  - It should play no part in training or validation
- Data should be distributed at random among the sets
  - 65:15:20 distribution of training:validation:test is typical

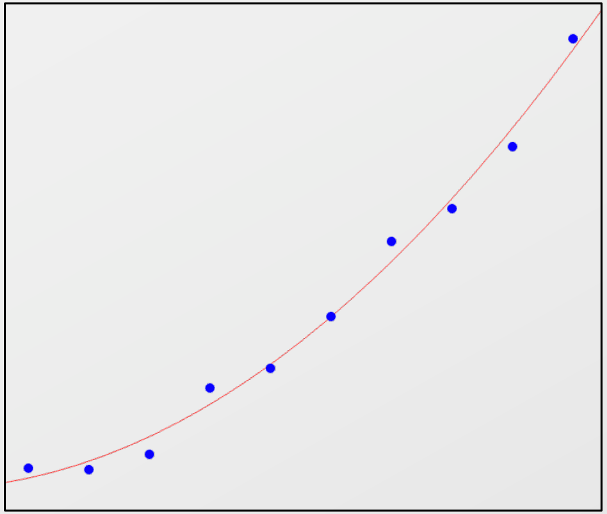
# Both over- and underfitting are bad



## Underfitting

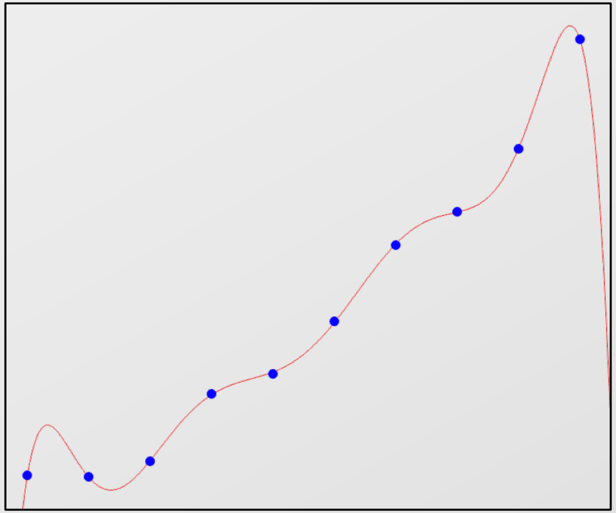
Meaningful relationships in datasets are not learned

