Machine learning and chemistry

Jon Paul Janet [1] Aditya Nandy [2] Nick (Tzuhsiung) Yang [1]

[1]Department of Chemical Engineering, Massachusetts Institute of Technology

[2]Department of Chemistry, Massachusetts Institute of Technology

the Kulik group at MIT, `hjkgrp.mit.edu/`

under the supervision of Professor Heather J. Kulik

for the most recent version and demos: `github.com/jpjanet/ML-chem-workshop`
this revision: `74a2c6d6ccb9047c6c590d07abd7ed567af8d488` on branch `master`

compiled 03/29/2019

Rise of the (chemical) machines

Rise of the (chemical) machines

Something interesting happened at the **CASP 13** protein folding prediction competition in Mexico in December 2018...

Rise of the (chemical) machines

Something interesting happened at the **CASP 13** protein folding prediction competition in Mexico in December 2018...

A new entry, competing in their first CASP, ran away with the no-information category, **winning 25 out of 43 tests**.

Rise of the (chemical) machines

Something interesting happened at the **CASP 13** protein folding prediction competition in Mexico in December 2018...

A new entry, competing in their first CASP, ran away with the no-information category, **winning 25 out of 43 tests**.The next best team won 3 of the remaining tests.

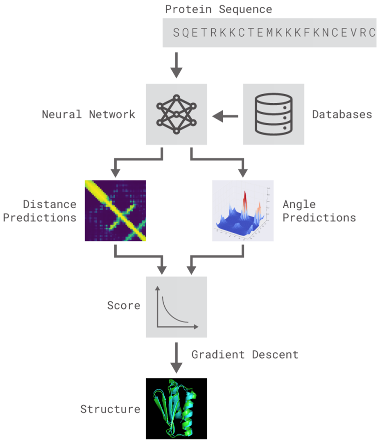
Rise of the (chemical) machines

Something interesting happened at the **CASP 13** protein folding prediction competition in Mexico in December 2018...

A new entry, competing in their first CASP, ran away with the no-information category, **winning 25 out of 43 tests**.The next best team won 3 of the remaining tests.The team was Alphafold, by DeepMind.

Rise of the (chemical) machines

The team was Alphafold, by DeepMind.



Rise of the (chemical) machines

The team was Alphafold, by DeepMind.

"*It is not that machines are going to replace chemists. It's that the chemists who use machines will replace those that do not*"
-Derek Lowe, In the Pipeline

Rise of the (chemical) machines

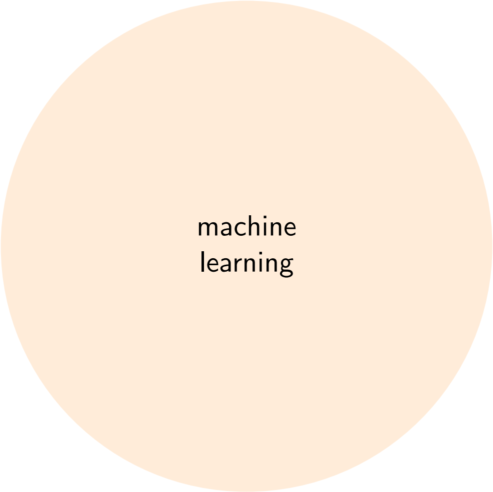
The team was Alphafold, by DeepMind.

"*It is not that machines are going to replace chemists. It's that the chemists who use machines will replace those that do not*"
-Derek Lowe, In the Pipeline

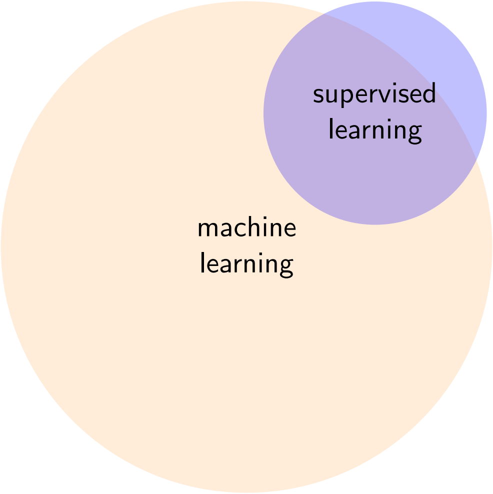
This is probably a bit strong, but all scientists generate data as a product. ML provides new, powerful ways to exploit their that information.

Types of machine learning

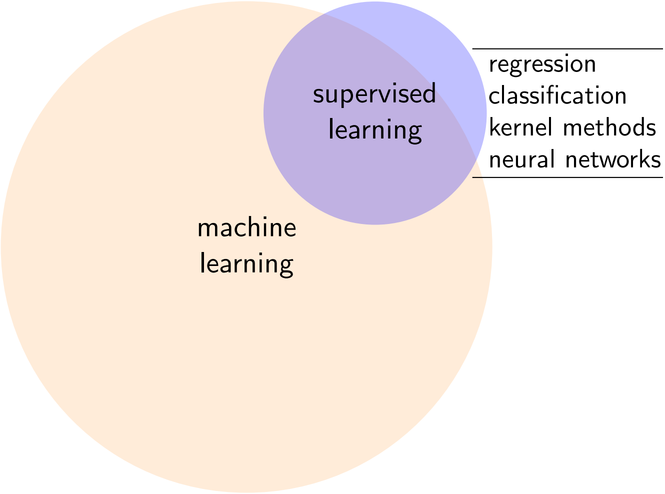
Types of machine learning



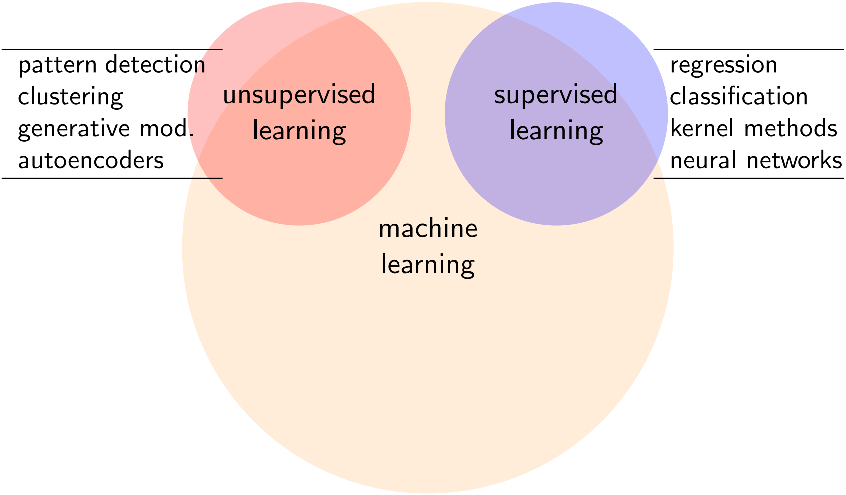
Types of machine learning



Types of machine learning

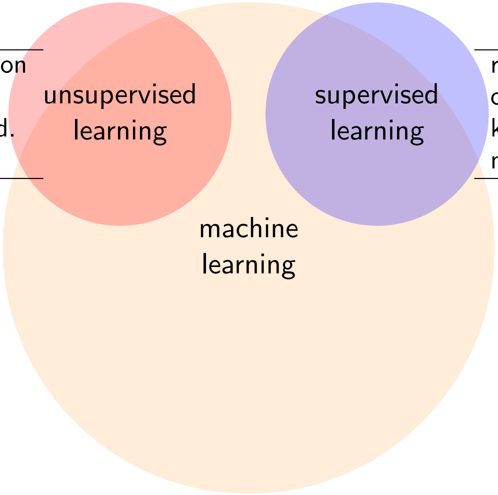


Types of machine learning



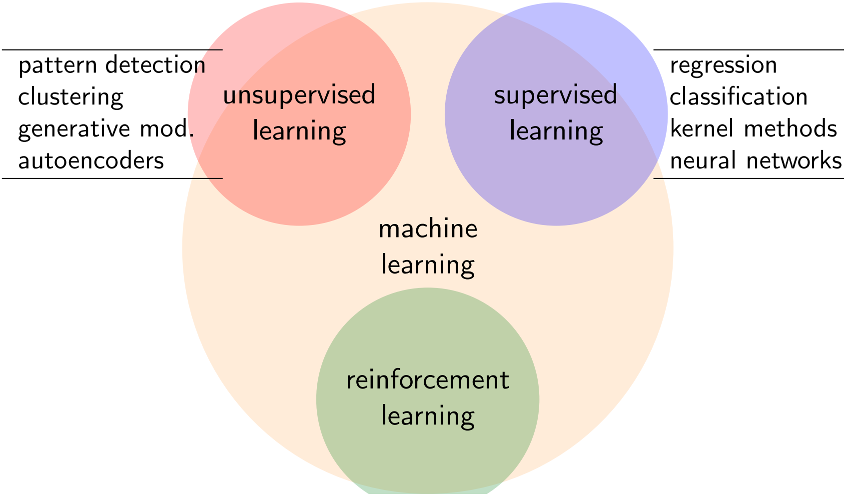
Types of machine learning

pattern detection
clustering
generative mod.
autoencoders

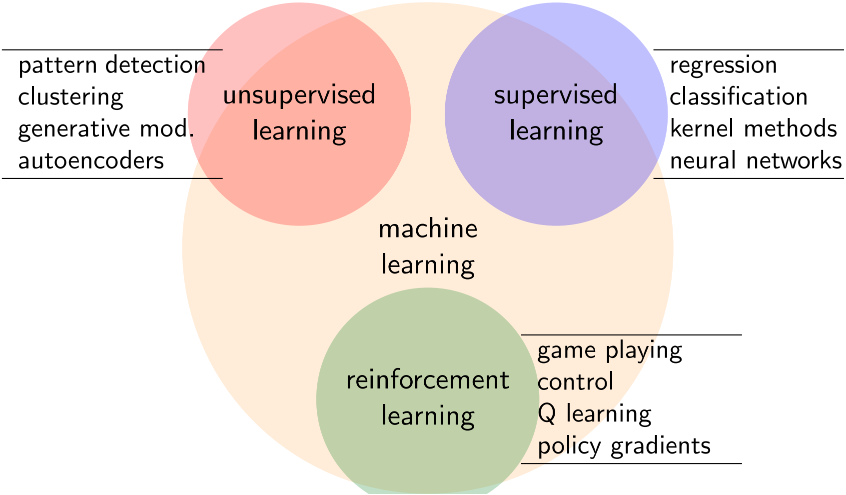


regression
classification
kernel methods
neural networks

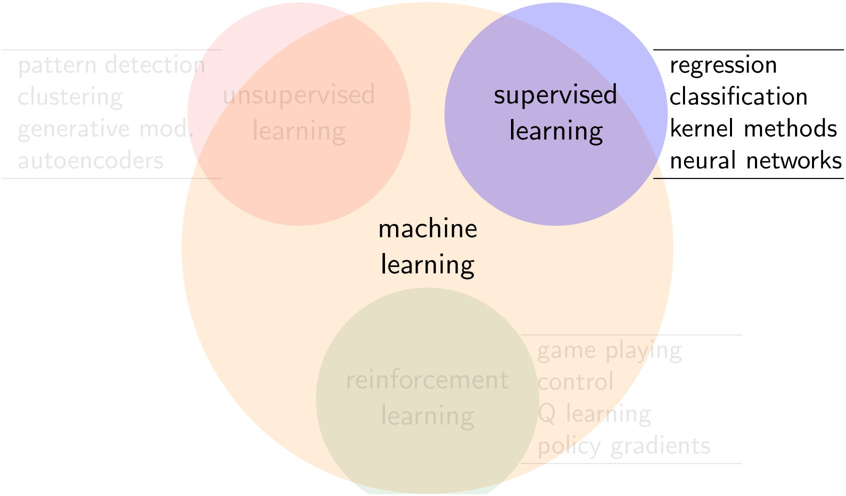
Types of machine learning



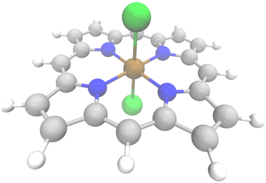
Types of machine learning



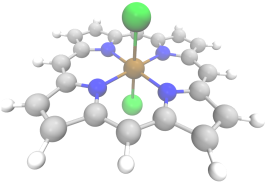
Types of machine learning



Why ML in chemistry?

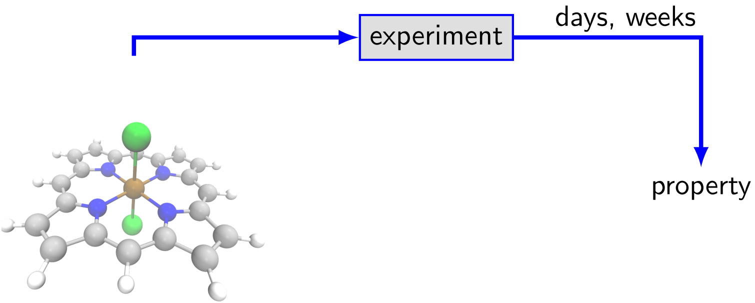


Why ML in chemistry?

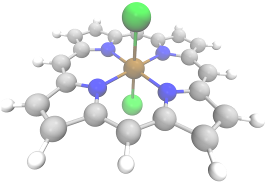


property

Why ML in chemistry?



Why ML in chemistry?



experiment

days, weeks

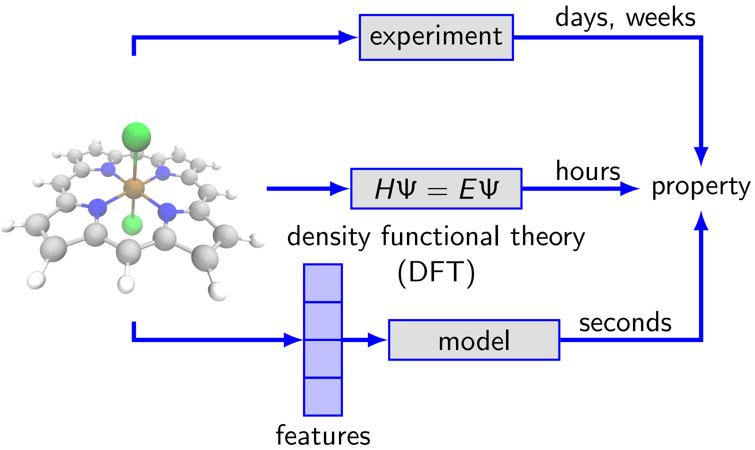
$H\Psi = E\Psi$

hours

property

density functional theory
(DFT)

Why ML in chemistry?



Structure of this talk

We will cover the following topics:

Structure of this talk

We will cover the following topics:

1. Statistical learning theory

Structure of this talk

We will cover the following topics:

1. Statistical learning theory
2. Representations

Structure of this talk

We will cover the following topics:

1. Statistical learning theory
2. Representations
3. Linear and kernel models

Structure of this talk

We will cover the following topics:

1. Statistical learning theory
2. Representations
3. Linear and kernel models
4. Feature Selection and Forests

Structure of this talk

We will cover the following topics:

1. Statistical learning theory
2. Representations
3. Linear and kernel models
4. Feature Selection and Forests
5. Neural networks and representation learning

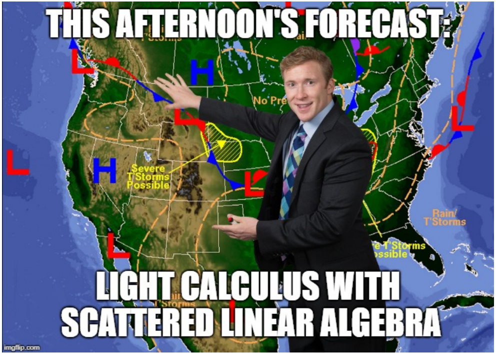
Please ask questions throughout!

Disclaimer

Warning: this talk contains some *light* mathematics.

Disclaimer

Warning: this talk contains some *light* mathematics.



Disclaimer

Warning: this talk contains some *light* mathematics.

Some useful notation:

X	training data, as rows
x^*	one new molecule/systems
$y, \hat{y}$	property(energy?), predicted value
$\mathcal{L} = \|y - \hat{y}\|_2^2$	loss function
W, w	model parameters
$\hat{y} = f(x, W)$	our model

The goal of statistical learning

The goal of statistical learning

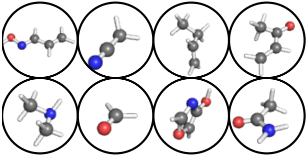
observation

property

The goal of statistical learning

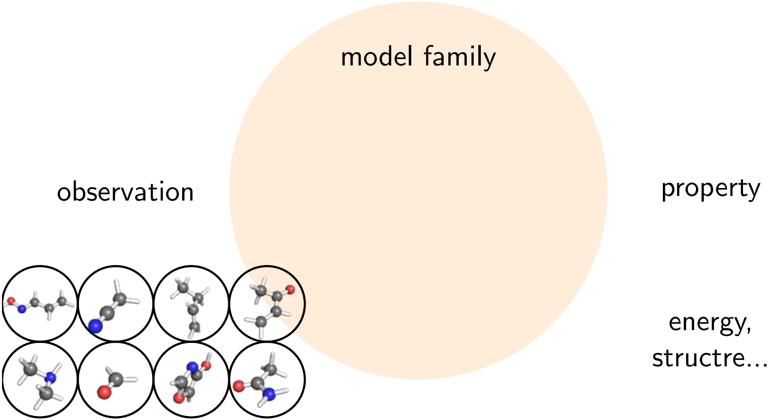
observation

property

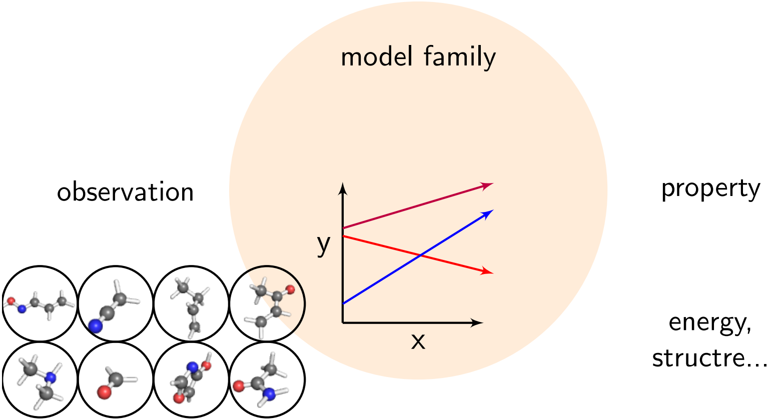


energy,
structre...

