NANO281 Review

Shyue Ping Ong

University of California, San Diego

NANO281

Overview

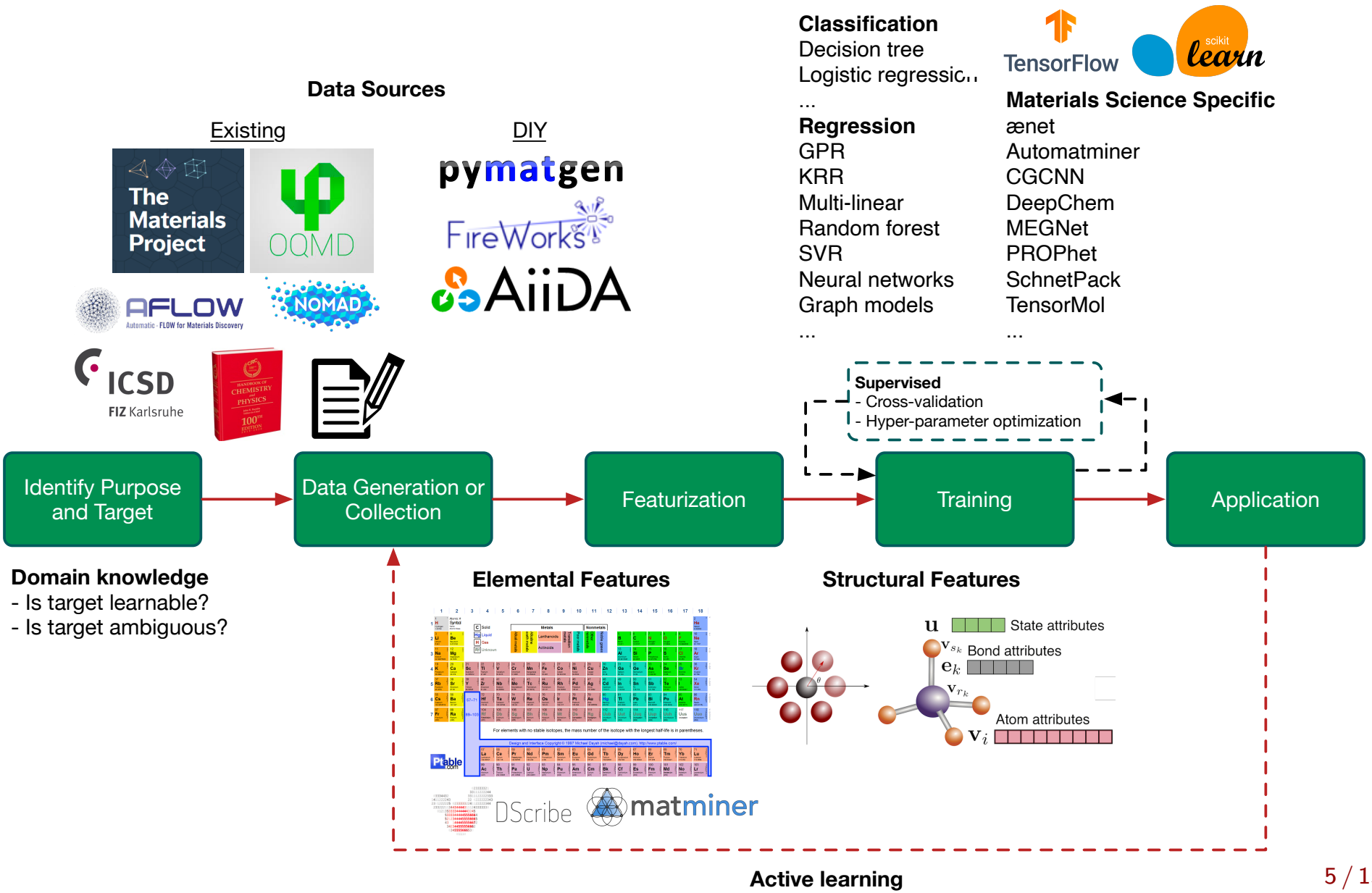
Preliminaries

- In this class, we have provided a broad survey of data science techniques as they are applied to materials science problems.
- In doing so, we have opted for breadth rather than depth, and eschewed most of the mathematical derivations that is typical in data science classes.
- In this final lecture, I want to do an overview of the coverage to reinforce the lessons learnt.
- The structure of this overview is deliberately different from the sequence in which we have done the entire class. Now that you have the details in your mind, this lecture provides a high-level overview of key takeaways and guiding principles.

Why Data Science?

- Materials data is exploding in quantity and quality.
  - High-throughput/combinatorial experiments.
  - High-throughput first principles computations.
- Types of problems that can most benefit from data science:
  - Things that are still too difficult to perform experiments on or compute.
  - Relationships that are beyond our understanding (at the present moment).