training error: high
validation error: high



## Appropriate fitting

training error: low
validation error: low



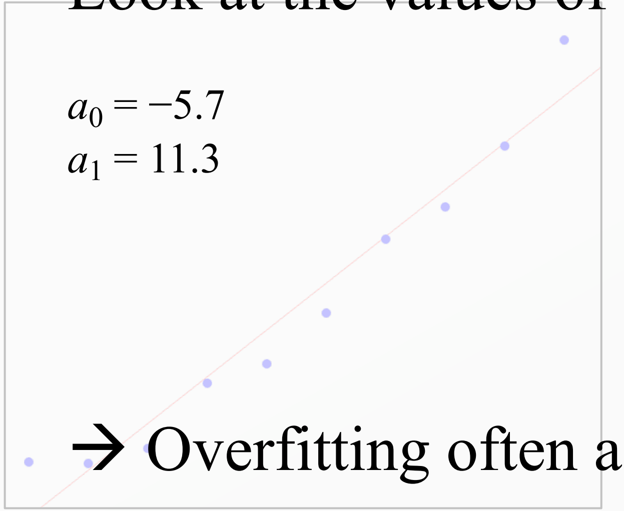
## Overfitting

Noise in training data is learned, and does not generalize

training error: low
validation error: high

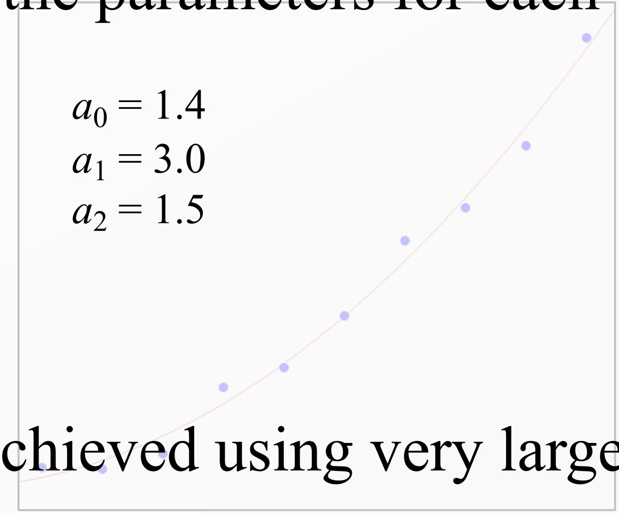
# Both over- and underfitting are bad

Look at the values of the parameters for each of these fits $\sum a_i x^i$



$a_0 = -5.7$

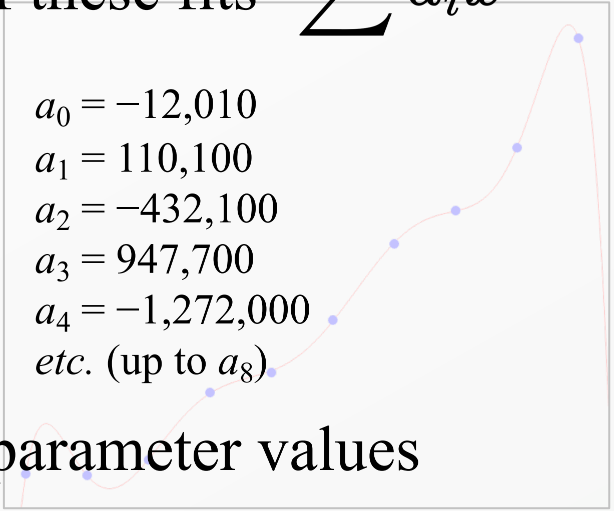
$a_1 = 11.3$



$a_0 = 1.4$

$a_1 = 3.0$

$a_2 = 1.5$



$a_0 = -12{,}010$

$a_1 = 110{,}100$

$a_2 = -432{,}100$

$a_3 = 947{,}700$

$a_4 = -1{,}272{,}000$

*etc.* (up to $a_8$)

→ Overfitting often achieved using very large parameter values

| Underfitting | Appropriate fitting | Overfitting |
|---|---|---|
| Meaningful relationships in datasets are not learned | | Noise in training data is learned, and does not generalize |
| training error: high<br>validation error: high | training error: low<br>validation error: low | training error: low<br>validation error: high |

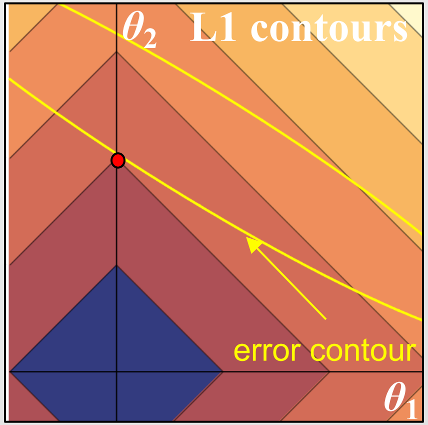
# *Regularization* calibrates ML models to prevent underfitting or overfitting

- One approach: add a cost-function penalty for large parameters
  - L1 regularization or *lasso* regression:

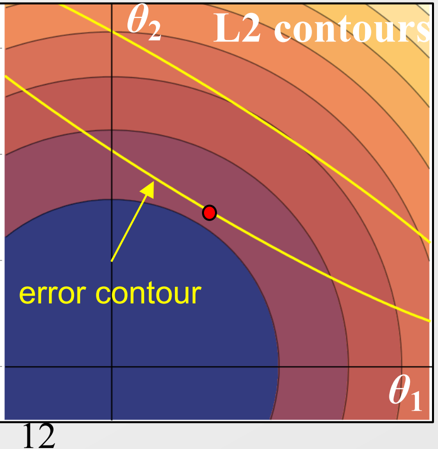
$$\text{cost} = \text{error} + \lambda \sum_i |\theta_i|$$