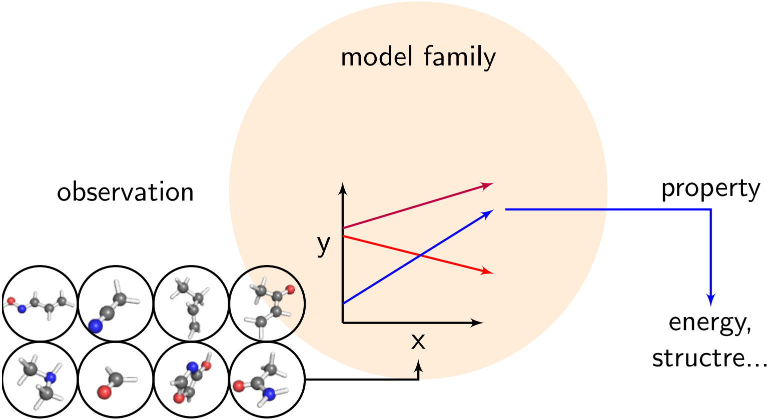
The goal of statistical learning



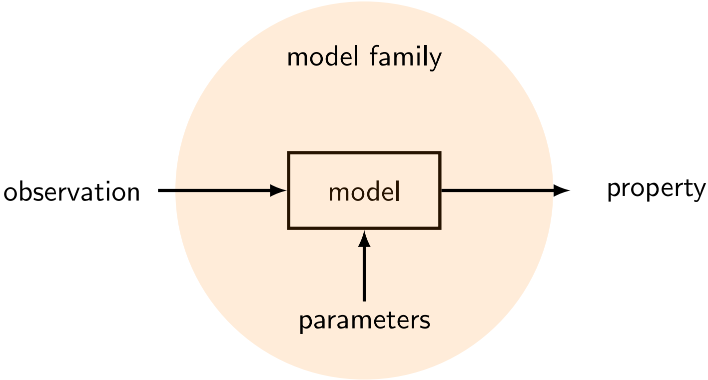
The goal of statistical learning



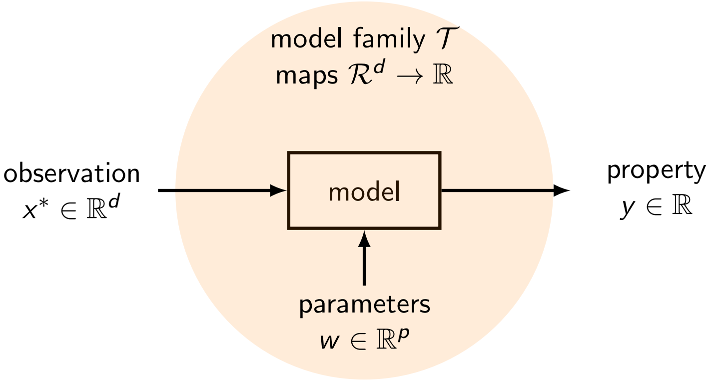
The goal of statistical learning



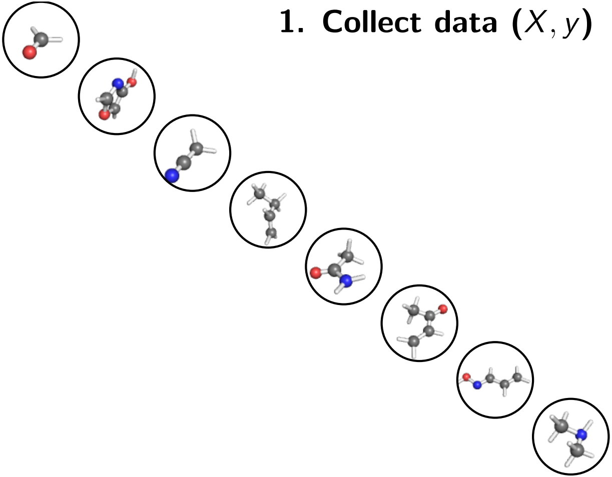
The goal of statistical learning



The goal of statistical learning

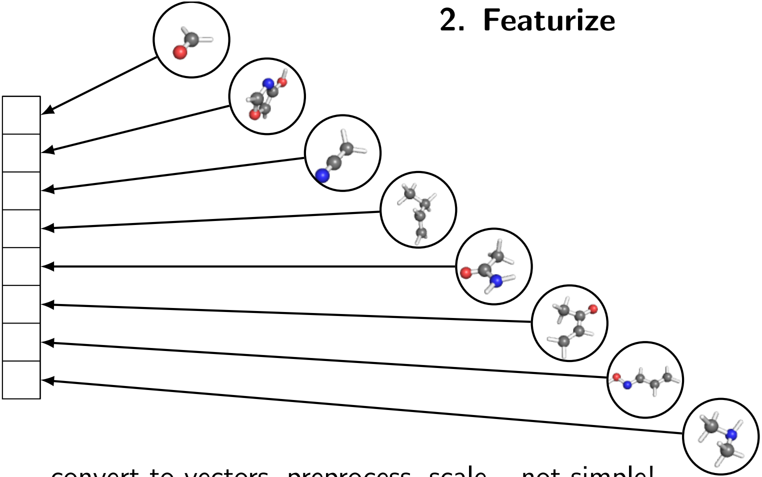


Overview of supervised learning



1. Collect data (X, y)

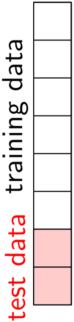
Overview of supervised learning



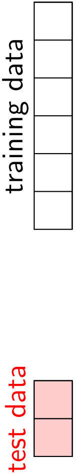
convert to vectors, preprocess, scale – not simple!

Overview of supervised learning

3. Partition data

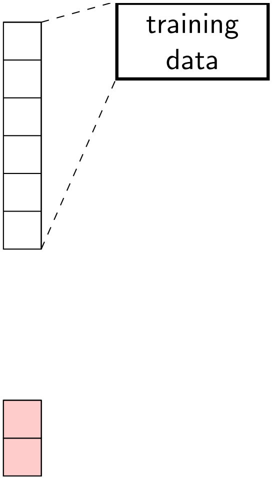


Overview of supervised learning



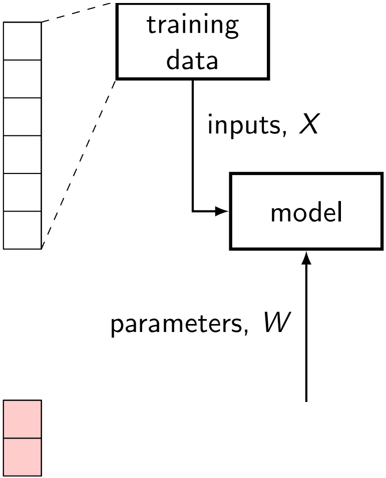
3. Partition data

Overview of supervised learning



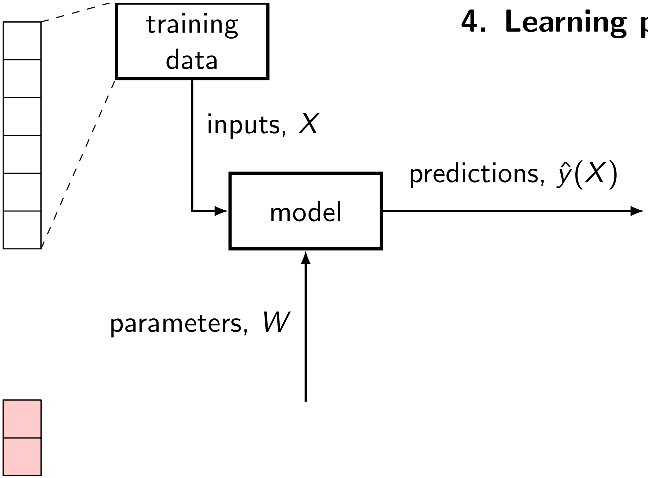
4. Learning phase

Overview of supervised learning



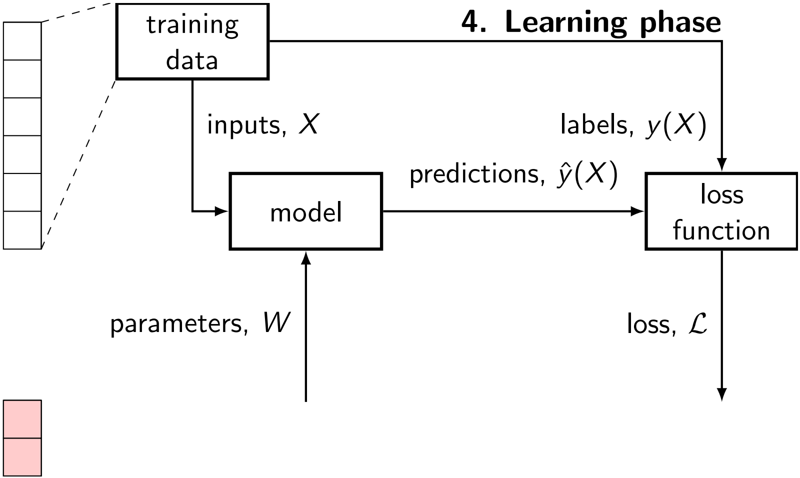
4. Learning phase

Overview of supervised learning

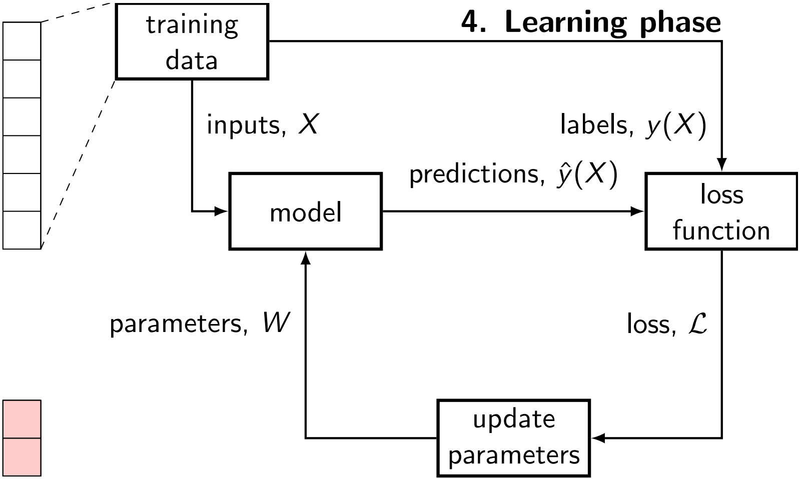


4. Learning phase

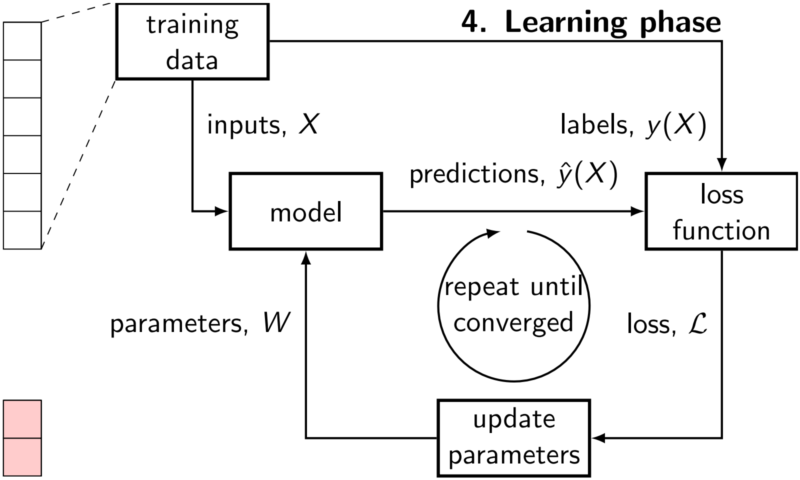
Overview of supervised learning



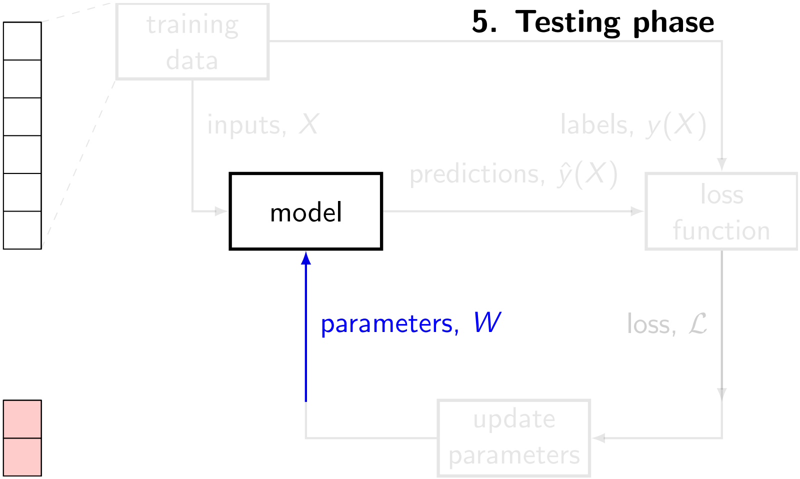
Overview of supervised learning



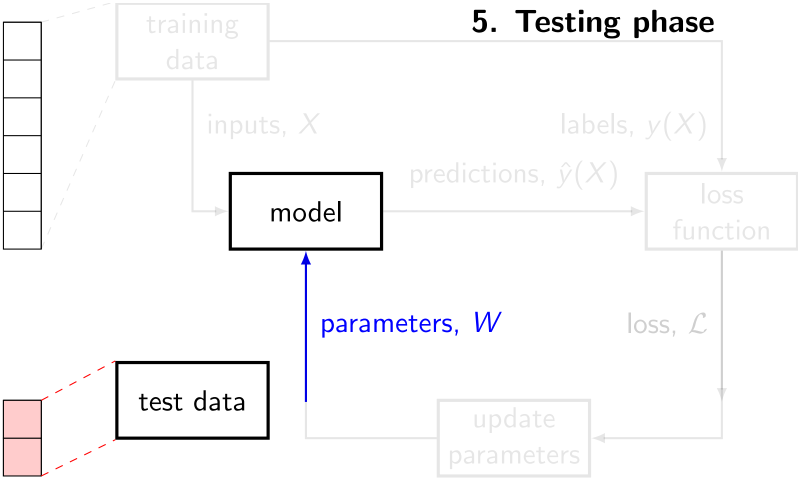
Overview of supervised learning



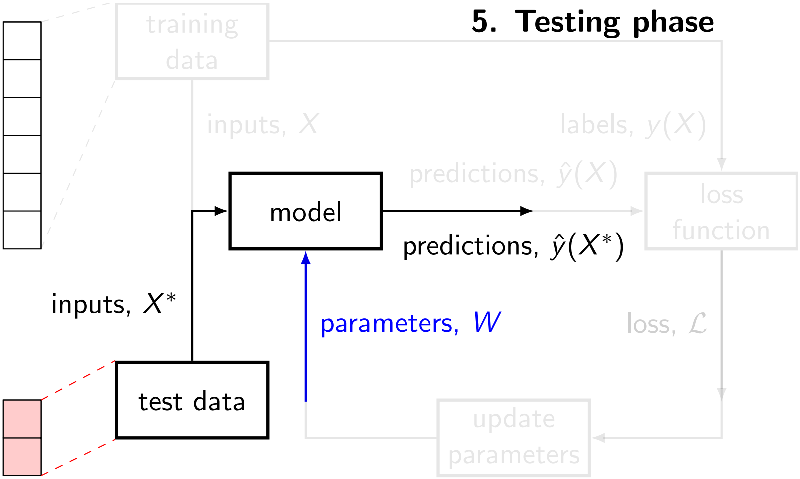
Overview of supervised learning



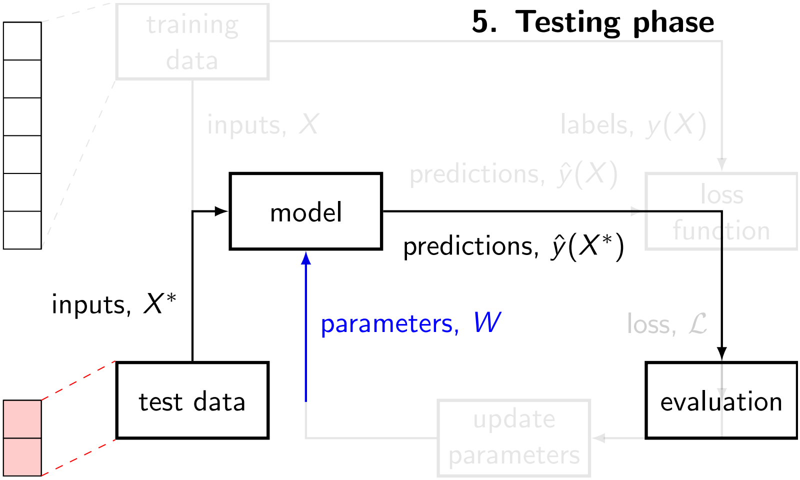
Overview of supervised learning



Overview of supervised learning



Overview of supervised learning



Risk and generalization - I

Our training data defines the *empirical risk*

$$\mathcal{E}_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(y_i, f(x_i)) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

we choose the model to minimize $\mathcal{E}_{emp}(f)$ over all the models in $\mathcal{T}$

$$\hat{f} = \arg\min_{f\in\mathcal{T}}\mathcal{E}_{emp}(f)$$

Risk and generalization - I

Our training data defines the *empirical risk*

$$\mathcal{E}_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(y_i, f(x_i)) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

we choose the model to minimize $\mathcal{E}_{emp}(f)$ over all the models in $\mathcal{T}$

$$\hat{f} = \arg\min_{f\in\mathcal{T}} \mathcal{E}_{emp}(f)$$

we actually care about is the overall risk:

$$\mathcal{E}(f) = \int \mathcal{L}(y, f(x))p(x,y)dxdy = \mathbb{E}\left[\mathcal{L}(Y, f(X))\right]$$

Risk and generalization - I

Our training data defines the *empirical risk*

$$\mathcal{E}_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(y_i, f(x_i)) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

we choose the model to minimize $\mathcal{E}_{emp}(f)$ over all the models in $\mathcal{T}$

$$\hat{f} = \arg\min_{f\in\mathcal{T}}\mathcal{E}_{emp}(f)$$

we actually care about is the overall risk:

$$\mathcal{E}(f) = \int \mathcal{L}(y, f(x))p(x,y)dxdy = \mathbb{E}\left[\mathcal{L}(Y, f(X))\right]$$

with minimum

$$f^* = \mathbb{E}\left[Y|X = x\right]$$

Risk and generalization - II

We cannot expect that f^* is in $\mathcal{T}$. The best we can do is $f^\dagger$:

$$f^\dagger = \arg\min_{f \in \mathcal{T}} \mathcal{E}(f) \qquad \text{(the best model we could pick)}$$

Risk and generalization - II

We cannot expect that f^* is in $\mathcal{T}$. The best we can do is $f^\dagger$:

$$f^\dagger = \arg\min_{f\in\mathcal{T}} \mathcal{E}(f) \qquad \text{(the best model we could pick)}$$

$$\hat{f} = \arg\min_{f\in\mathcal{T}} \mathcal{E}_{emp}(f) \qquad \text{(the model we do pick)}$$

$$f^* = \mathbb{E}\left[Y|X = x\right] \qquad \text{(the `truth')}$$

Risk and generalization - II

We cannot expect that f^* is in $\mathcal{T}$. The best we can do is $f^\dagger$:

$$f^\dagger = \arg\min_{f \in \mathcal{T}} \mathcal{E}(f) \qquad \text{(the best model we could pick)}$$

$$\hat{f} = \arg\min_{f \in \mathcal{T}} \mathcal{E}_{emp}(f) \qquad \text{(the model we do pick)}$$

$$f^* = \mathbb{E}\left[Y|X = x\right] \qquad \text{(the `truth')}$$

We want to generalize, we want the *excess risk* to be small:

$$\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]$$

these terms are **approximation error** and **estimation error**.

Risk and generalization - II

We cannot expect that f^* is in $\mathcal{T}$. The best we can do is $f^\dagger$:

$$
\begin{aligned}
f^\dagger &= \arg\min_{f\in\mathcal{T}} \mathcal{E}(f) && \text{(the best model we could pick)}\\
\hat{f} &= \arg\min_{f\in\mathcal{T}} \mathcal{E}_{emp}(f) && \text{(the model we do pick)}\\
f^* &= \mathbb{E}\left[Y|X=x\right] && \text{(the `truth')}
\end{aligned}
$$

We want to generalize, we want the *excess risk* to be small:

$$
\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]
$$

these terms are **approximation error** and **estimation error**.

Risk and generalization - III

$$\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]$$

Two critical ideas that are worth noting. Under mild assumptions one can show that:

Risk and generalization - III

$$\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]$$

Two critical ideas that are worth noting. Under mild assumptions one can show that:

1. with enough data, a sufficiently complicated model with near-zero approximation error will generalize arbitrarily well.

Risk and generalization - III

$$\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]$$

Two critical ideas that are worth noting. Under mild assumptions one can show that:

1. with enough data, a sufficiently complicated model with near-zero approximation error will generalize arbitrarily well.
2. estimation error is inversely related to model complexity, i.e. more complicated spaces of models require more data to generalize.

Risk and generalization - III

$$\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]$$

Two critical ideas that are worth noting. Under mild assumptions one can show that:

1. with enough data, a sufficiently complicated model with near-zero approximation error will generalize arbitrarily well.
2. estimation error is inversely related to model complexity, i.e. more complicated spaces of models require more data to generalize.

We want $\mathcal{T}$ to be large/complicated enough to have low approximation error, **but no more complicated**.

Risk and generalization - III

$$\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]$$

Two critical ideas that are worth noting. Under mild assumptions one can show that:

1. with enough data, a sufficiently complicated model with near-zero approximation error will generalize arbitrarily well.
2. estimation error is inversely related to model complexity, i.e. more complicated spaces of models require more data to generalize.

With limited data, we are often better off searching for a model in a simpler family models of that 'learn' more robustly and quickly as opposed to very complicated models with lots of parameters.

Risk and generalization - III

$$\mathcal{E}(f^*) - \mathcal{E}(\hat{f}) = \left[\mathcal{E}(f^*) - \mathcal{E}(f^\dagger)\right] + \left[\mathcal{E}(f^\dagger) - \mathcal{E}(\hat{f})\right]$$

Two critical ideas that are worth noting. Under mild assumptions one can show that:

1. with enough data, a sufficiently complicated model with near-zero approximation error will generalize arbitrarily well.
2. estimation error is inversely related to model complexity, i.e. more complicated spaces of models require more data to generalize.

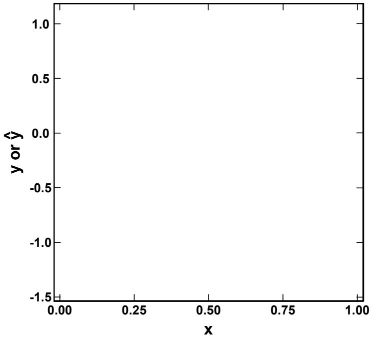
Conversely, a simple model will stop improving with more data past a certain point – where the approximation error dominates.