ML flowchart



Data and Featurization

- Data generation, collection and wrangling is typically the most time-consuming portion of the whole process.
- Sources: Experimental (ICSD, Pauling file, literature, ...), Computational (Materials Project, OQMD, AFLOW, NOMAD, ...).
- Quality and quantity remains a big issue in materials science.
- Featurization - typically the choice that affects model performance the most
  - Composition-based features (e.g., electronegativity, atomic radii, etc.): intuitively simple, readily available, but clearly cannot capture structural differences.
  - Structure-based features (including graphs): much greater complexity, typically must obey symmetries of system to be effective.

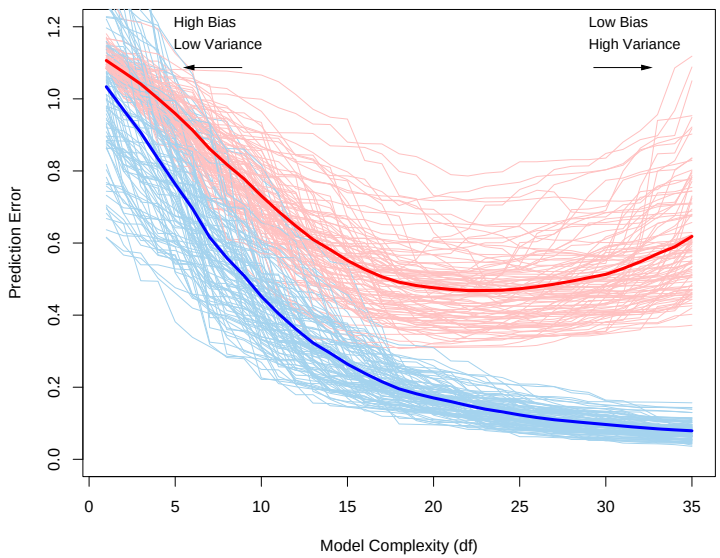
Model fitting

- All model fittings follow the same basic principle - minimizing of some *loss function*, $L(\theta; y_i, \mathbf{x_i})$ either analytically or by numerical procedures (e.g., gradient descent).

Task	Loss function	Equation
Regression	Mean squared error	$\sum_{i=1}^{N}(y_i - f(x_i))^2$
	Mean absolute error	$\sum_{i=1}^{N} \lvert y_i - f(x_i) \rvert$
Classification	Missclassification rate	$I(\text{sign}(f) \neq y)$
	Exponential loss	$e^{-yf(x)}$
	Binomial/multinomial	$-\sum_{k=1}^{K} I(y = G_k) f_k(x) + \log\left(\sum_{l=1}^{K} e^{f_l(x)}\right)$
	Binomial deviance	$\log(1 + e^{-2yf})$

Model assessment and selection

- Model performance is related to its performance on *independent test data*.
  - Training error: L over training set.
  - Test error: L over independent test set.
- Model complexity increases as the number of parameters increases.
- Training errors **always** decrease with increasing model complexity.
- Test errors are high when model complexity is too low (underfitting) or too high (overfitting).



Training, validation and test data

- Model selection: estimating the performance of different models in order to choose the best one.
- Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.
  - Training set: For training the model.
  - Validation set: For estimating prediction error to select the model.
  - Test set: For assessing the generalization error of the final model. This should not be used in fitting the model.
- Typical training:validation:test split is 50:25:25 or 80:10:10, or in very data-poor situations, maybe even 90:5:5.

K-fold cross validation (CV)

- Simplest and most widely used approach for model validation.
- Data set is split into K buckets (usually by random).
- Typical values of K is 5 or 10. $K = N$ is known as "leave-one-out" CV.

Train	Train	Validate	Train	Train

- CV score is computed on the validate data set after training on the train data:

$$CV(\hat{f}^{-k(i)}, \alpha) = \frac{1}{N_{k(i)}} \sum_{i=1}^{N_{k(i)}} L(y_i, \hat{f}^{-k(i)}(x_i, \alpha))$$

- assuming the k^{th} data bucket has $N_{k(i)}$ data points and $\hat{f}^{-k(i)}$ refers to the model fitted with the k^{th} data left out.

Regularization

- Often, one starts from a model that is more complex first, and then reduce model complexity.
- Reducing model complexity decreases risk of overfitting (improve generalizability).
  - Shrink coefficients or weights, sometimes to zero.
  - Subset selection / tree pruning.
- General approach is to add a *penalty term* to loss functions that penalizes overcomplex models, e.g., sum of squares or absolute value of coefficients (e.g., MLR) or weights (NNs), tree size (decision trees), etc. Controlled by some parameter to be specified by model developer that determines size of penalty.
- Bias-variance trade-off:

$$\text{MSE} = \text{var}(\hat{\theta}) + [E(\hat{\theta}) - \theta]^2$$

Types of ML models

- Supervised learning
  - Linear (MLR, Ridge, Lasso, Linear/Quadratic Discriminant)
  - Local/Kernel methods (kNN, kernel density estimation, Gaussian mixture models)
  - Trees (CART, Adaboost, Gradient Boosting, Random Forest)
  - Neural networks
- Unsupervised learning
  - Principal Component Analysis
  - K-means
  - DBSCAN
- And many others that have not been covered (e.g., support vector machines, graph-based models, etc.)

Final Lab

Bibliography

The End