    - Favors sparsity of coefficients, making some exactly 0
    - A tool for feature selection

λ is an example of a hyperparameter



For L1 regularization, intersection with minimum contour is likely to happen at a vertex of penalty function



  - L2 regularization, or *ridge* regression:

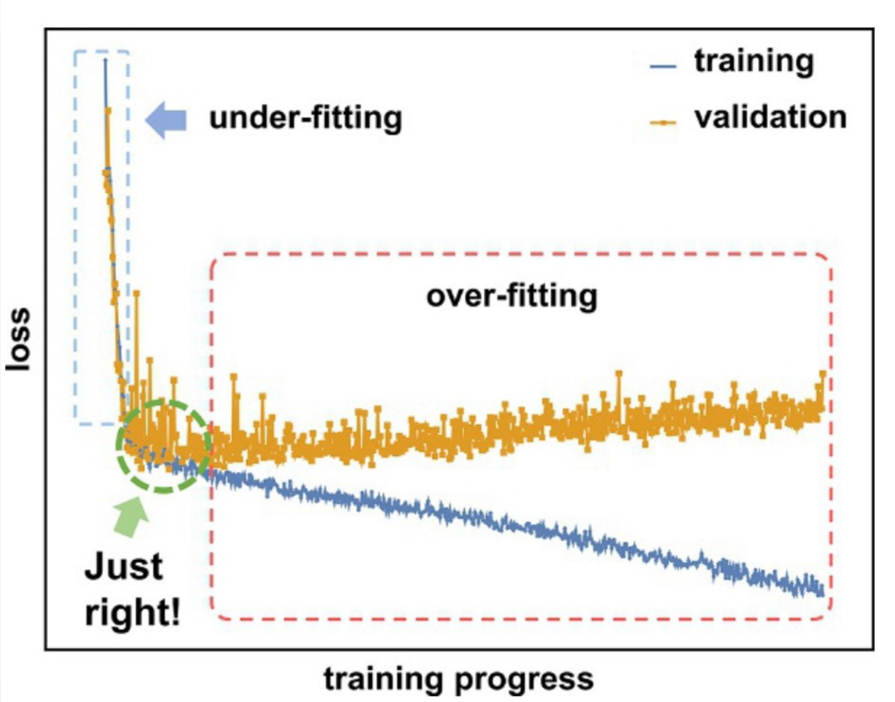
$$\text{cost} = \text{error} + \lambda \sum_i \theta_i^2$$

Overall compression toward smaller coefficients

Also known as *Tikhonov regularization*

# *Early stopping* is another regularization method

- Perform parameter optimization on training set
- Occasionally evaluate error using validation set
- Where validation error begins to increase, halt optimization