Risk and generalization - IV

Let us use **polynomials** to estimate:

$$y(x) = \sin(2\pi x)$$

Note that $f^* \notin \mathcal{T}$!

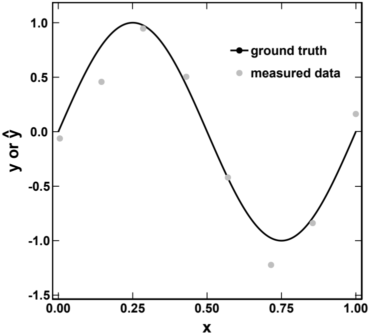


Risk and generalization - IV

Let us use **polynomials** to estimate:

$$y(x) = \sin(2\pi x)$$

Note that $f^* \notin \mathcal{T}$!
Assume 8 measurements with noise $\mathcal{N}(0, 0.2)$



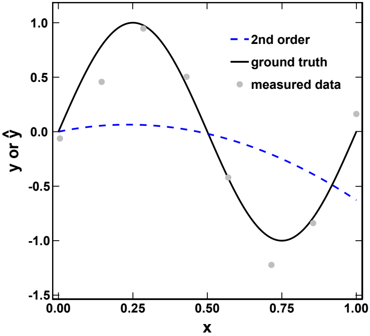
Risk and generalization - IV

Let us use **polynomials** to estimate:

$$y(x) = \sin(2\pi x)$$

Note that $f^* \notin \mathcal{T}$!

Start with degree 2...



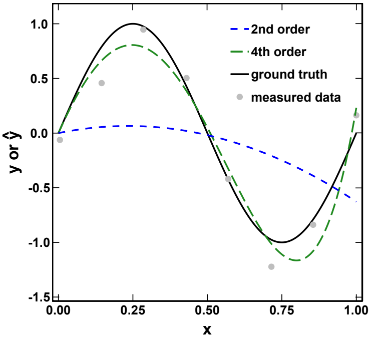
Risk and generalization - IV

Let us use **polynomials** to estimate:

$$y(x) = \sin(2\pi x)$$

Note that $f^* \notin \mathcal{T}$!

Start with degree 2...What happens when we increase the order?

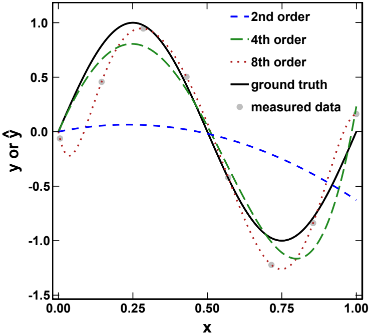


Risk and generalization - IV

Let us use **polynomials** to estimate:

$$y(x) = \sin(2\pi x)$$

Note that $f^* \notin \mathcal{T}$!



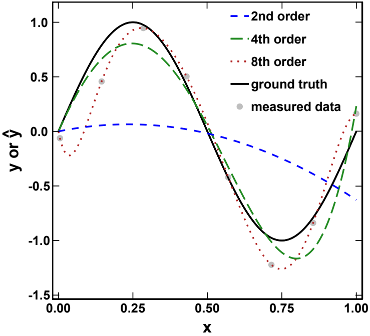
Risk and generalization - IV

Let us use **polynomials** to estimate:

$$y(x) = \sin(2\pi x)$$

Note that $f^* \notin \mathcal{T}$!

What do the risk terms look like?



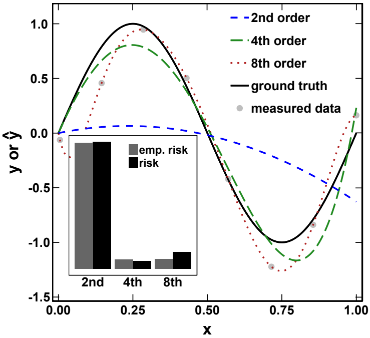
Risk and generalization - IV

Let us use **polynomials** to estimate:

$$y(x) = \sin(2\pi x)$$

Note that $f^* \notin \mathcal{T}$!

What do the risk terms look like?



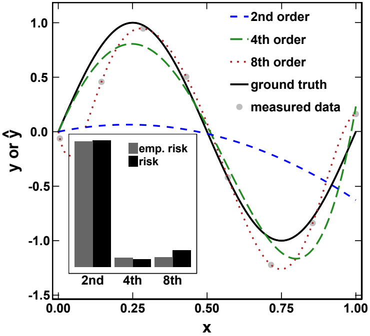
Risk and generalization - IV

Let us use **polynomials** to estimate:

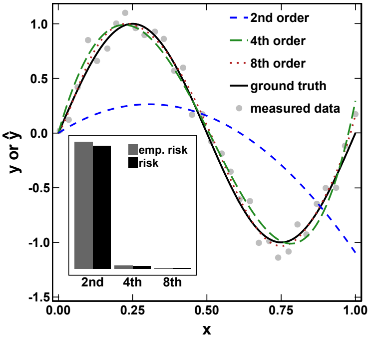
$$y(x) = \sin(2\pi x)$$

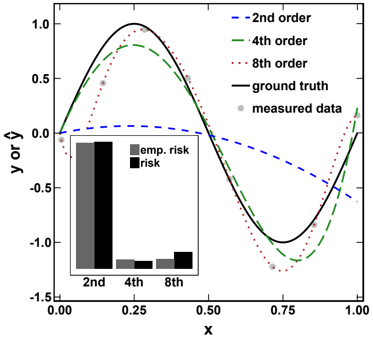
Note that $f^* \notin \mathcal{T}$!

What happens if we add more data?



Risk and generalization - IV





What happens if we add more data?

Controlling Complexity

Adapting model complexity to fit the problem is necessary. However, it is not obvious how to do this is a constrolled way. Hence, we need a way prevent over-fitting that is applicable to many different types of model – **regularization**.

.

Controlling Complexity

Adapting model complexity to fit the problem is necessary. However, it is not obvious how to do this is a constrolled way. Hence, we need a way prevent over-fitting that is applicable to many different types of model – **regularization**.
We instead choose to minimize:

$$\mathcal{L}'(y, f(x, w)) = \mathcal{L}(y, f(x, W)) + \lambda R(W)$$

.

Controlling Complexity

Adapting model complexity to fit the problem is necessary. However, it is not obvious how to do this is a constrolled way. Hence, we need a way prevent over-fitting that is applicable to many different types of model – **regularization**.
We instead choose to minimize:

$$\mathcal{L}'(y, f(x, w)) = \mathcal{L}(y, f(x, W)) + \lambda R(W)$$
$$\text{(Tiknohov/Ridge)} = \frac{1}{n} \|y - f(x, W)\|_2^2 + \lambda \|W\|_2^2$$

.

Controlling Complexity

Adapting model complexity to fit the problem is necessary. However, it is not obvious how to do this is a constrolled way. Hence, we need a way prevent over-fitting that is applicable to many different types of model – **regularization**.
We instead choose to minimize:

$$\mathcal{L}'(y, f(x, w)) = \mathcal{L}(y, f(x, W)) + \lambda R(W)$$
$$\text{(Tiknohov/Ridge)} = \frac{1}{n} \|y - f(x, W)\|_2^2 + \lambda \|W\|_2^2$$

We choose to penalize terms with large weights, i.e. complicated functions.

Controlling Complexity

Adapting model complexity to fit the problem is necessary. However, it is not obvious how to do this is a constrolled way. Hence, we need a way prevent over-fitting that is applicable to many different types of model – **regularization**.
We instead choose to minimize:

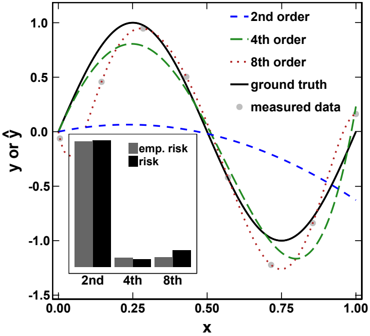
$$\mathcal{L}'(y, f(x,w)) = \mathcal{L}(y, f(x,W)) + \lambda R(W)$$
$$(\text{Tiknohov/Ridge}) = \frac{1}{n}\|y - f(x,W)\|_2^2 + \lambda \|W\|_2^2$$

We choose to penalize terms with large weights, i.e. complicated functions.

- This makes empirical errors worse.
- This *can* improve generalization/excess risk.

Controlling Complexity

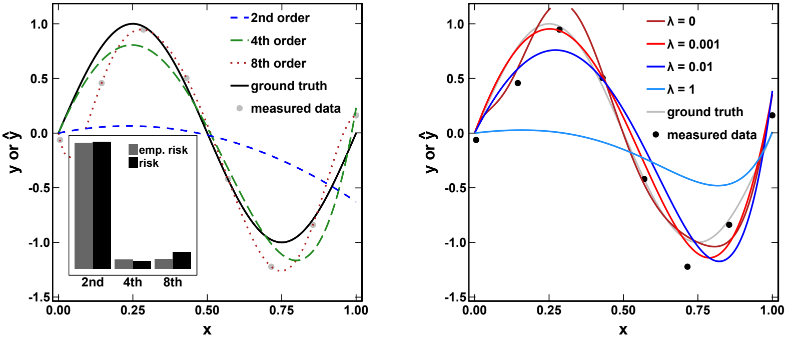
Let's return to our previous example:



Remember: larger λ = simpler, flatter model.

Controlling Complexity

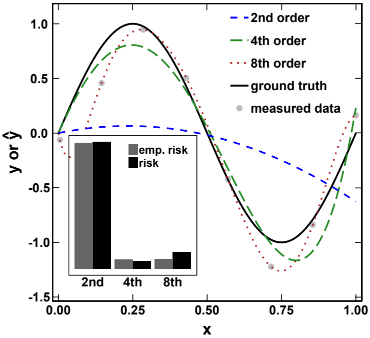
Let's return to our previous example:

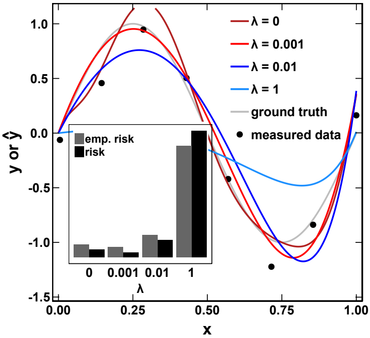


Remember: larger λ = simpler, flatter model.

Controlling Complexity

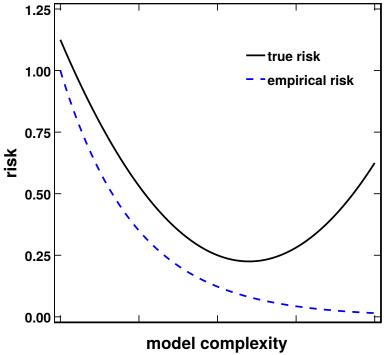
Let's return to our previous example:



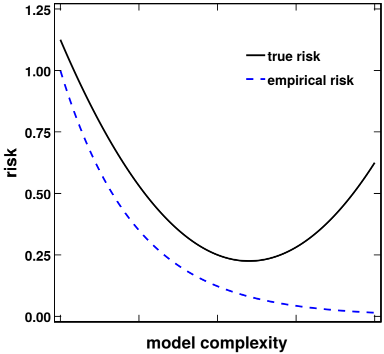


Remember: larger λ = simpler, flatter model.

Controlling Complexity

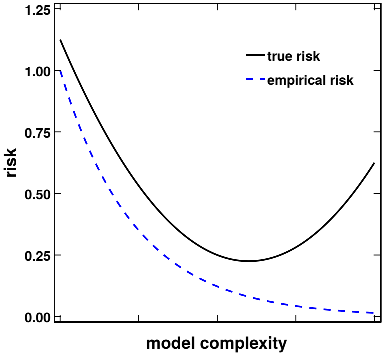


Controlling Complexity



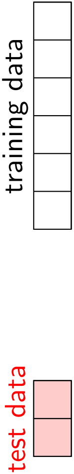
None of this helps us understand a how complicated our model should be.

Controlling Complexity

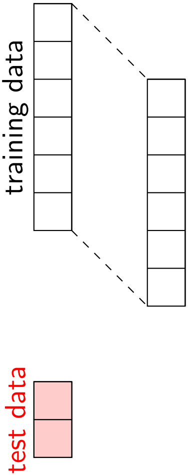


None of this helps us understand a how complicated our model should be. Unfortunately, **errors on our training data cannot tell us the answer**

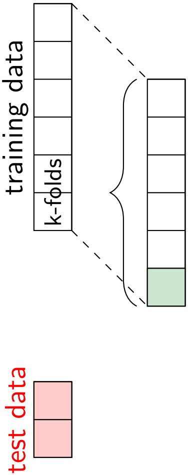
(Cross)-validation



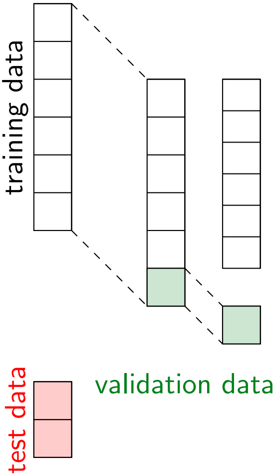
(Cross)-validation



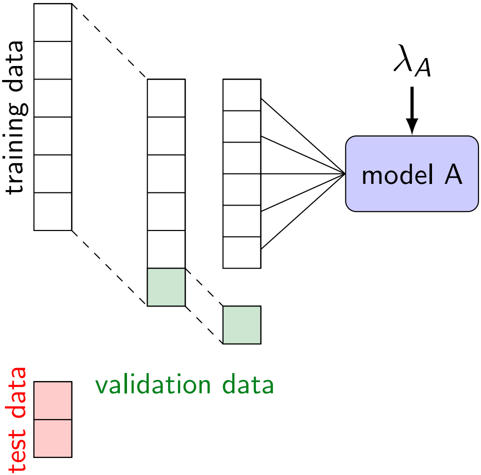
(Cross)-validation



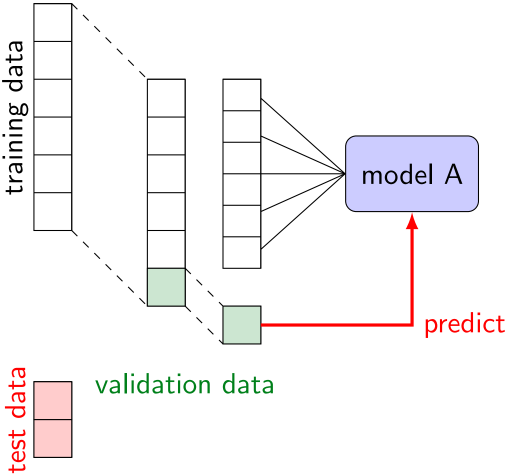
(Cross)-validation



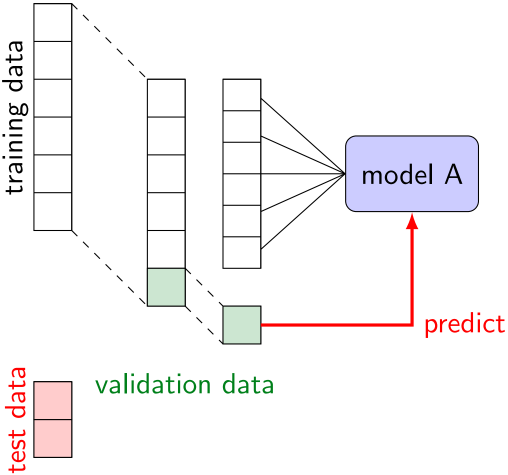
(Cross)-validation



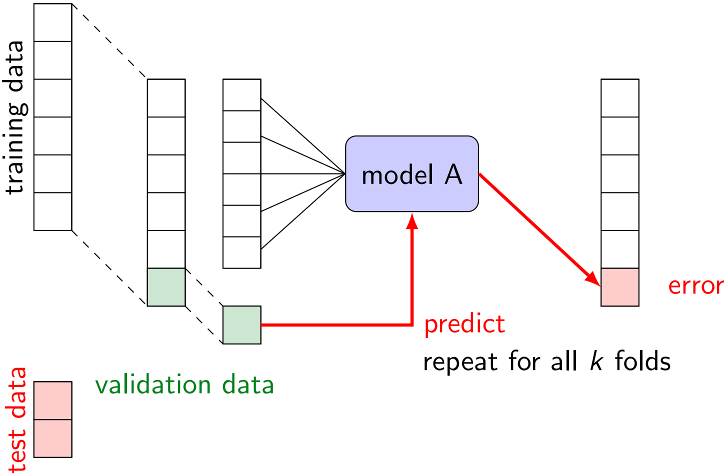
(Cross)-validation



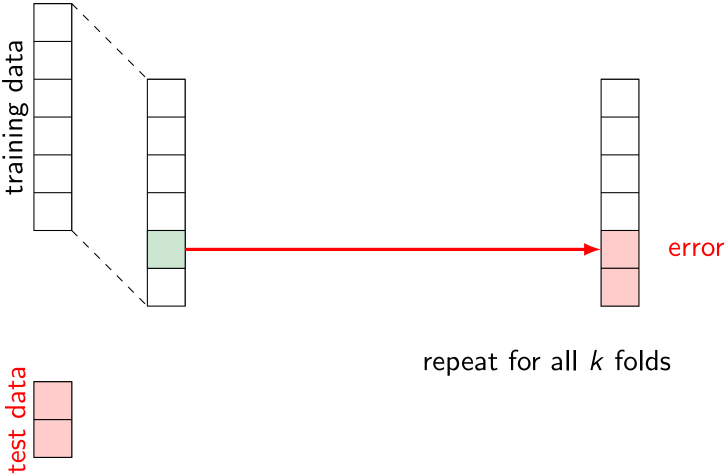
(Cross)-validation



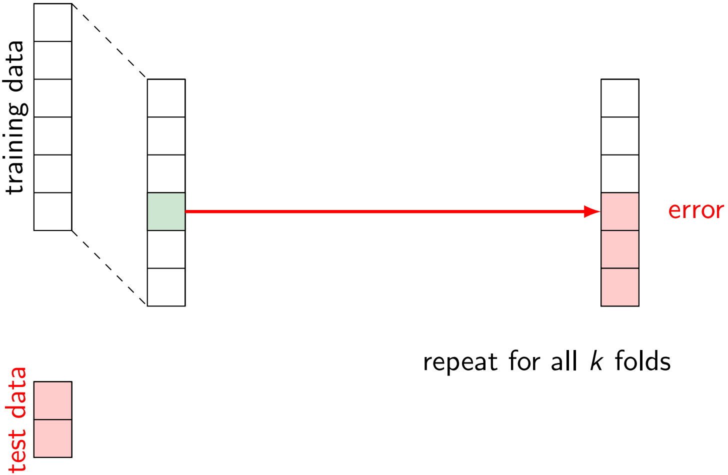
(Cross)-validation



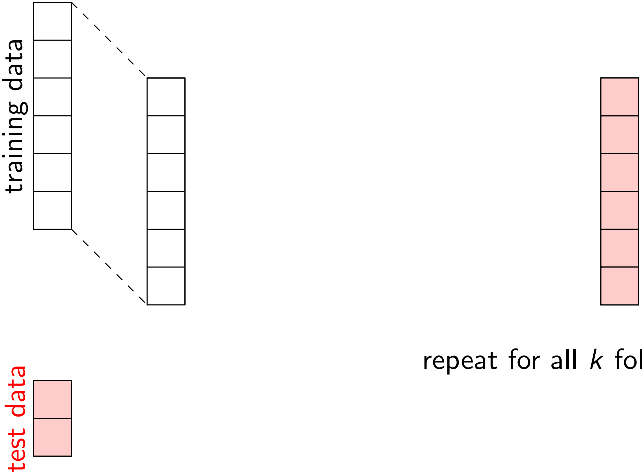
(Cross)-validation



(Cross)-validation

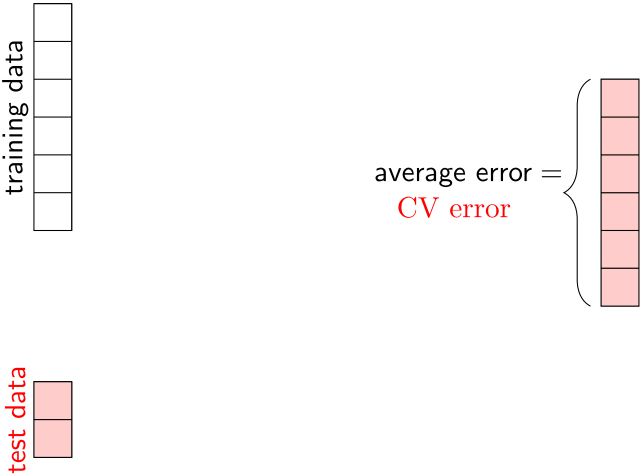


(Cross)-validation

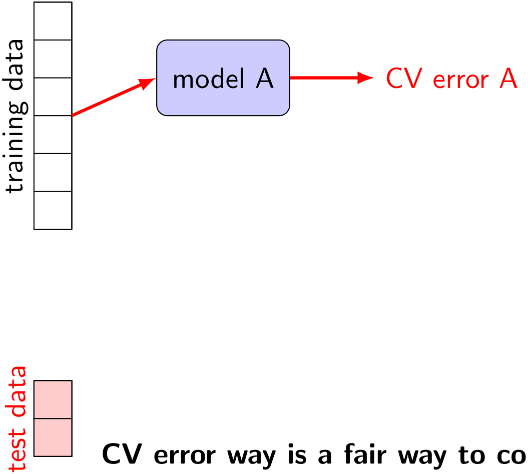


repeat for all k folds

(Cross)-validation

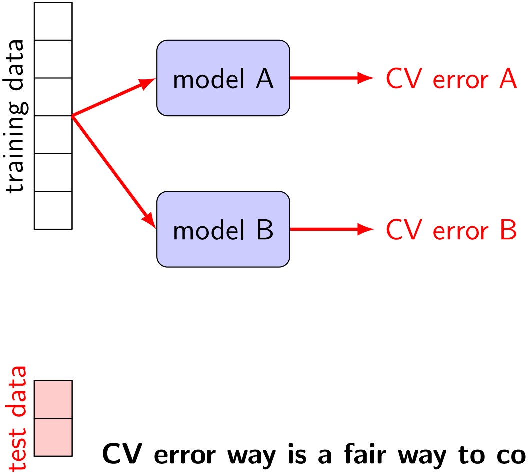


(Cross)-validation



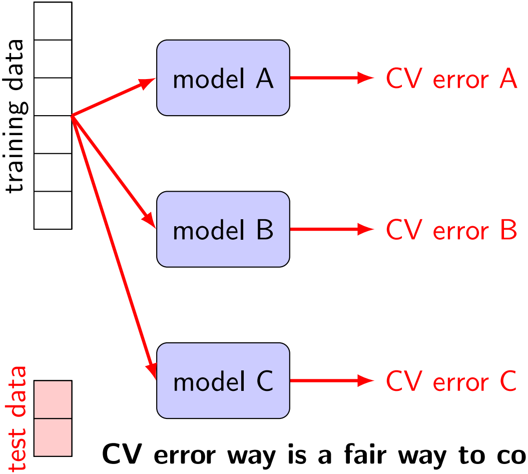
CV error way is a fair way to compare models

(Cross)-validation



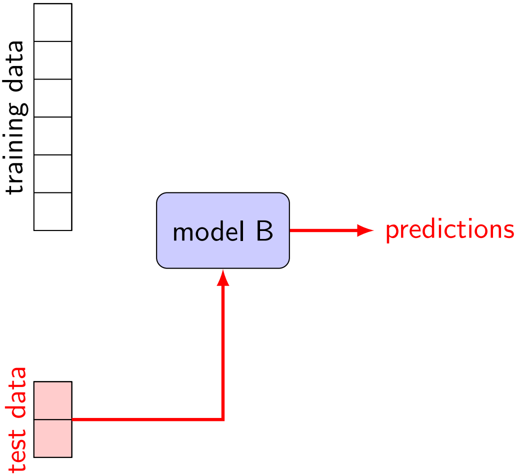
CV error way is a fair way to compare models

(Cross)-validation

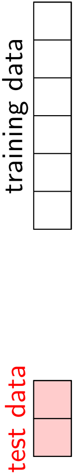


CV error way is a fair way to compare models

(Cross)-validation



(Cross)-validation



Conclusion

In summary:

Conclusion

In summary:

1. We train models by changing their parameters to reduce errors on training data

Conclusion

In summary:

1. We train models by changing their parameters to reduce errors on training data
2. More complicated models learn more slowly, but have more capacity

Conclusion

In summary:

1. We train models by changing their parameters to reduce errors on training data
2. More complicated models learn more slowly, but have more capacity
3. Regularization helps control complexity

Conclusion

In summary:

1. We train models by changing their parameters to reduce errors on training data
2. More complicated models learn more slowly, but have more capacity
3. Regularization helps control complexity
4. Cross-validation (and related techniques) must be used to choose hyperparameters

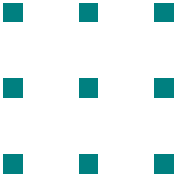
Conclusion

In summary:

1. We train models by changing their parameters to reduce errors on training data
2. More complicated models learn more slowly, but have more capacity
3. Regularization helps control complexity
4. Cross-validation (and related techniques) must be used to choose hyperparameters

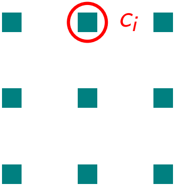
Deep neural networks (might) need better theories.

Purpose



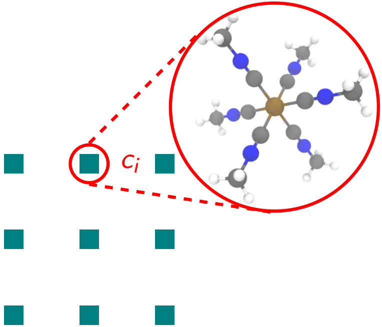
Chemical Space C_f

Purpose



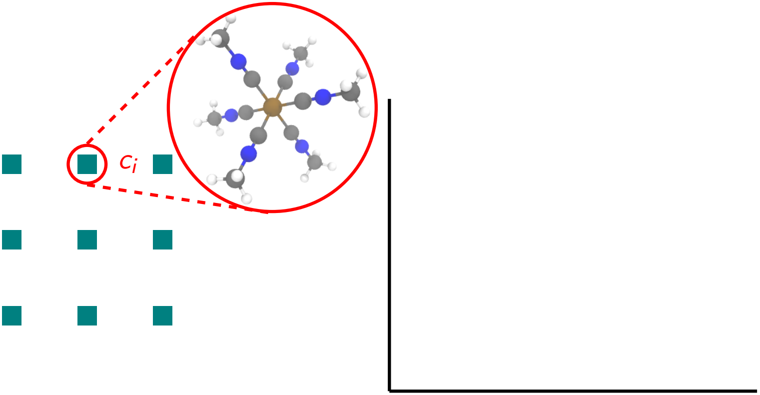
Chemical Space C_f

Purpose



Chemical Space C_f

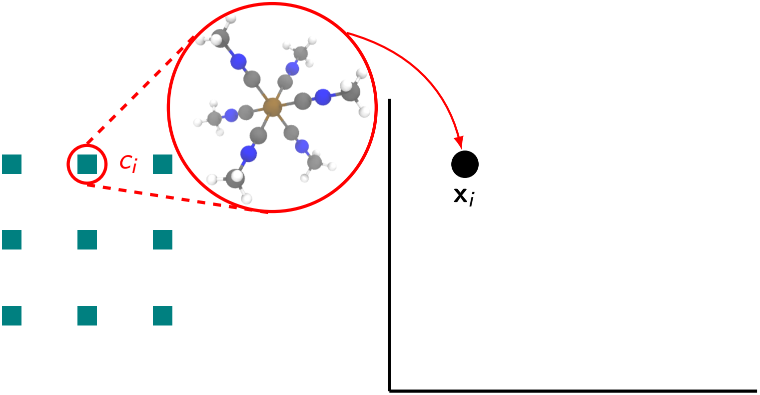
Purpose



Chemical Space C_f

Descriptor Space $\mathcal{X} \subset \mathbb{R}^d$

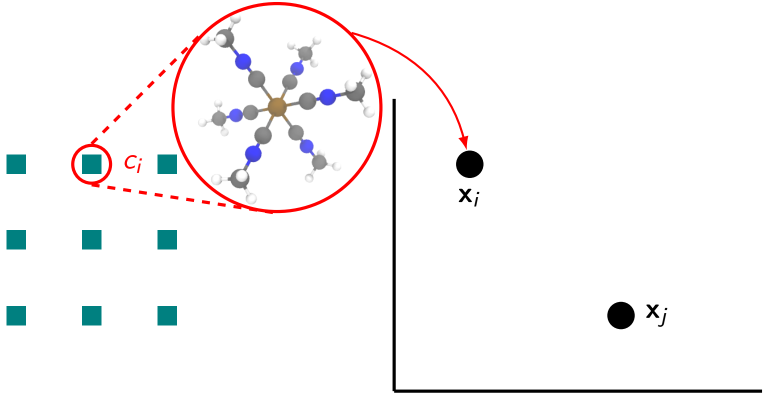
Purpose



Chemical Space C_f

Descriptor Space $\mathcal{X} \subset \mathbb{R}^d$

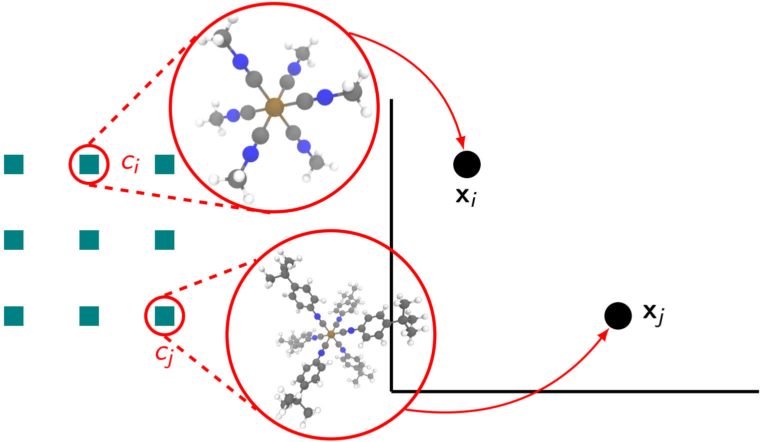
Purpose



Chemical Space C_f

Descriptor Space $\mathcal{X} \subset \mathbb{R}^d$

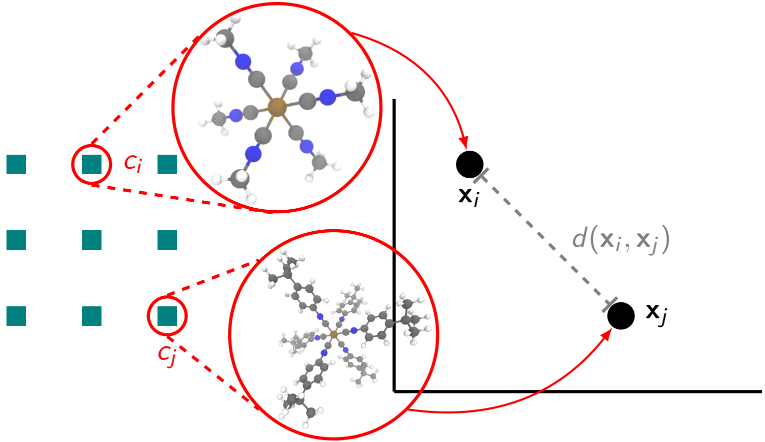
Purpose



Chemical Space C_f

Descriptor Space $\mathcal{X} \subset \mathbb{R}^d$

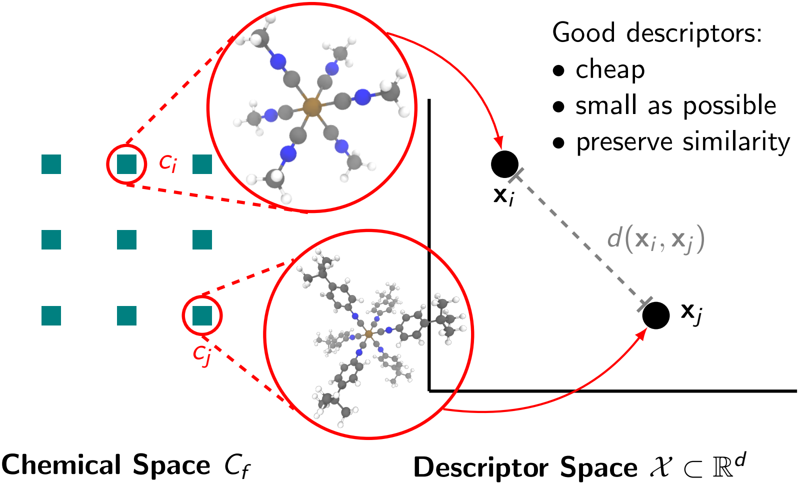
Purpose



Chemical Space C_f

Descriptor Space $\mathcal{X} \subset \mathbb{R}^d$

Purpose



Good descriptors:

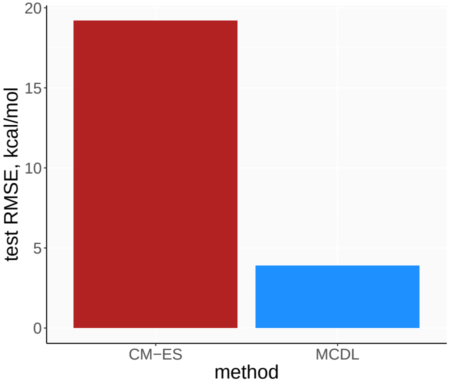
- cheap
- small as possible
- preserve similarity

Why similarity is important

Different representations can have very different performance, particularly if they do not preserve notions of chemical similarity correctly:

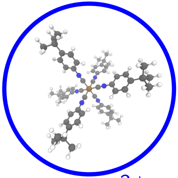
Why similarity is important

Different representations can have very different performance, particularly if they do not preserve notions of chemical similarity correctly:

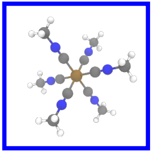


Janet, J.P., and Kulik, H.J. *Chem. Sci.*, 2017, 8, 5137-5152.

Why similarity is important

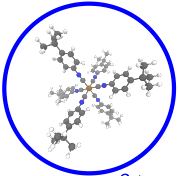


$Fe[pisc]_6^{3+}$



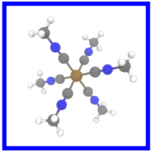
$Fe[misc]_6^{3+}$

Why similarity is important



$Fe[pisc]_6^{3+}$

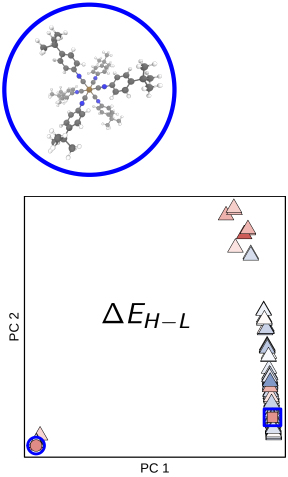
$\Delta E_{\text{H-L}} = 37.7$ kcal/mol

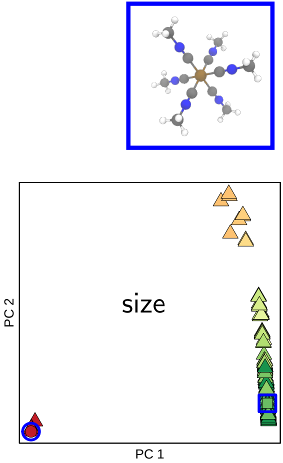


$Fe[misc]_6^{3+}$

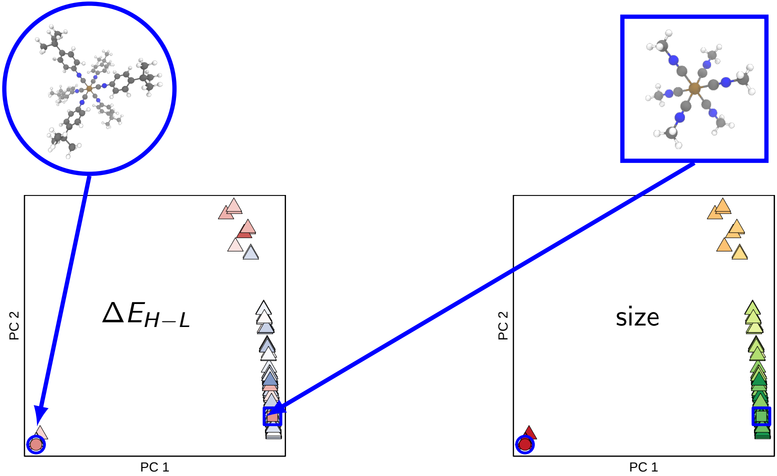
$\Delta E_{\text{H-L}} = 40.7$ kcal/mol

Why similarity is important

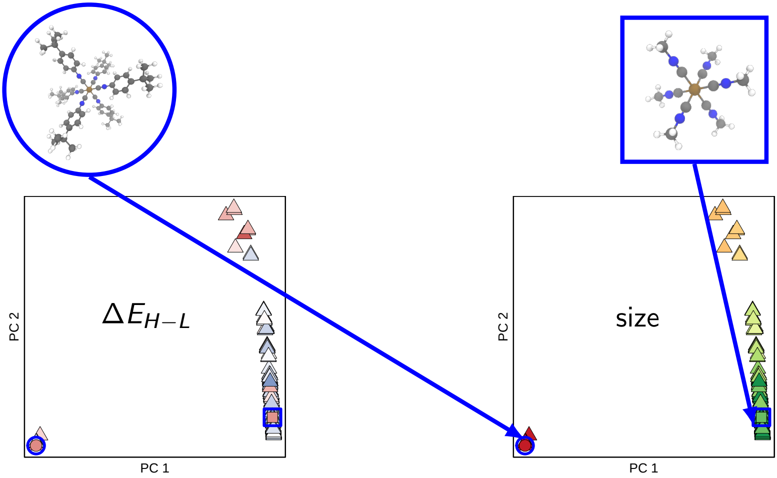




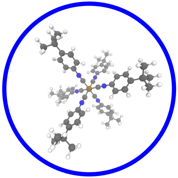
Why similarity is important

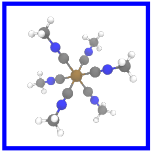


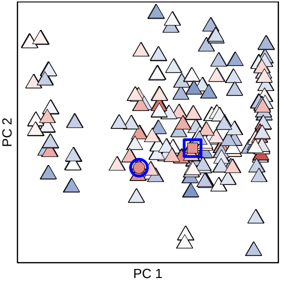
Why similarity is important

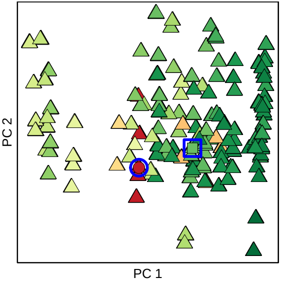


Why similarity is important







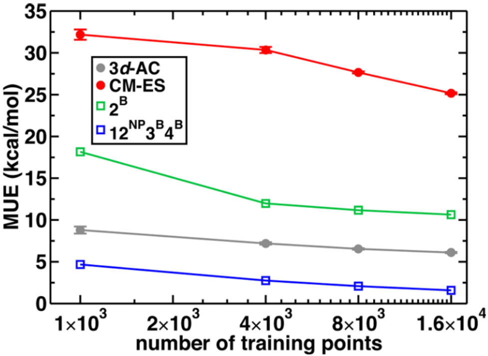


Why similarity is important

The richness of the representation also affects learning rate:

Why similarity is important

The richness of the representation also affects learning rate:



Types of representation

complexity

Types of representation

complexity

Fingerprints

– considerable use in drug design

– no information related to molecular topology

– cheap to compute

Types of representation

complexity

Fingerprints

– considerable use in drug design

– no information related to molecular topology

– cheap to compute

Graph-theoretic

– *topological* and *topochemical* representations

– includes all connectivity but no 3D information

– relatively easy to compute

Types of representation

complexity →

Fingerprints

– considerable use in drug design

– no information related to molecular topology

– cheap to compute

Graph-theoretic

– *topological* and *topochemical* representations

– includes all connectivity but no 3D information

– relatively easy to compute

3D structure

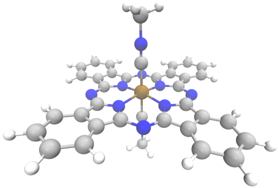
– fine-grained structural information in 3D

– mimic input to a quantum chemistry code

–expensive to compute, rich information

Ad-hoc properties

Sometimes, simple lists of atomic propties are sufficient, especially if informed by domain knowledge:



Ad-hoc properties

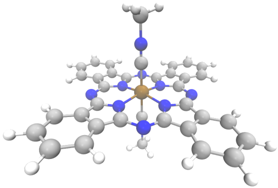
Sometimes, simple lists of atomic propties are sufficient, especially if informed by domain knowledge:

metal properties

identity

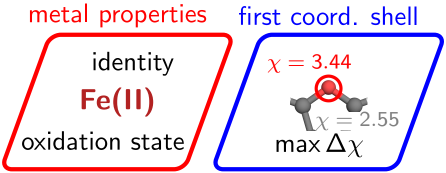
Fe(II)

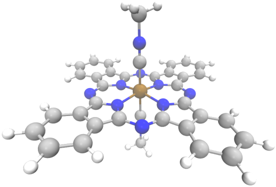
oxidation state



Ad-hoc properties

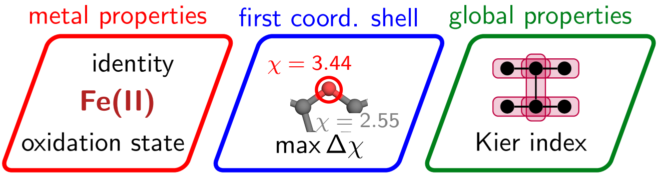
Sometimes, simple lists of atomic propties are sufficient, especially if informed by domain knowledge:

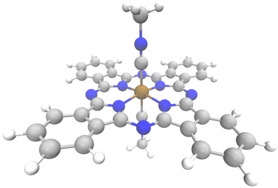




Ad-hoc properties

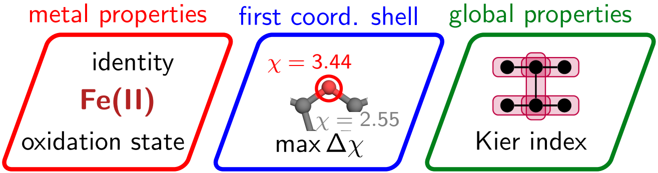
Sometimes, simple lists of atomic propties are sufficient, especially if informed by domain knowledge:

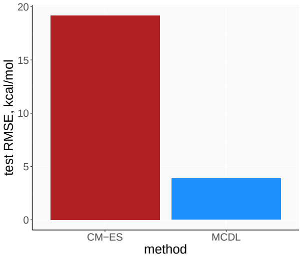


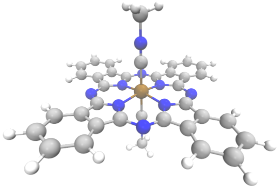


Ad-hoc properties

Sometimes, simple lists of atomic propties are sufficient, especially if informed by domain knowledge:







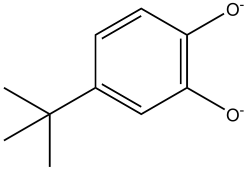
Janet, J.P., and Kulik, H.J. *Chem. Sci.*, 2017, 8, 5137-5152.