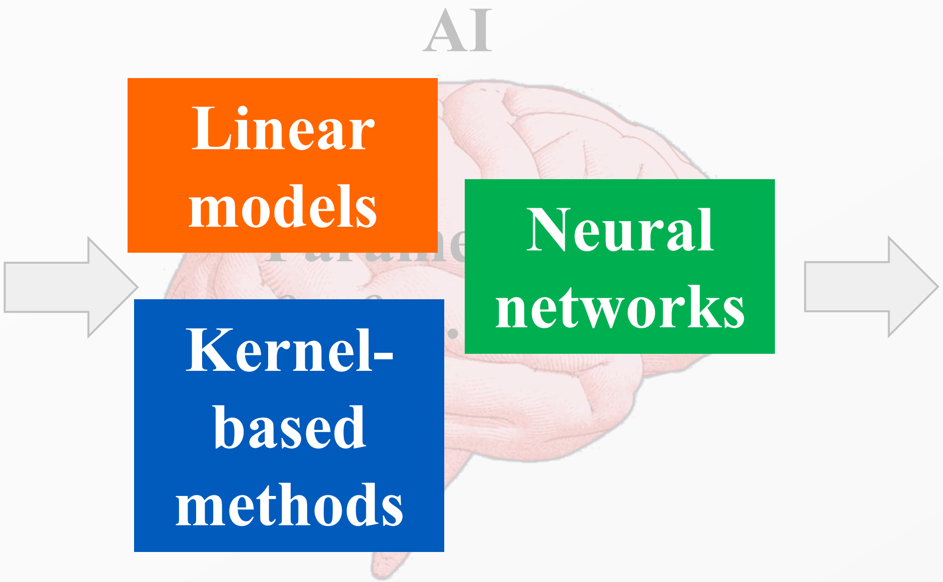


Dral et al., doi.org/10.1016/B978-0-323-90049-2.00011-1

# We will examine three general approaches to formulating a machine-learning potential

Input variables, or *features*

$x_1, x_2, \ldots, x_m$



Output variable, or *label*, *y*

Selection of features (descriptors) is often an art that can be crucial to ML success

There may be many, many parameters

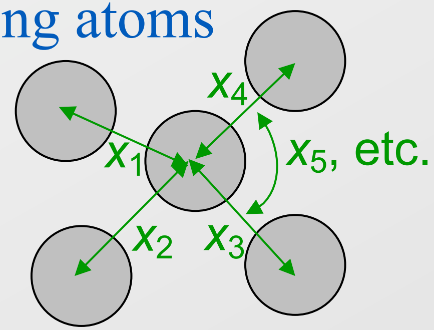
Output is usually a scalar, but can be multivariate

# A model is *linear* if it has linear dependence on its parameters (not that it fits using linear functions)

- Energy given as a sum of one-body energies

$$E = \sum_{i}^{N_{\text{atoms}}} E_i$$

  - Atom energies are, in turn, given via a set of descriptors that depend on positions of neighboring atoms



- General linear model has a simple form: $\mathbf{y} = \mathbf{X}\widehat{\mathbf{w}}$

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix} = \underbrace{\begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{pmatrix}}_{} \begin{pmatrix} \widehat{w}_1 \\ \widehat{w}_2 \\ \vdots \\ \widehat{w}_p \end{pmatrix}$$

*n* {**x**,*y*} observation pairs

*p* features and parameters

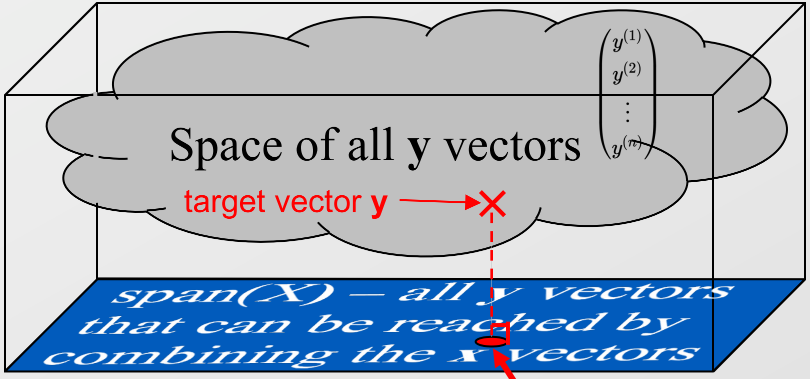
# Ordinary least squares (OLS) is the basic, assumption-free optimization of a linear model

$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathcal{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \left( y^{(i)} - \sum\nolimits_j^p x_j^{(i)} w_j \right)^2 = \underset{\mathbf{w}\in\mathcal{R}^p}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2$$

- Given a set of $\boldsymbol{x}_j$ vectors, how can we combine them to get as close to the **y** vector as possible? Think geometrically.

$$\begin{array}{cccc} \multicolumn{4}{c}{\begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{pmatrix}} \\ \uparrow & \uparrow & & \uparrow \\ \mathbf{x}_1 & \mathbf{x}_2 & \ldots & \mathbf{x}_p \end{array}$$