Fingerprints and the low-information limit

In cheminformatics (esp. drug design literature) fingerprints are binary vectors used to determine molecular similarity. For example, FP2 fingerprint is a 1024 bit fingerprint:

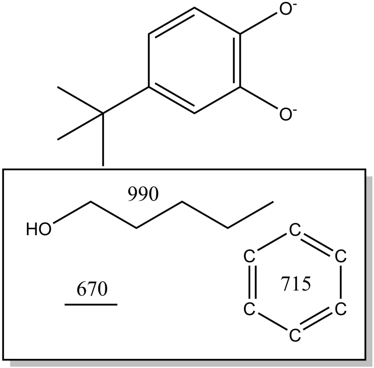
Fingerprints and the low-information limit

In cheminformatics (esp. drug design literature) fingerprints are binary vectors used to determine molecular similarity. For example, FP2 fingerprint is a 1024 bit fingerprint:



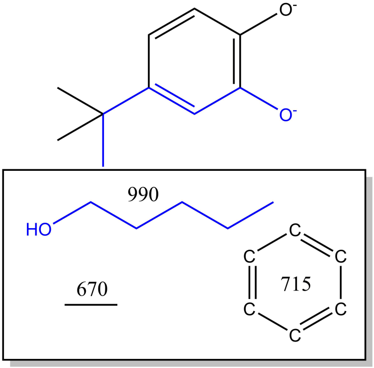
Fingerprints and the low-information limit

In cheminformatics (esp. drug design literature) fingerprints are binary vectors used to determine molecular similarity. For example, FP2 fingerprint is a 1024 bit fingerprint:



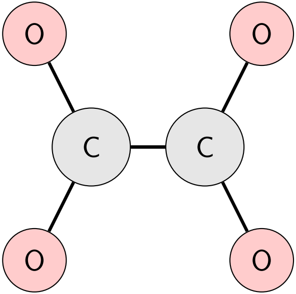
Fingerprints and the low-information limit

In cheminformatics (esp. drug design literature) fingerprints are binary vectors used to determine molecular similarity. For example, FP2 fingerprint is a 1024 bit fingerprint:



Molecular graphs

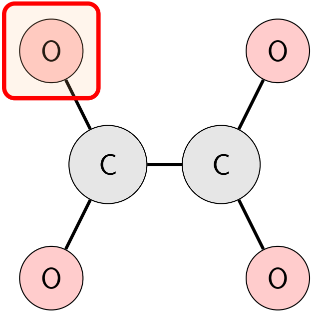
Based on autocorrelations[1]



[1] Broto, P., Moreau, G. and Vandycke, C. *Eur. J. Med. Chem.*, 19(1):71-78, 1984.

Molecular graphs

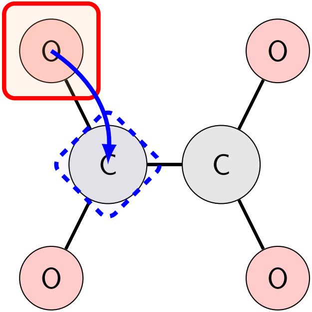
Based on autocorrelations and modified for TMCs[4]



[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

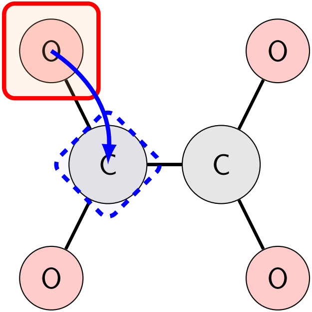
Based on autocorrelations and modified for TMCs[4]



[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]

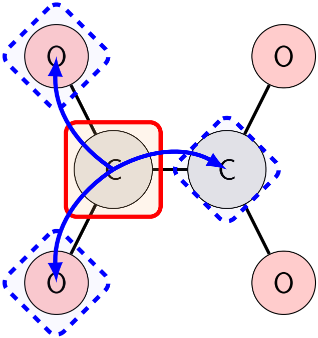


$$d_1 : \sum_{O,C} Z_O Z_C = 48$$

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]

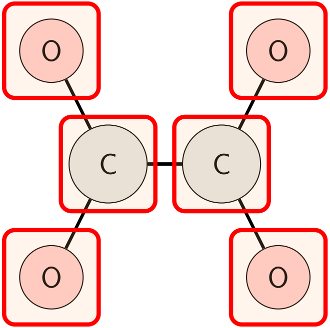


$$d_1 : 48 + \sum_{C,O} Z_O Z_C = 144 + 48$$

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]

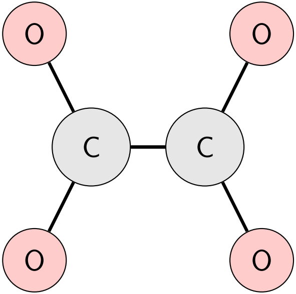


$$d_1 : \sum_i \sum_j Z_i Z_j \delta(d_{i,j}, 1)$$

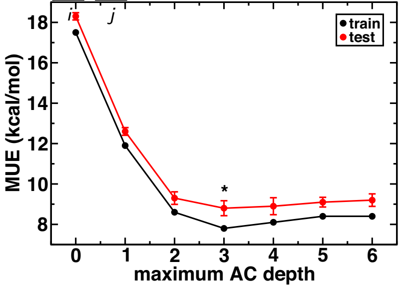
[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



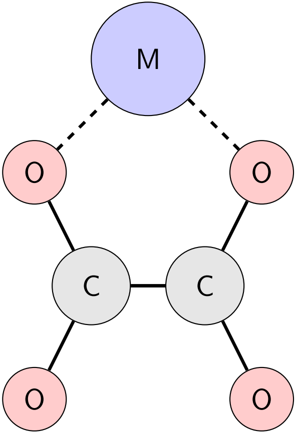
$$d_x : \sum_i \sum_j Z_i Z_j \delta(d_{ij}, x)$$



[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]

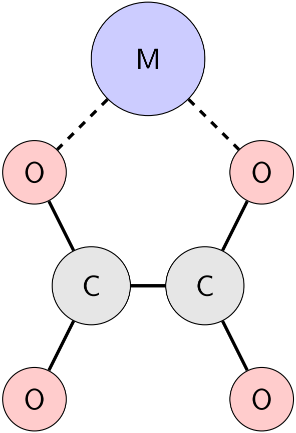


How to adapt to TM complexes?

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]

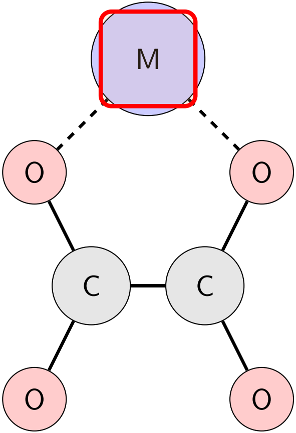


How to adapt to TM complexes?
restrict the scope to focus on
near-metal atoms

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]

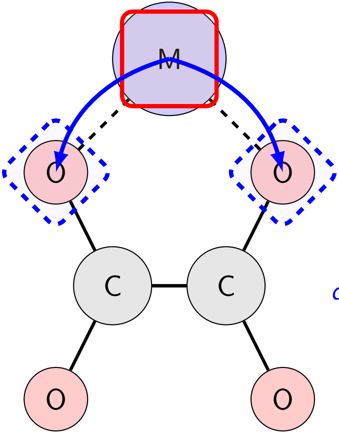


How to adapt to TM complexes?
restrict the scope to focus on
near-metal atoms

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



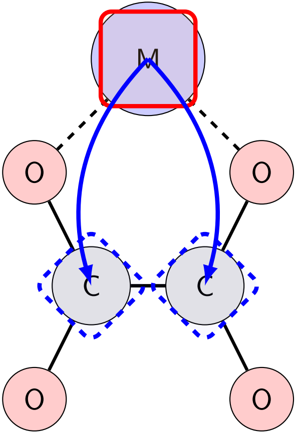
How to adapt to TM complexes?
restrict the scope to focus on
near-metal atoms

$$d_1 : \sum_{M,O} Z_M Z_O$$

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



How to adapt to TM complexes?

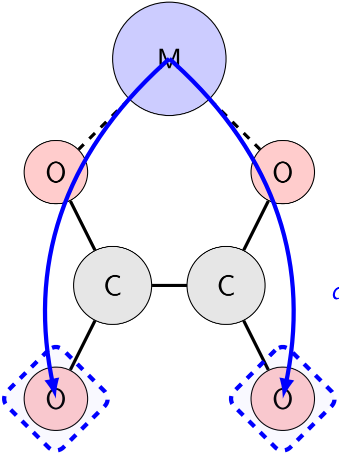
restrict the scope to focus on *near-metal atoms*

$$d_2 : \sum_{M,C} Z_M Z_C$$

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



How to adapt to TM complexes?

restrict the scope to focus on

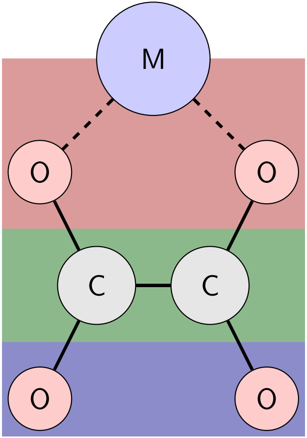
near-metal atoms

$$d_3 : \sum_{M,O} Z_M Z_O$$

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



How to adapt to TM complexes?

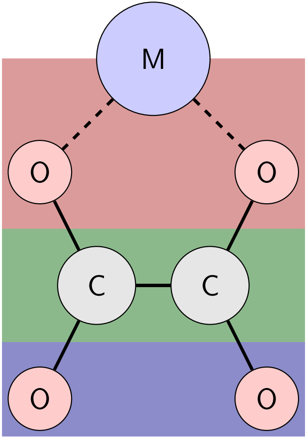
restrict the scope to focus on *near-metal atoms*

$$d_3 : \sum_{M,O} Z_M Z_O$$

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



How to adapt to TM complexes?
restrict the scope to focus on
near-metal atoms

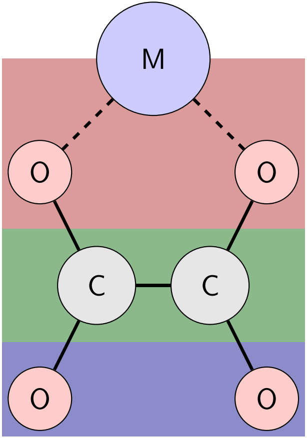
$$d_3 : \sum_{M,O} Z_M Z_O$$

$$(Z_i - Z_j)$$

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



How to adapt to TM complexes?
restrict the scope to focus on
near-metal atoms

$$d_3 : \sum_{M,O} Z_M Z_O$$

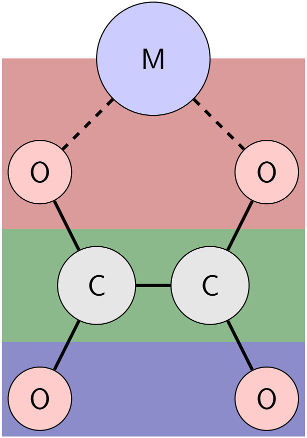
$$(Z_i - Z_j)$$

properties: T, χ, Z, I, S

[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

Molecular graphs

Based on autocorrelations and modified for TMCs[4]



How to adapt to TM complexes?
restrict the scope to focus on
near-metal atoms

$$d_3 : \sum_{M,O} Z_M Z_O$$

$$(Z_i - Z_j)$$

$\boxed{\sim 160 \text{ features in total}}$

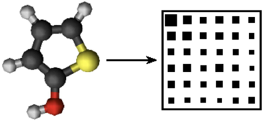
[4] Janet, J.P., and Kulik, H.J. *J. Phys. Chem. A*, 2017,121, 46, 8939-8954.

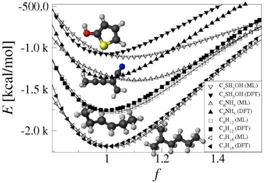
Coulomb matrices

One family of 3D desciptors attempt to copy information used in quantum chemistry codes, e.g. Coulomb Matricies:

Montavon, G. *et al.*. Learning Invariant Representations of Molecules for Atomization Energy Prediction, NIPS 25, 2012

$$M_{I,J} = \begin{cases} 0.5 Z_I^{2.4} & \text{for } I = J \\ \frac{Z_I Z_J}{|R_I - R_J|} & \text{for } I \neq J \end{cases}$$



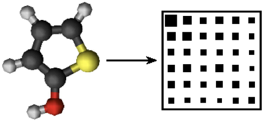


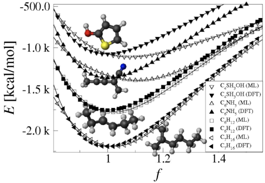
Coulomb matrices

One family of 3D desciptors attempt to copy information used in quantum chemistry codes, e.g. Coulomb Matricies:

Montavon, G. *et al.*. Learning Invariant Representations of Molecules for Atomization Energy Prediction, NIPS 25, 2012

$$M_{I,J} = \begin{cases} 0.5 Z_I^{2.4} & \text{for } I = J \\ \frac{Z_I Z_J}{|R_I - R_J|} & \text{for } I \neq J \end{cases}$$



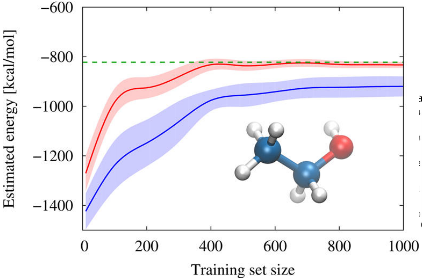


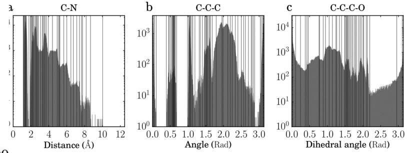
rotational and translational invariance

HDAD and beyond-CM

Subsequent work adds descriptors derived from geometric parameters, i.e. bonds, angles, and dihedral angles:

Faber, F. *et al.*. Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error, *J. Chem. Theory Comput.* 2017, 13, 11, 5255-5264





System and atom level features

molecule-level

- one vector for each system of interest
- commonly used in QSAR/QSPR, related to how we think about molecules
- easy to compare whole molecules
- some properties are really 'global', like logP

System and atom level features

molecule-level

- one vector for each system of interest
- commonly used in QSAR/QSPR, related to how we think about molecules
- easy to compare whole molecules
- some properties are really 'global', like logP

atom-level

- one vector for each **atom** in the system of interest

System and atom level features

molecule-level

- one vector for each system of interest
- commonly used in QSAR/QSPR, related to how we think about molecules
- easy to compare whole molecules
- some properties are really 'global', like logP

atom-level

- one vector for each **atom** in the system of interest
- commonly used in NNPs, most accurate results

System and atom level features

molecule-level

- one vector for each system of interest
- commonly used in QSAR/QSPR, related to how we think about molecules
- easy to compare whole molecules
- some properties are really 'global', like logP

atom-level

- one vector for each **atom** in the system of interest
- commonly used in NNPs, most accurate results
- energy can be naturally decomposed into atomic contributions

System and atom level features

molecule-level

- one vector for each system of interest
- commonly used in QSAR/QSPR, related to how we think about molecules
- easy to compare whole molecules
- some properties are really 'global', like logP

atom-level

- one vector for each **atom** in the system of interest
- commonly used in NNPs, most accurate results
- energy can be naturally decomposed into atomic contributions
- enormous scaling advantages

System and atom level features

molecule-level

- one vector for each system of interest
- commonly used in QSAR/QSPR, related to how we think about molecules
- easy to compare whole molecules
- some properties are really 'global', like logP

atom-level

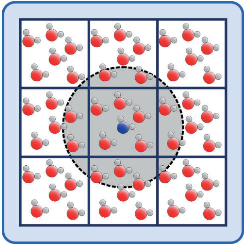
- one vector for each **atom** in the system of interest
- commonly used in NNPs, most accurate results
- energy can be naturally decomposed into atomic contributions
- enormous scaling advantages

System and atom level features

Basic idea is to create an atomic level represenstation that 'knows' about the neighborhood:
J. Behler. First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems, *Angew. Chem. Int. Ed.*, 56, 12828, 2017

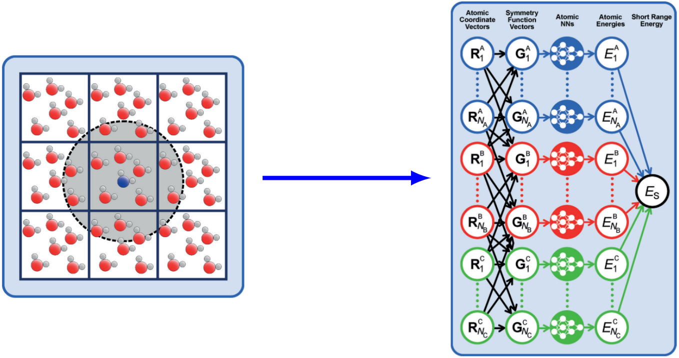
System and atom level features

Basic idea is to create an atomic level represenstation that 'knows' about the neighborhood:
J. Behler. First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems, *Angew. Chem. Int. Ed.*, 56, 12828, 2017



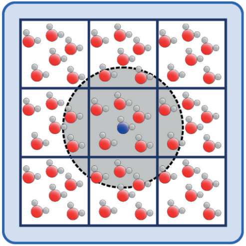
System and atom level features

Basic idea is to create an atomic level representation that 'knows' about the neighborhood:
J. Behler. First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems, *Angew. Chem. Int. Ed.*, 56, 12828, 2017

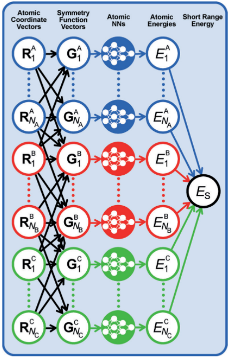


System and atom level features

Basic idea is to create an atomic level representstation that 'knows' about the neighborhood:
J. Behler. First Principles Neural Network Potentials for Reactive Simulations of Large Molecular and Condensed Systems, *Angew. Chem. Int. Ed.*, 56, 12828, 2017



J. Behler and M. Parrinello. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces, *Phys. Rev. Lett.*, 98, 146401, 2007



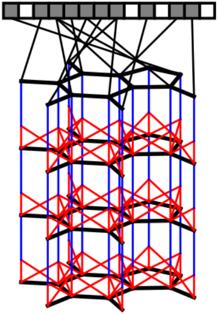
Learning representations

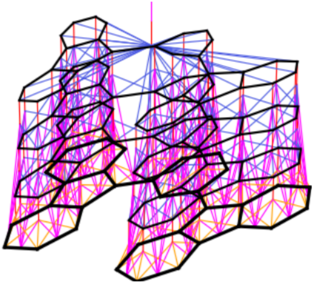
We can directly learn what features to have using NNs – graph convolutions (more on this in NN section)

Duvenaud, D. *et al.*.Convolutional Networks on Graphs for Learning Molecular Fingerprints, NIPS 28, 2015

Learning representations

We can directly learn what features to have using NNs – graph convolutions (more on this in NN section)
Duvenaud, D. *et al.*.Convolutional Networks on Graphs for Learning Molecular Fingerprints, NIPS 28, 2015

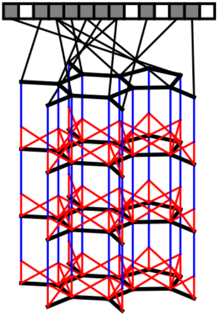


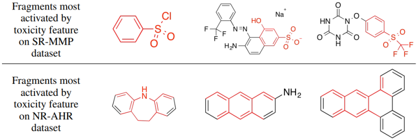


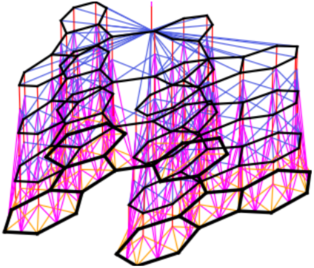
Learning representations

We can directly learn what features to have using NNs – graph convolutions (more on this in NN section)

Duvenaud, D. *et al.*.Convolutional Networks on Graphs for Learning Molecular Fingerprints, NIPS 28, 2015







Conclusions

In summary:

Conclusions

In summary:

1. Different encoding schemes use different amounts of information

Conclusions

In summary:

1. Different encoding schemes use different amounts of information
2. The representation needs to matched to the application in question

Conclusions

In summary:

1. Different encoding schemes use different amounts of information
2. The representation needs to matched to the application in question
3. Representations should encode our notions of chemical similarity

Conclusions

In summary:

1. Different encoding schemes use different amounts of information
2. The representation needs to matched to the application in question
3. Representations should encode our notions of chemical similarity
4. Complicated, high dimensional featurization schemes require more data

Conclusions

In summary:

1. Different encoding schemes use different amounts of information
2. The representation needs to matched to the application in question
3. Representations should encode our notions of chemical similarity
4. Complicated, high dimensional featurization schemes require more data
5. atom-level featurization can be very effective for total energies

Multiple linear regression

Multiple linear regression

Linear models give $\hat{y}$ as linear function of the data matrix of X:

$$\hat{y}_{MLR}(x^*) = \sum_{j=1}^{d} w_j x_j^* + w_0$$

Multiple linear regression

Linear models give $\hat{y}$ as linear function of the data matrix of X:

$$\hat{y}_{MLR}(x^*) = \sum_{j=1}^{d} w_j x_j^* + w_0 = \begin{bmatrix} 1 & x_1^* & \dots & x_d^* \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

Multiple linear regression

Linear models give $\hat{y}$ as linear function of the data matrix of X:

$$\hat{y}_{MLR}(x^*) = \sum_{j=1}^{d} w_j x_j^* + w_0 = \begin{bmatrix} 1 & x_1^* & \dots & x_d^* \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

We can write this in a matrix form as well:

$$\hat{y}_{MLR}(X) = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots x_d^{(1)} \\ \vdots & & & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \dots x_d^{(n)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = Xw$$

Multiple linear regression

Linear models give $\hat{y}$ as linear function of the data matrix of X:

$$\hat{y}_{MLR}(x^*) = \sum_{j=1}^{d} w_j x_j^* + w_0 = \begin{bmatrix} 1 & x_1^* & \dots & x_d^* \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

We can write this in a matrix form as well:

$$\hat{y}_{MLR}(X) = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots x_d^{(1)} \\ \vdots & & & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \dots x_d^{(n)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = Xw$$

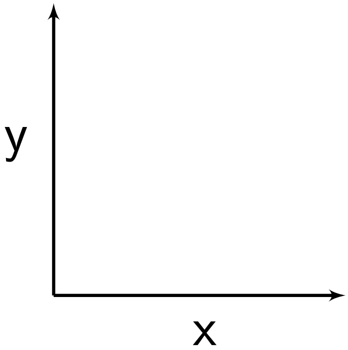
Notice how we handle the constant terms

Multiple linear regression II

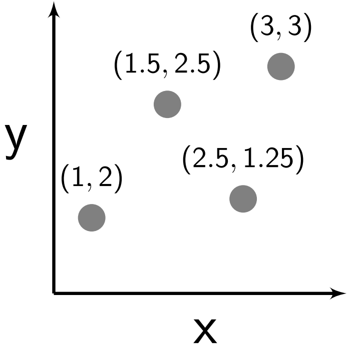
Let's solve our regularized least-squares problem:

$$
\begin{aligned}
w &= \arg\min_{w\in\mathbb{R}^p} \frac{1}{n} \left\| y_{data} - Xw \right\|_2^2 + +\lambda \left\| w \right\|_2^2 \\
&= \frac{1}{n} \left(y_{data} - Xw \right)^T \left(y_{data} - Xw \right) + \lambda w^T w \\
\frac{\partial \mathcal{L}}{\partial w} &= -\frac{2}{n} X^T \left(y_{data} - Xw \right) + 2\lambda w = 0 \\
\implies (\lambda I + X^T X) w &= X^T y_{data} \\
w &= \left(\lambda I + X^T X \right)^{-1} X^T y_{data}
\end{aligned}
$$

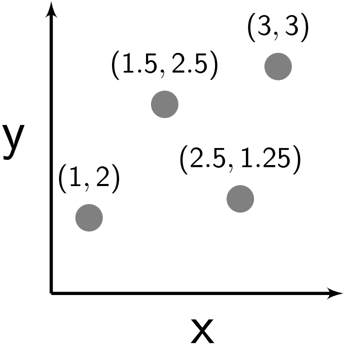
Simple example in 1D



Simple example in 1D

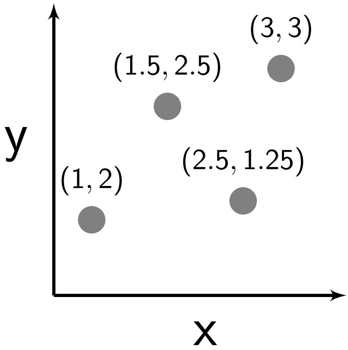


Simple example in 1D



$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2.5 \\ 1 & 3.0 \end{bmatrix}, \; y_{data} = \begin{bmatrix} 2 \\ 2.5 \\ 1.25 \\ 3 \end{bmatrix}$$

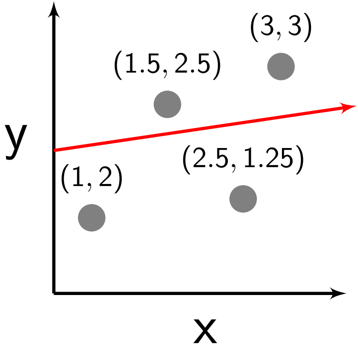
Simple example in 1D



$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2.5 \\ 1 & 3.0 \end{bmatrix}, \; y_{data} = \begin{bmatrix} 2 \\ 2.5 \\ 1.25 \\ 3 \end{bmatrix}$$

$$w = \left(X^T X + \lambda I\right)^{-1} X^T y_{data}$$

Simple example in 1D

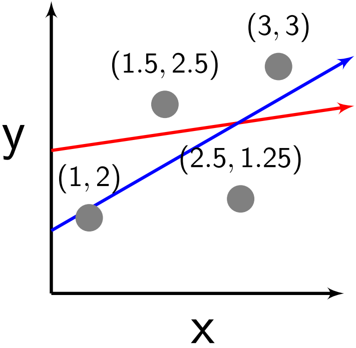


$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2.5 \\ 1 & 3.0 \end{bmatrix}, \; y_{data} = \begin{bmatrix} 2 \\ 2.5 \\ 1.25 \\ 3 \end{bmatrix}$$

$$w = \left(X^T X + \lambda I\right)^{-1} X^T y_{data}$$

$$= \begin{bmatrix} 1.9 \\ 0.15 \end{bmatrix} (\lambda = 0.0)$$

Simple example in 1D

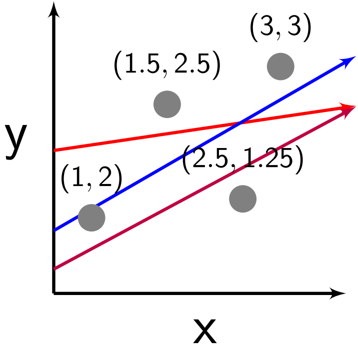


$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2.5 \\ 1 & 3.0 \end{bmatrix}, \ y_{data} = \begin{bmatrix} 2 \\ 2.5 \\ 1.25 \\ 3 \end{bmatrix}$$

$$w = \left(X^T X + \lambda I\right)^{-1} X^T y_{data}$$

$$= \begin{bmatrix} 0.82 \\ 0.58 \end{bmatrix} (\lambda = 1.0)$$

Simple example in 1D



$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2.5 \\ 1 & 3.0 \end{bmatrix}, \; y_{data} = \begin{bmatrix} 2 \\ 2.5 \\ 1.25 \\ 3 \end{bmatrix}$$

$$w = \left(X^T X + \lambda I\right)^{-1} X^T y_{data}$$

$$= \begin{bmatrix} 0.32 \\ 0.54 \end{bmatrix} (\lambda = 10)$$

The linear kernel

We can rewrite our result to express $w = X^T a$ for $a \in \mathbb{R}^n$ (shift of basis).

$$\hat{y}_{MLR}(x^*) = x^* w = \sum_{j=1}^{d} x_j^* w_j$$

The linear kernel

We can rewrite our result to express $w = X^T a$ for $a \in \mathbb{R}^n$ (shift of basis).

$$\hat{y}_{MLR}(x^*) = x^* w = \sum_{j=1}^{d} x_j^* w_j$$

The linear kernel

We can rewrite our result to express $w = X^T a$ for $a \in \mathbb{R}^n$ (shift of basis).

$$\hat{y}_{MLR}(x^*) = x^* w = \sum_{j=1}^{d} x_j^* w_j$$

$$= x^* X^T a = \begin{bmatrix} x_1^* & \dots & x_d^* \end{bmatrix} \begin{bmatrix} x_1^{(1)} & \dots x_1^{(n)} \\ \vdots & \vdots \\ x_d^{(1)} & \dots x_d^{(n)} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

The linear kernel

We can rewrite our result to express $w = X^T a$ for $a \in \mathbb{R}^n$ (shift of basis).

$$\begin{aligned}
\hat{y}_{MLR}(x^*) &= x^* w = \sum_{j=1}^{d} x_j^* w_j \\
&= x^* X^T a = \begin{bmatrix} x_1^* & \dots & x_d^* \end{bmatrix} \begin{bmatrix} x_1^{(1)} & \dots x_1^{(n)} \\ \vdots & \vdots \\ x_d^{(1)} & \dots x_d^{(n)} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \\
&= \sum_{i=1}^{n} x^* x_i^T a_i
\end{aligned}$$

The linear kernel

We can rewrite our result to express $w = X^T a$ for $a \in \mathbb{R}^n$ (shift of basis).

$$\hat{y}_{MLR}(x^*) = x^* w = \sum_{j=1}^{d} x_j^* w_j$$

$$= x^* X^T a = \begin{bmatrix} x_1^* & \dots & x_d^* \end{bmatrix} \begin{bmatrix} x_1^{(1)} & \dots x_1^{(n)} \\ \vdots & \vdots \\ x_d^{(1)} & \dots x_d^{(n)} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

$$= \sum_{i=1}^{n} x^* x_i^T a_i = \sum_{i=1}^{n} k\left(x^*, x_i\right) a_i$$

The linear kernel

We can rewrite our result to express $w = X^T a$ for $a \in \mathbb{R}^n$ (shift of basis).

$$\hat{y}_{MLR}(x^*) = x^* w = \sum_{j=1}^{d} x_j^* w_j$$

$$= x^* X^T a = \begin{bmatrix} x_1^* & \dots & x_d^* \end{bmatrix} \begin{bmatrix} x_1^{(1)} & \dots x_1^{(n)} \\ \vdots & \vdots \\ x_d^{(1)} & \dots x_d^{(n)} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

$$= \sum_{i=1}^{n} x^* x_i^T a_i = \sum_{i=1}^{n} k(x^*, x_i)\, a_i$$

The term $k(x^*, x_i) = x^* x_i^T = \langle x^*, x_i \rangle$is the **linear kernel**.

The linear kernel II

The matrix $K_{i,j} = \langle x_i, x_j \rangle$ is called the (linear) **kernel matrix**.
We can write the solution of the regression problem in this form – it is **exactly equivalent**:

$$\hat{y}(X) = Ka$$
$$a = (K + I_n\lambda)^{-1}y$$

The linear kernel II

The matrix $K_{i,j} = \langle x_i, x_j \rangle$ is called the (linear) **kernel matrix**.
We can write the solution of the regression problem in this form – it is **exactly equivalent**:

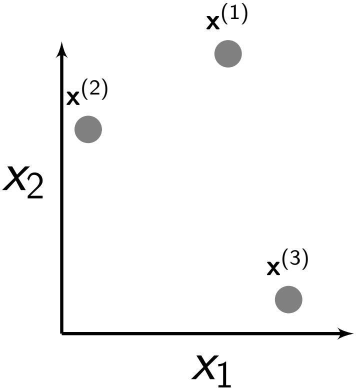
$$\hat{y}(X) = Ka$$
$$a = (K + I_n \lambda)^{-1} y$$

The prediction at any new point is proportional to the inner product of each training point and the new point:

$$\hat{y}_{MLR}(x^*) = \sum_{i=1}^{n} k(x^*, x_i)\, a_i = \sum_{i=1}^{n} k \langle x^*, x_i \rangle\, a_i$$

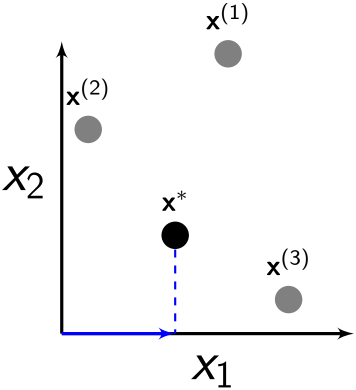
Linear kernel geometric picture

$$y(x^*) = w_1 x_1^* + w_2 x_2^*$$



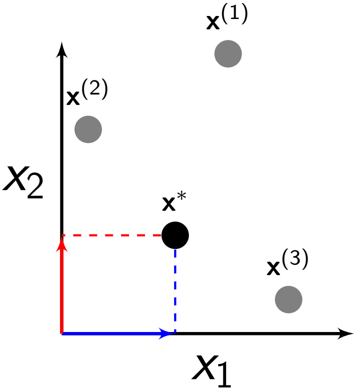
Linear kernel geometric picture

$$y(x^*) = w_1 x_1^* + w_2 x_2^*$$



Linear kernel geometric picture

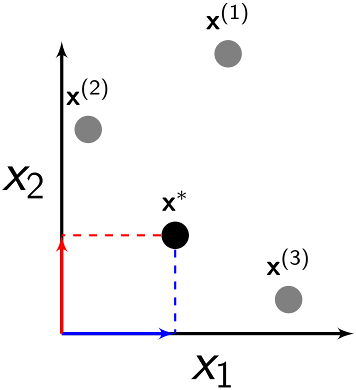
$$y(x^*) = w_1 x_1^* + w_2 x_2^*$$

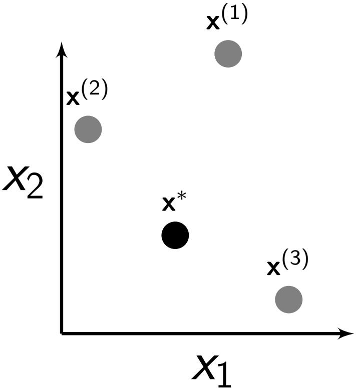


Linear kernel geometric picture

$y(x^*) = w_1 x_1^* + w_2 x_2^*$

$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$

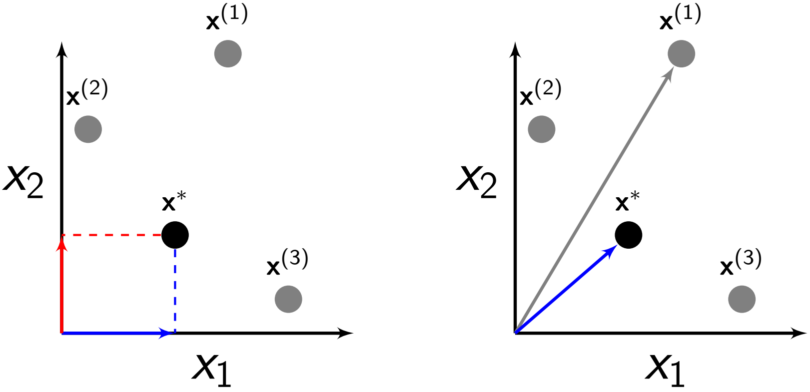




Linear kernel geometric picture

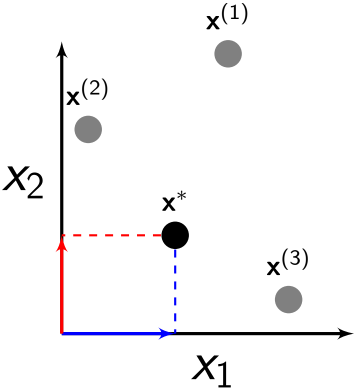
$y(x^*) = w_1 x_1^* + w_2 x_2^*$

$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$

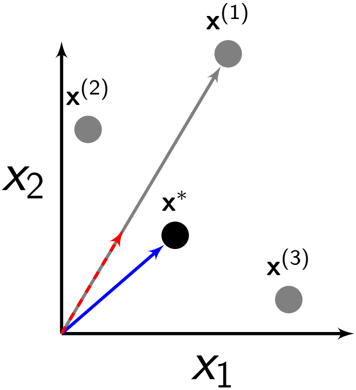


Linear kernel geometric picture

$$y(x^*) = w_1 x_1^* + w_2 x_2^*$$

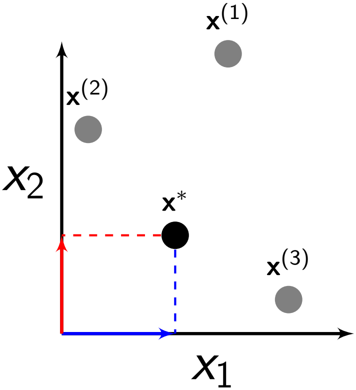


$$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$$

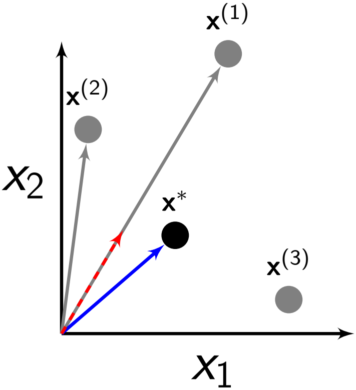


Linear kernel geometric picture

$y(x^*) = w_1 x_1^* + w_2 x_2^*$

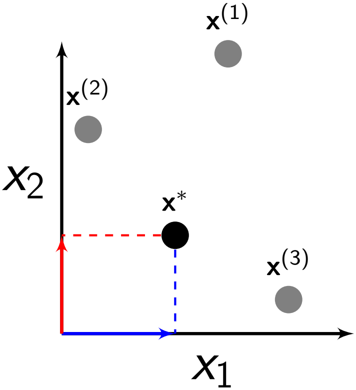


$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$

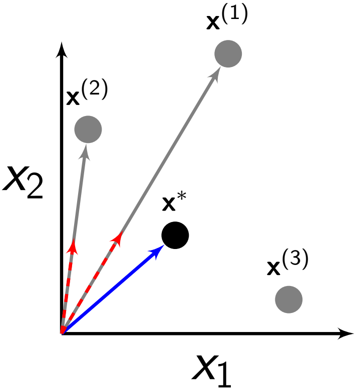


Linear kernel geometric picture

$y(x^*) = w_1 x_1^* + w_2 x_2^*$

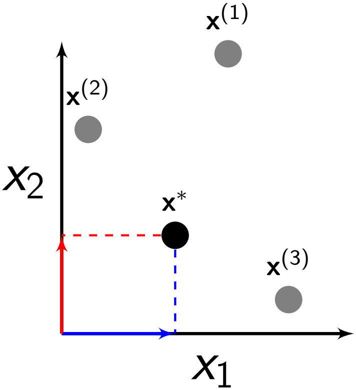


$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$

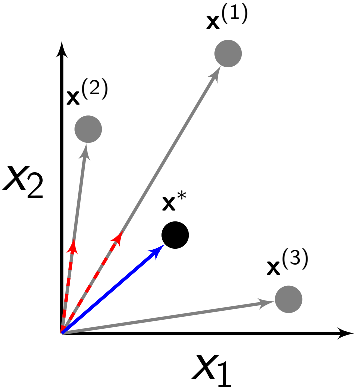


Linear kernel geometric picture

$$y(x^*) = w_1 x_1^* + w_2 x_2^*$$

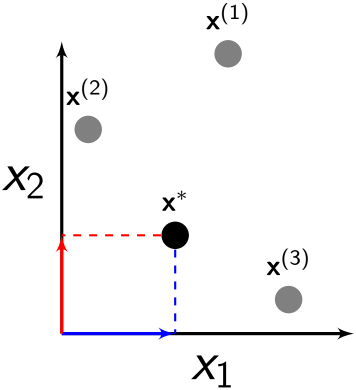


$$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$$

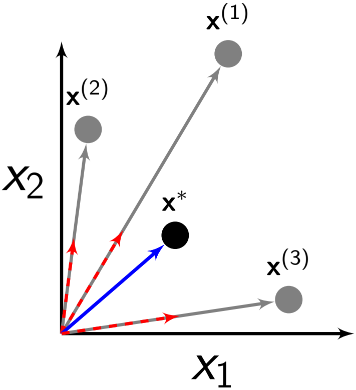


Linear kernel geometric picture

$y(x^*) = w_1 x_1^* + w_2 x_2^*$



$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$



Nonlinear regression

What about non-linear regression?