$\mathbf{x}_\mathrm{j}$ vectors


Space of all **y** vectors
$\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}$
target vector **y**
span(X) – all y vectors that can be reached by combining the x vectors

closest vector in span(**X**): $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}$

$\mathbf{y} - \hat{\mathbf{y}}$ is orthogonal to all $\mathbf{x}_\mathrm{j}$:

$$\mathbf{X}^\mathrm{T}(\mathbf{y} - \hat{\mathbf{y}}) = 0$$

$$\mathbf{X}^\mathrm{T}\mathbf{X}\hat{\mathbf{w}} = \mathbf{X}^\mathrm{T}\mathbf{y}$$

$$\boxed{\hat{\mathbf{w}} = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y}}$$

Direct solution for optimum is obtained (if enough data)

# Regularization is needed if the parameters exceed the number of data ($X^TX$ not invertible)

- Where **w** doesn't have a unique solution, its evaluation is arbitrary to a degree, and prediction performance will suffer
  - Situation is likely where $p > n$
  - Regularization can alleviate this

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{pmatrix}$$

$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

- Ridge regression

$$\hat{\mathbf{w}} = \underset{\mathbf{w} \in \mathcal{R}^p}{\operatorname{argmin}} \left( \frac{1}{2} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda ||\mathbf{w}||_2^2 \right)$$

- Still a quadratic form; analytic minimum

$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

  - Inverse will exist for nonzero λ

# Lasso regression does not allow a direct solution for the minimum cost

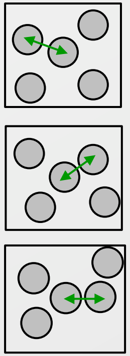
$$\hat{\mathbf{w}} = \underset{\mathbf{w}\in\mathcal{R}^p}{\operatorname{argmin}}\left(\frac{1}{2}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_{\boxed{1}}\right)$$

- Not a quadratic form, so more complicated to minimize
- Consider a single-feature example $\hat{w} = \underset{w\in\mathcal{R}}{\operatorname{argmin}}\ f(w)$

$$f(w) = \frac{1}{2}\left(w^2||\mathbf{x}||_2^2 + ||\mathbf{y}||_2^2\right) - w\mathbf{x}^{\mathrm{T}}\mathbf{y} + \lambda|w|$$

Example

$x_1 \equiv r_{ij}^{\min}$



$x_1^{(1)} = 0.5$

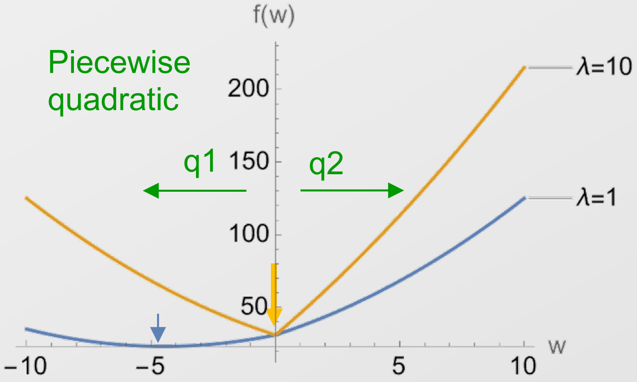
$y^{(1)} = -2$

$x_1^{(2)} = 0.2$

$y^{(2)} = -7$

$x_1^{(3)} = 0.7$

$y^{(3)} = -3$

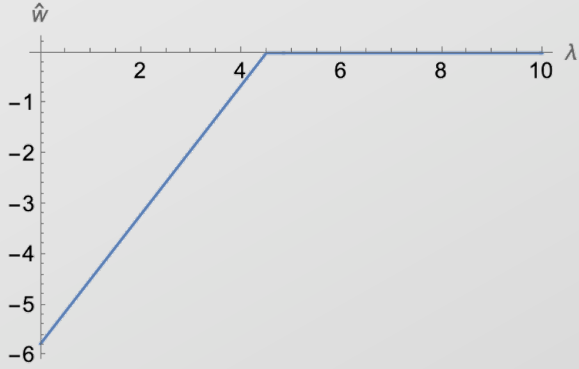


Piecewise quadratic

In general, $\hat{w} = 0$ for $|\mathbf{x}_1^{\mathrm{T}}\mathbf{y}| \leq \lambda$

$$\begin{array}{cc} \mathbf{x}_1 & \mathbf{y} \\ \begin{pmatrix} 0.5 \\ 0.2 \\ 0.7 \end{pmatrix} & \begin{pmatrix} -2 \\ -7 \\ -3 \end{pmatrix} \end{array}$$

$\mathbf{x}_1^{\mathrm{T}}\mathbf{y} = -4.5$



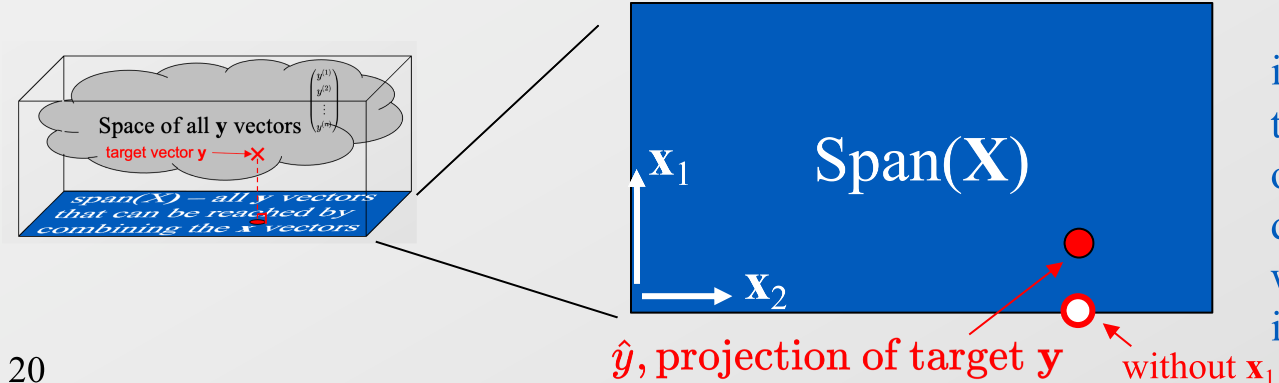
# General lasso case is also piecewise quadratic, but with more ($2^p$) pieces

$$f(w) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \frac{1}{2}\|\mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

- Solution indicates

$$\begin{array}{ll} \left|\mathbf{x}_k^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}})\right| \le \lambda & w_k = 0 \\ \left|\mathbf{x}_k^{\mathrm{T}}(\mathbf{y} - \underbrace{\mathbf{X}\hat{\mathbf{w}}}_{\hat{y}})\right| > \lambda & \mathbf{x}_k^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}) = \lambda\,\mathrm{sgn}(w_k) \end{array}$$