Nonlinear regression

What about non-linear regression? Let's say we want to use a quadratic model and assume a 2D input variable:

$$y_{QUAD}(x) = w_1 + w_2 x_1 + w_3 x_2 + w_4 x_1 x_2 + w_5 x_1^2 + w_6 x_2^2$$

Nonlinear regression

What about non-linear regression? Let's say we want to use a quadratic model and assume a 2D input variable:

$$y_{QUAD}(x) = w_1 + w_2\sqrt{2}x_1 + w_3\sqrt{2}x_2 + w_4\sqrt{2}x_1x_2 + w_5x_1^2 + w_6x_2^2$$

Nonlinear regression

What about non-linear regression? Let's say we want to use a quadratic model and assume a 2D input variable:

$$y_{QUAD}(x) = w_1 + w_2\sqrt{2}x_1 + w_3\sqrt{2}x_2 + w_4\sqrt{2}x_1x_2 + w_5x_1^2 + w_6x_2^2$$

Notice that this is *linear* in w for a 'lifted' feature space, $\varphi(X)$:

$$\begin{aligned} y_{QUAD}(x) &= \varphi(X)w \\ &= \begin{bmatrix} 1 & \sqrt{2}x_1^{(1)} & \sqrt{2}x_2^{(1)} & \sqrt{2}x_1^{(1)}x_2^{(1)} & (x_1^{(1)})^2 & (x_2^{(1)})^2 \\ \vdots & & \vdots & & & \\ 1 & \sqrt{2}x_1^{(n)} & \sqrt{2}x_2^{(n)} & \sqrt{2}x_1^{(1)}x_2^{(n)} & (x_1^{(n)})^2 & (x_2^{(n)})^2 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_6 \end{bmatrix} \end{aligned}$$

except the dimension has increased from $\mathbb{R}^{n\times 2} \rightarrow \mathbb{R}^{n\times 6}$

Nonlinear regression II

We can solve these equations exactly as before:

$$y_{QUAD}(x) = \varphi(X)w$$

Nonlinear regression II

We can solve these equations exactly as before:

$$y_{QUAD}(x) = \varphi(X)w$$
$$\implies w = (\varphi(X)^T\varphi(X) + I_{d'}\lambda)^{-1}\varphi(X)^T y_{data}$$

Nonlinear regression II

We can solve these equations exactly as before:

$$
\begin{aligned}
y_{QUAD}(x) &= \varphi(X)w \\
\implies w &= (\varphi(X)^T\varphi(X) + I_{d'}\lambda)^{-1}\varphi(X)^T y_{data}
\end{aligned}
$$

(c.f. the linear case) $w = (X^T X + I_d\lambda)^{-1}X^T y_{data}$

Nonlinear regression II

We can solve these equations exactly as before:

$$
\begin{aligned}
y_{QUAD}(x) &= \varphi(X)w \\
&\implies w = (\varphi(X)^T\varphi(X) + I_{d'}\lambda)^{-1}\varphi(X)^T y_{data}
\end{aligned}
$$

$$\text{(c.f. the linear case)} \;\; w = (X^TX + I_d\lambda)^{-1}X^T y_{data}$$

by direct analogy to the previous slides, there is also a kernel form:

$$
\begin{aligned}
\hat{y}(x^*) &= \sum_{i=1}^{n} k(x^*, x_i)\, a_i \\
k(x^*, x_i) &= \langle \varphi(x_i), \varphi(x_j) \rangle \\
&= \begin{bmatrix} 1 & \sqrt{2}x_1^{(i)} & \dots & (x_1^{(i)})^2 & (x_2^{(i)})^2 \end{bmatrix} \begin{bmatrix} 1 \\ \sqrt{2}x_1^{(j)} \\ \vdots \\ (x_1^{(j)})^2 \\ (x_2^{(j)})^2 \end{bmatrix}
\end{aligned}
$$

The "kernel trick"

Notice that all that is required is vector products, i.e.

$$K_{i,j} = \langle \varphi(x_i), \varphi(x_j) \rangle$$

It turns out that these can often be computed *much* more efficiently, without ever forming the big, nonlinear feature space directly.

The "kernel trick"

Notice that all that is required is vector products, i.e.

$$K_{i,j} = \langle \varphi(x_i), \varphi(x_j) \rangle$$

It turns out that these can often be computed *much* more efficiently, without ever forming the big, nonlinear feature space directly. For our quadratic example:

$$K_{i,j} = \left((x^{(i)})^T x^{(j)} + 1 \right)^2 = (x_1^{(i)})^2 (x_1^{(j)})^2 + 2 x_1^{(i)} x_1^{(j)} x_2^{(i)} x_2^{(j)} + \ldots$$

The "kernel trick"

Notice that all that is required is vector products, i.e.

$$K_{i,j} = \langle \varphi(x_i), \varphi(x_j) \rangle$$

It turns out that these can often be computed *much* more efficiently, without ever forming the big, nonlinear feature space directly. For our quadratic example:

$$K_{i,j} = \left((x^{(i)})^T x^{(j)} + 1 \right)^2 = (x_1^{(i)})^2 (x_1^{(j)})^2 + 2 x_1^{(i)} x_1^{(j)} x_2^{(i)} x_2^{(j)} + \dots$$

which can be computed entirely using vectors in $\mathbb{R}^2$, so we never have to allocate the (factorially large) feature space.

Detailed example of nonlinear regression

jupyter notebook: `github.com/jpjanet/ML-chem-workshop/blob/master/notebooks/workshop_compare_models.ipynb`

General kernels

Both kernel methods are the same except:

	linear	quadratic
K_{ij}	$(x^{(i)})^T x^{(j)}$	$\left((x^{(i)})^T x^{(j)} + 1\right)^2$

$$\hat{y}(X) = Ka \qquad\qquad a = (K + I_n\lambda)^{-1} y$$

General kernels

Both kernel methods are the same except:

	linear	quadratic
K_{ij}	$(x^{(i)})^T x^{(j)}$	$\left((x^{(i)})^T x^{(j)} + 1\right)^2$

$$\hat{y}(X) = Ka \qquad\qquad a = (K + I_n\lambda)^{-1} y$$

we use the same machinery as the linear kernel, but the definition of similarity, defined by the kernel, has changed.

General kernels

Both kernel methods are the same except:

	linear	quadratic
K_{ij}	$(x^{(i)})^T x^{(j)}$	$\left((x^{(i)})^T x^{(j)} + 1\right)^2$

$$\hat{y}(X) = Ka \qquad\qquad a = (K + I_n\lambda)^{-1} y$$

we use the same machinery as the linear kernel, but the definition of similarity, defined by the kernel, has changed.

From the perspective of similarity, we can imagine arbitrary functions to be our kernel, without ever needing to know what the underlying feature map φ is.

The Gaussian kernel and KRR

The most widely used kernel is the Gaussian kernel:o

$$K_{i,j} = \exp\left(-\sigma^{-2}\left\|x^{(i)} - x^{(j)}\right\|_2^2\right)$$

The Gaussian kernel and KRR

The most widely used kernel is the Gaussian kernel:o

$$K_{i,j} = \exp\left(-\sigma^{-2}\left\|x^{(i)} - x^{(j)}\right\|_2^2\right)$$

What is $\varphi(X)$ here?

The Gaussian kernel and KRR

The most widely used kernel is the Gaussian kernel:o

$$K_{i,j} = \exp\left(-\sigma^{-2}\left\|x^{(i)} - x^{(j)}\right\|_2^2\right)$$

What is $\varphi(X)$ here?

$$\varphi(x) = e^{\frac{-x^2}{2\sigma^2}}\begin{bmatrix}1 & \frac{1}{\sigma}x & \sqrt{\frac{1}{2!\sigma^4}}x^2 & \dots\end{bmatrix}^T$$

Actually a Hilbert space (infinite dimensions...)

The Gaussian kernel and KRR

The most widely used kernel is the Gaussian kernel:o

$$K_{i,j} = \exp\left(-\sigma^{-2}\left\|x^{(i)} - x^{(j)}\right\|_2^2\right)$$

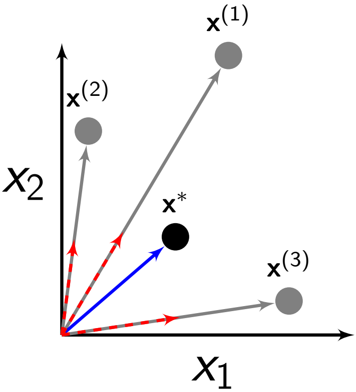
What is $\varphi(X)$ here?

$$\varphi(x) = e^{\frac{-x^2}{2\sigma^2}}\begin{bmatrix}1 & \frac{1}{\sigma}x & \sqrt{\frac{1}{2!\sigma^4}}x^2 & \dots\end{bmatrix}^T$$

Actually a Hilbert space (infinite dimensions...)
Depends on σ to control non-locality.

Similarity and Gaussian KRR

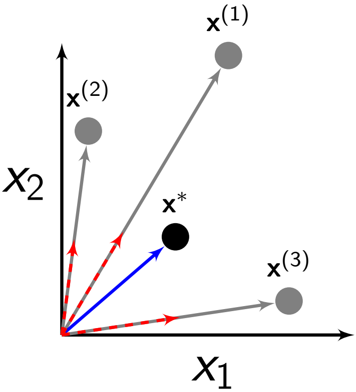
$$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$$



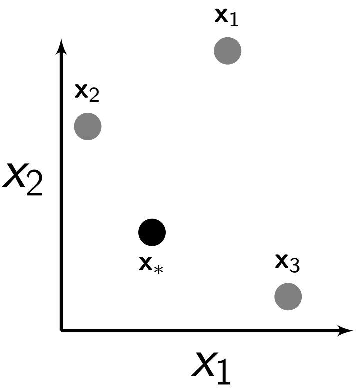
linear kernel

Similarity and Gaussian KRR

$$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$$



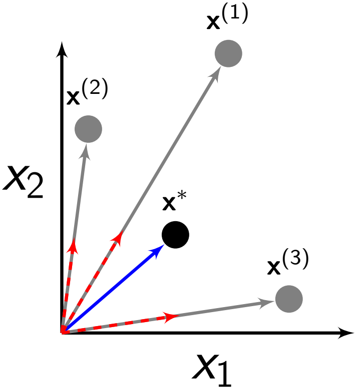
linear kernel



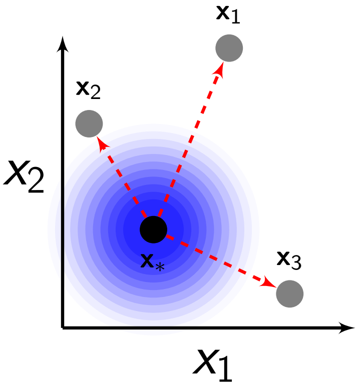
Gaussian kernel

Similarity and Gaussian KRR

$$y(x^*) = \sum_{i=1}^{n} a_i k(x^*, x^{(i)})$$



linear kernel



Gaussian kernel

A simple recipe

A simple recipe for kernel ridge regression (KRR):

A simple recipe

A simple recipe for kernel ridge regression (KRR):

1. form the kernel matrix, $K \in \mathbb{R}^{n \times n}$ with

$$K_{ij} = k(x^{(i)}, x^{(j)}) = \exp\left(-\sigma^{-2} \left\| x^{(i)} - x^{(j)} \right\|_2^2\right)$$

A simple recipe

A simple recipe for kernel ridge regression (KRR):

1. form the kernel matrix, $K \in \mathbb{R}^{n \times n}$ with

$$K_{ij} = k(x^{(i)}, x^{(j)}) = \exp\left(-\sigma^{-2} \left\| x^{(i)} - x^{(j)} \right\|_2^2\right)$$

2. solve for $a \in \mathbb{R}^n$ from

$$a = (K + \lambda I)^{-1} y_{data}$$

A simple recipe

A simple recipe for kernel ridge regression (KRR):

1. form the kernel matrix, $K \in \mathbb{R}^{n \times n}$ with

$$K_{ij} = k(x^{(i)}, x^{(j)}) = \exp\left(-\sigma^{-2} \left\| x^{(i)} - x^{(j)} \right\|_2^2\right)$$

2. solve for $a \in \mathbb{R}^n$ from

$$a = (K + \lambda I)^{-1} y_{data}$$

3. compute values a new points $x^* \in \mathbb{R}^d$ by

$$y(x^*) = \sum_{i=1}^{n} k(x^*, x^{(i)}) a_i$$

A simple recipe

A simple recipe for kernel ridge regression (KRR):

1. form the kernel matrix, $K \in \mathbb{R}^{n \times n}$ with

$$K_{ij} = k(x^{(i)}, x^{(j)}) = \exp\left(-\sigma^{-2} \left\| x^{(i)} - x^{(j)} \right\|_2^2\right)$$

2. solve for $a \in \mathbb{R}^n$ from

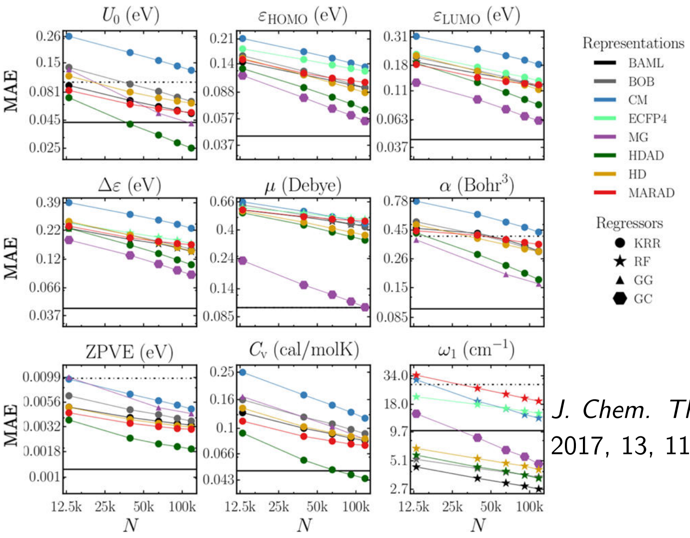
$$a = (K + \lambda I)^{-1} y_{data}$$

3. compute values a new points $x^* \in \mathbb{R}^d$ by

$$y(x^*) = \sum_{i=1}^{n} k(x^*, x^{(i)}) a_i$$

4. check using cross-validation to choose σ and λ

KRR is widely used in chemistry



J. Chem. Theory Comput. 2017, 13, 11, 5255–5264

Conclusion

In summary:

Conclusion

In summary:

1. Models that are linear in their parameters can be formulated as kernel models

Conclusion

In summary:

1. Models that are linear in their parameters can be formulated as kernel models
2. The choice of kernel uniquely defines the working of similarity in your application

Conclusion

In summary:

1. Models that are linear in their parameters can be formulated as kernel models
2. The choice of kernel uniquely defines the working of similarity in your application
3. Kernel models are quick to train, easy to understand and suitable for low-data applications

Conclusion

In summary:

1. Models that are linear in their parameters can be formulated as kernel models
2. The choice of kernel uniquely defines the working of similarity in your application
3. Kernel models are quick to train, easy to understand and suitable for low-data applications
4. Cross-validation (and related techniques) must be used to choose hyperparameters

Conclusion

In summary:

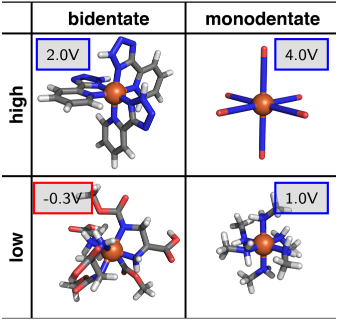
1. Models that are linear in their parameters can be formulated as kernel models
2. The choice of kernel uniquely defines the working of similarity in your application
3. Kernel models are quick to train, easy to understand and suitable for low-data applications
4. Cross-validation (and related techniques) must be used to choose hyperparameters

KRR example

jupyter notebook: `github.com/jpjanet/ML-chem-workshop/blob/master/notebooks/KRR.ipynb`

Why do feature selection?

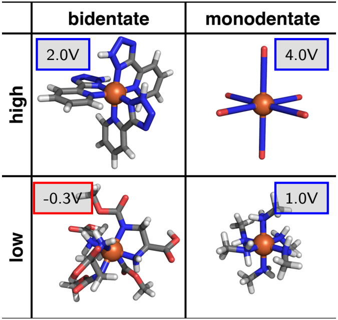
This can help explain what factors are important[5]:



[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:



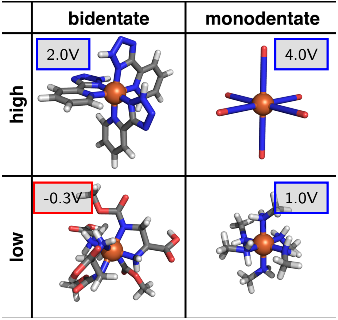
How can we account for the difference?

[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:

Could correlate with:



How can we account for the difference?

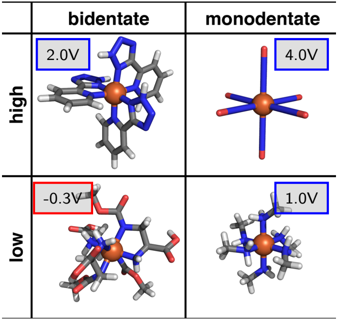
[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:

Could correlate with:

- χ (ANN paper)



How can we account for the difference?

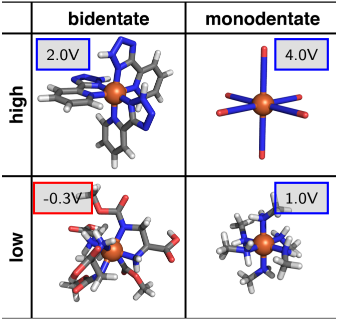
[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:

Could correlate with:

- χ (ANN paper)
- Z (Coulomb matrix)



How can we account for the difference?

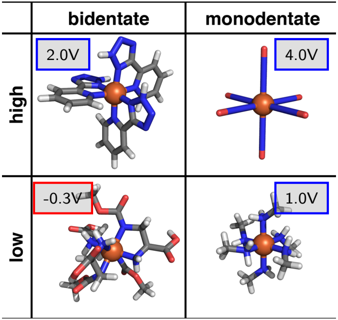
[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:

Could correlate with:

- χ (ANN paper)
- Z (Coulomb matrix)
- T (topology $\sim$ Randić)



How can we account for the difference?

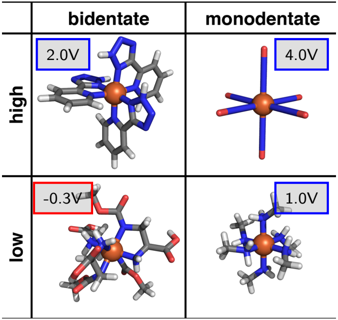
[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:

Could correlate with:

- χ (ANN paper)
- Z (Coulomb matrix)
- T (topology $\sim$ Randić)
- I (count/size)



How can we account for the difference?

[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

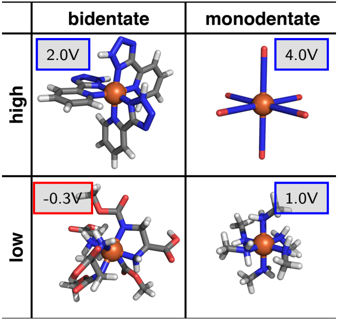
Why do feature selection?