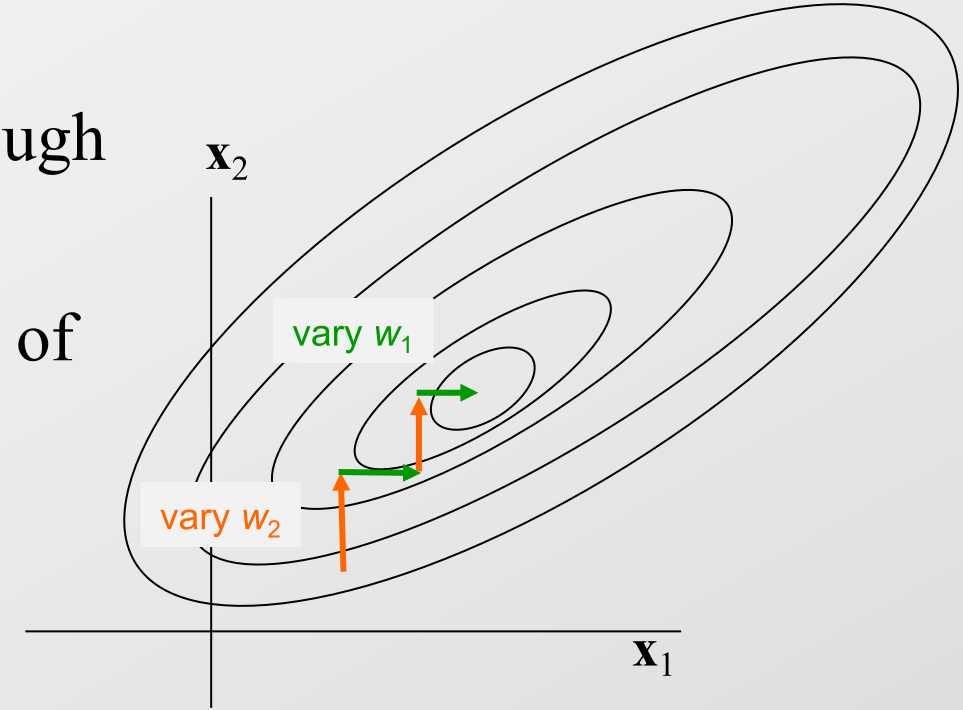
  - Not a closed-form solution, but it gives some geometric insight



if $\mathbf{x}_1^{\mathrm{T}}(\mathbf{y} - \hat{\mathbf{y}}) \le \lambda$ then feature #1 is almost orthogonal to difference and doesn't add enough to be worthwhile to keep, so its weight is zeroed out

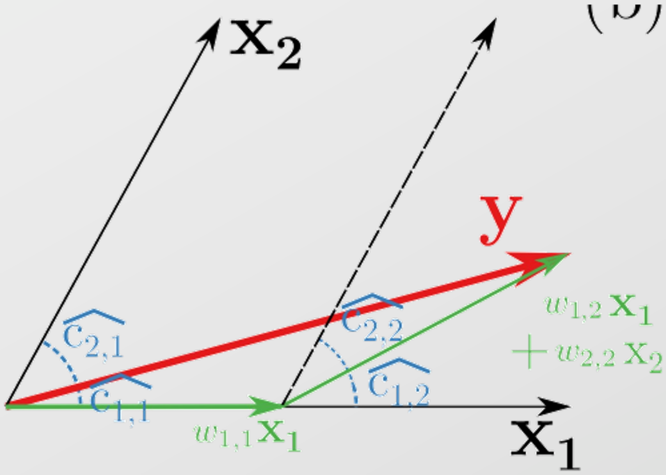
# Coordinate descent is a popular algorithm for finding lasso optimum

- Pick initial guess **w**
- In iteration *i*, proceed through all $w_k$ in random order
- For each *k*, find minimum of $f(\mathbf{w})$ by varying only $w_k$
- Repeat for next *i* to convergence



# *Least angle regression selection* (LARS) is a lasso solution that leads to sparse w (many zero)

- Start with all $w_k = 0$
- Select $k$ for which $\mathbf{x}_k^T\mathbf{y}$ is largest
- Increase $w_k$ until another descriptor $j$ has larger correlation with residual: $\mathbf{x}_j^T(\mathbf{y}-w_k\mathbf{x}_k)$
- Optimize $w_k$ and $w_j$ in direction equiangular between $\mathbf{x}_k$ and $\mathbf{x}_j$ until a 3rd is more correlated with new residual



Tallec et al.

# Suggested Reading/Viewing

- Eugen Hruska and Fang Liu, Chapter 6, Machine learning: An overview. In *Quantum Chemistry in the Age of Machine Learning*.
  - Posted on UBLearns
- Gauthier Tallec, Gaétan Laurens, Owen Fresse-Colson, and Julien Lam, Chapter 11, Potentials based on linear models. In *Quantum Chemistry in the Age of Machine Learning*.
  - Posted on UBLearns