This can help explain what factors are important[5]:

Could correlate with:

- χ (ANN paper)
- Z (Coulomb matrix)
- T (topology $\sim$ Randić)
- I (count/size)

Can use *autocorrelation* functions to describe how ligand atoms are connected



How can we account for the difference?

[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:

Could correlate with:

- χ (ANN paper)
- Z (Coulomb matrix)
- T (topology $\sim$ Randić)
- I (count/size)

Can use *autocorrelation* functions to describe how ligand atoms are connected

$$\eta_{i,0} = p_i p_i$$

$$\eta_{i,k} = \sum_{i \neq j} p_i p_j \delta(d_{i,j} - d_k), \quad k \neq 0$$

$$\mathrm{AC}_k = \sum_i \eta_{i,k}$$

4 properties, $k \in [0, 5]$

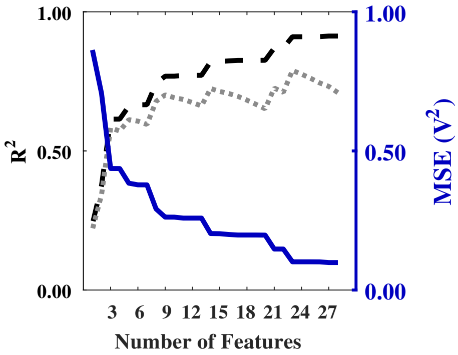
$\implies$ 28 variables.

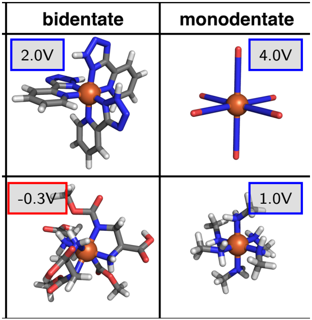
How to choose?

[5] JP Janet *et al. Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:

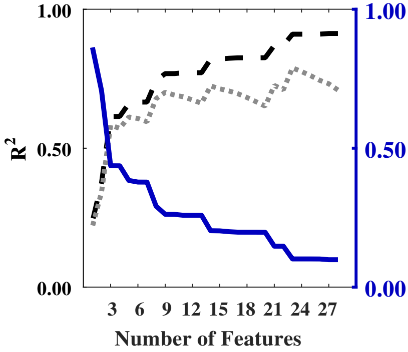


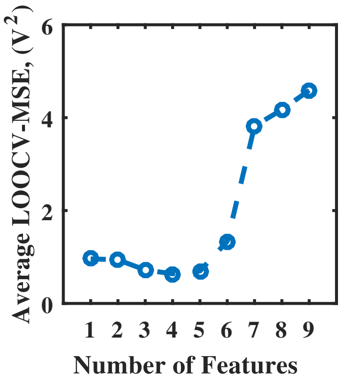


[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

Why do feature selection?

This can help explain what factors are important[5]:





[5] JP Janet *et al.* *Ind. Eng. Chem. Res.* 2017, 56, 17, 4898–4910

How to pick important features?

Using unnecessary features can degrade model performance, so we want to able to pick the subset of variables that is best correlated with our objective, formally:

$$w = \arg\min_{w \in} \left(\|y_{data} - Xw\|_2^2 + \lambda \|w\|_0^0 \right)$$

How to pick important features?

Using unnecessary features can degrade model performance, so we want to able to pick the subset of variables that is best correlated with our objective, formally:

$$w = \arg\min_{w\in} \left(\|y_{data} - Xw\|_2^2 + \lambda \|w\|_0^0 \right)$$

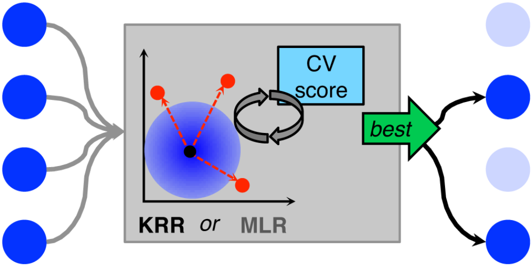
We don't know the optimal number upfront, and this is a combinatorial problem – possible for ≤ 30 dimensions or so, but rapidly becomes unfeasible.

How to pick important features?

Instead, we can do recursive feature addition/removal. Starting from all (or no) features, we test each feature and remove the one that improves performance most (**crucial to use CV error here**):

How to pick important features?

Instead, we can do recursive feature addition/removal. Starting from all (or no) features, we test each feature and remove the one that improves performance most (**crucial to use CV error here**):



LASSO regression

LASSO regression gives us a one-shot option for linear models:

$$w = \arg\min_{w} \mathcal{L}(w) = \|Xw - y_{data}\|_2^2 + \lambda \|W\|_0^0$$

LASSO regression

LASSO regression gives us a one-shot option for linear models:

$$w = \arg\min_{w} \mathcal{L}(w) = \|Xw - y_{data}\|_2^2 + \lambda \|W\|_1^1$$

LASSO regression

LASSO regression gives us a one-shot option for linear models:

$$w = \arg\min_{w} \mathcal{L}(w) = \|Xw - y_{data}\|_2^2 + \lambda \|W\|_1^1$$

Using the l_1 makes the optimization problem more difficult, but automatically performs feature selection so we don't need RFE or other tricks.

LASSO regression

LASSO regression gives us a one-shot option for linear models:

$$w = \arg\min_{w} \mathcal{L}(w) = \|Xw - y_{data}\|_2^2 + \lambda \|W\|_1^1$$

Using the l_1 makes the optimization problem more difficult, but automatically performs feature selection so we don't need RFE or other tricks. Why? Note the above is equivalent to

$$w = \arg\min_{w} \mathcal{L}(w) = \|Xw - y_{data}\|_2^2$$
$$\text{s.t. } w \leq \lambda^{-1}$$

(by Lagrange multipliers)

LASSO regression

LASSO regression gives us a one-shot option for linear models:

$$w = \arg\min_w \mathcal{L}(w) = \|Xw - y_{data}\|_2^2 + \lambda \|W\|_1^1$$

Using the l_1 makes the optimization problem more difficult, but automatically performs feature selection so we don't need RFE or other tricks. Why? Note the above is equivalent to

$$w = \arg\min_w \mathcal{L}(w) = \|Xw - y_{data}\|_2^2$$
$$\text{s.t. } w \leq \lambda^{-1}$$

(by Lagrange multipliers)

- the **larger** λ, the fewer features survive

LASSO regression

LASSO regression gives us a one-shot option for linear models:

$$w = \arg\min_{w} \mathcal{L}(w) = \|Xw - y_{data}\|_2^2 + \lambda \|W\|_1^1$$

Using the l_1 makes the optimization problem more difficult, but automatically performs feature selection so we don't need RFE or other tricks. Why? Note the above is equivalent to

$$w = \arg\min_{w} \mathcal{L}(w) = \|Xw - y_{data}\|_2^2$$
$$\text{s.t. } w \leq \lambda^{-1}$$

(by Lagrange multipliers)

- the **larger** λ, the fewer features survive
- λ comes from cross-validation

Elastic net

$$w = \arg\min_{w \in \mathbb{R}^p} \|Xw - y_{data}\|_2^2$$

Elastic net

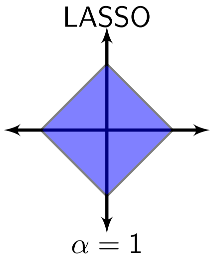
$$w = \arg\min_{w \in \mathbb{R}^p} \|Xw - y_{data}\|_2^2 + \lambda \left(\alpha \|w\|_1^1 \qquad\qquad\qquad \right)$$

Elastic net

$$w = \arg\min_{w \in \mathbb{R}^p} \|Xw - y_{data}\|_2^2 + \lambda \left(\alpha \|w\|_1^1 + (1-\alpha) \|w\|_2^2 \right)$$

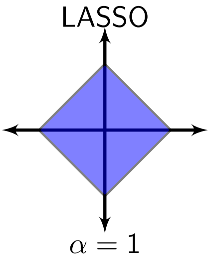
Elastic net

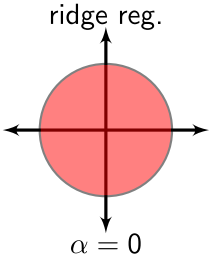
$$w = \arg\min_{w \in \mathbb{R}^p} \|Xw - y_{data}\|_2^2 + \lambda \left(\alpha \|w\|_1^1 + (1-\alpha) \|w\|_2^2 \right)$$



Elastic net

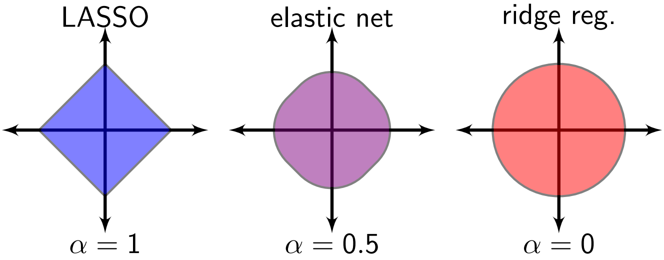
$$w = \arg\min_{w \in \mathbb{R}^p} \|Xw - y_{data}\|_2^2 + \lambda \left(\alpha \|w\|_1^1 + (1-\alpha) \|w\|_2^2 \right)$$





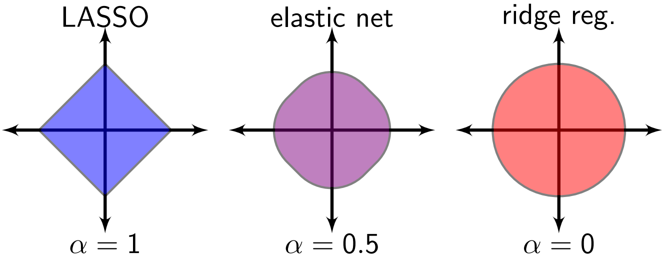
Elastic net

$$w = \arg\min_{w \in \mathbb{R}^p} \|Xw - y_{data}\|_2^2 + \lambda \left(\alpha \|w\|_1^1 + (1-\alpha) \|w\|_2^2 \right)$$



Elastic net

$$w = \arg\min_{w \in \mathbb{R}^p} \|Xw - y_{data}\|_2^2 + \lambda \left(\alpha \|w\|_1^1 + (1-\alpha) \|w\|_2^2 \right)$$



Using even a small $\alpha > 0$ ensures the minimization is stable.

How does LASSO work?

Why does l_1 character drive feature selection?

How does LASSO work?

Why does l_1 character drive feature selection?

$$\|W\|_2^2 = \begin{vmatrix}1\\0\end{vmatrix}_2 = \sqrt{1^2+0^2} = 1$$
$$\|W'\|_2^2 = \begin{vmatrix}0.5\\0.5\end{vmatrix}_2 = \sqrt{0.5^2+0.5^2} = \sqrt{0.5} < 1$$

How does LASSO work?

Why does l_1 character drive feature selection?

$$\|W\|_2^2 = \begin{vmatrix} 1 \\ 0 \end{vmatrix}_2 = \sqrt{1^2 + 0^2} = 1$$
$$\|W'\|_2^2 = \begin{vmatrix} 0.5 \\ 0.5 \end{vmatrix}_2 = \sqrt{0.5^2 + 0.5^2} = \sqrt{0.5} < 1$$

compare to

$$\|W\|_1^1 = \begin{vmatrix} 1 \\ 0 \end{vmatrix}_1 = |1| + |1| = 1$$
$$\|W'\|_1^1 = \begin{vmatrix} 0.5 \\ 0.5 \end{vmatrix}_1 = |0.5| + |0.5| = 1$$

How does LASSO work?

Why does l_1 character drive feature selection?

$$\|W\|_2^2 = \begin{vmatrix}1\\0\end{vmatrix}_2 = \sqrt{1^2+0^2} = 1$$
$$\|W'\|_2^2 = \begin{vmatrix}0.5\\0.5\end{vmatrix}_2 = \sqrt{0.5^2+0.5^2} = \sqrt{0.5} < 1$$

compare to

$$\|W\|_1^1 = \begin{vmatrix}1\\0\end{vmatrix}_1 = |1| + |1| = 1$$
$$\|W'\|_1^1 = \begin{vmatrix}0.5\\0.5\end{vmatrix}_1 = |0.5| + |0.5| = 1$$

Why not use even lower norms such as $\|w\|_{0.5}^{0.5}$?

How does LASSO work?

Why does l_1 character drive feature selection?

$$\|W\|_2^2 = \begin{vmatrix} 1 \\ 0 \end{vmatrix}_2 = \sqrt{1^2 + 0^2} = 1$$

$$\|W'\|_2^2 = \begin{vmatrix} 0.5 \\ 0.5 \end{vmatrix}_2 = \sqrt{0.5^2 + 0.5^2} = \sqrt{0.5} < 1$$

compare to

$$\|W\|_1^1 = \begin{vmatrix} 1 \\ 0 \end{vmatrix}_1 = |1| + |1| = 1$$

$$\|W'\|_1^1 = \begin{vmatrix} 0.5 \\ 0.5 \end{vmatrix}_1 = |0.5| + |0.5| = 1$$

Why not use even lower norms such as $\|w\|_{0.5}^{0.5}$? Convexity!

Conclusions

In summary:

Conclusions

In summary:

1. Feature selection techniques can help identify important features, for modeling and for interpretation
2. Iterative subset selection can be expensive since the model needs to be re-trained, including hyperparameters each time
3. LASSO/elastic net provide simple ways of extracting most important linear features

What is a neural network?

Neural networks are a family of nonlinear models distinguished by:

What is a neural network?

Neural networks are a family of nonlinear models distinguished by:

1. High, 'controllable', degree of nonlinearity (c.f. universal approximation theorems).

What is a neural network?

Neural networks are a family of nonlinear models distinguished by:

1. High, 'controllable', degree of nonlinearity (c.f. universal approximation theorems).

2. Continuity, differentiability (high Lipschitz constant)

What is a neural network?

Neural networks are a family of nonlinear models distinguished by:

1. High, 'controllable', degree of nonlinearity (c.f. universal approximation theorems).
2. Continuity, differentiability (high Lipschitz constant)
3. Easy access to cheap derivatives!

What is a neural network?

Neural networks are a family of nonlinear models distinguished by:

1. High, 'controllable', degree of nonlinearity (c.f. universal approximation theorems).
2. Continuity, differentiability (high Lipschitz constant)
3. Easy access to cheap derivatives!
4. Cheap at evaluation time

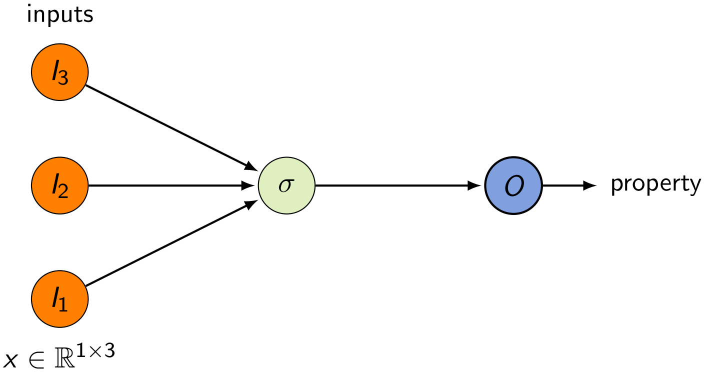
What is a neural network?

Neural networks are a family of nonlinear models distinguished by:

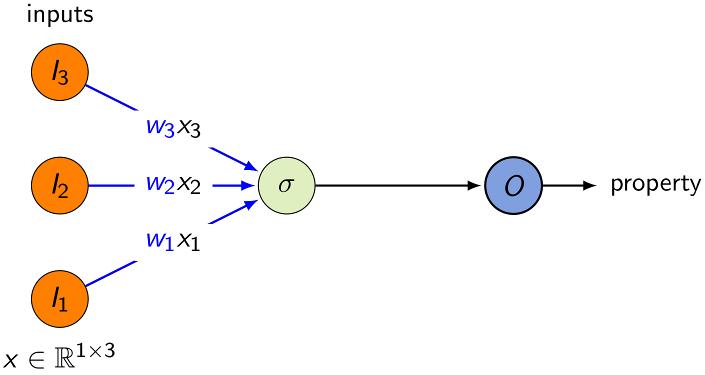
1. High, 'controllable', degree of nonlinearity (c.f. universal approximation theorems).
2. Continuity, differentiability (high Lipschitz constant)
3. Easy access to cheap derivatives!
4. Cheap at evaluation time
5. easily scaled for different input structures.

The neuron

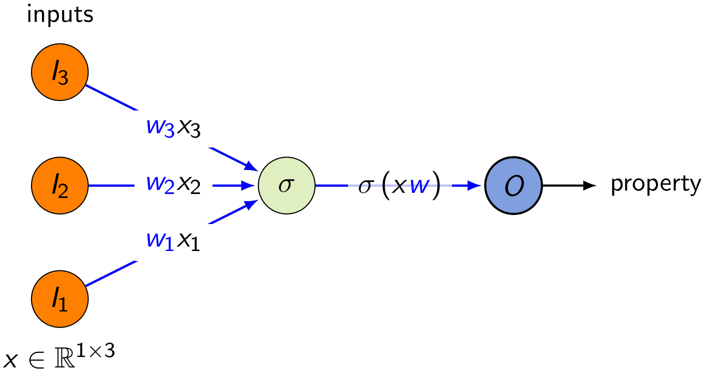
The neuron



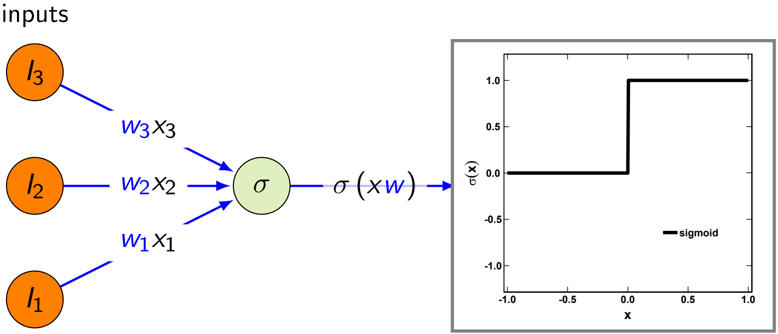
The neuron


inputs
I_3
I_2
I_1
$w_3 x_3$
$w_2 x_2$
$w_1 x_1$
σ
O
property
$x \in \mathbb{R}^{1\times 3}$

The neuron


inputs
I_3
I_2
I_1
$w_3 x_3$
$w_2 x_2$
$w_1 x_1$
σ
$\sigma(xw)$
O
property
$x \in \mathbb{R}^{1\times 3}$

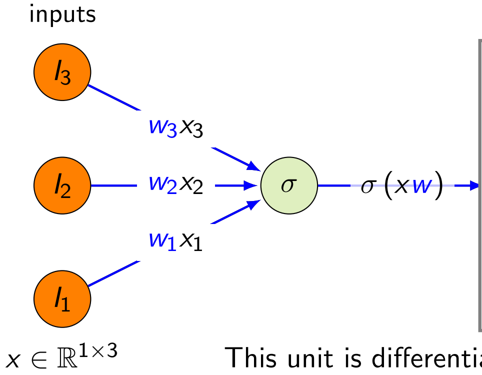
The neuron

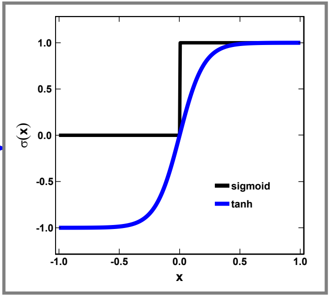


$x \in \mathbb{R}^{1\times 3}$

This unit is differentiable as long as σ is.

The neuron

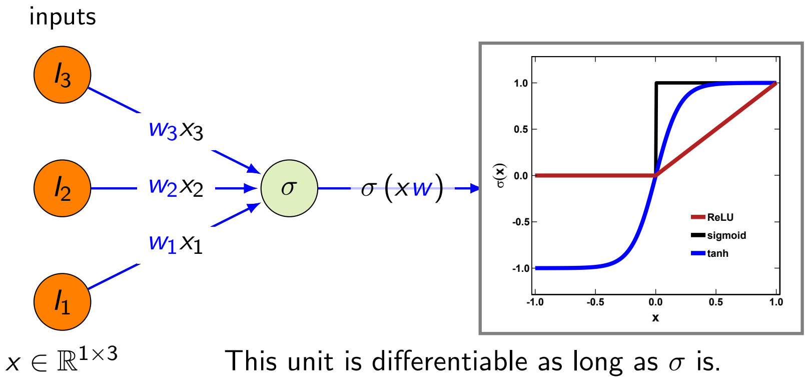




This unit is differentiable as long as σ is.

$$\frac{\partial O}{\partial w_1} = x_1 \sigma'(xw)$$

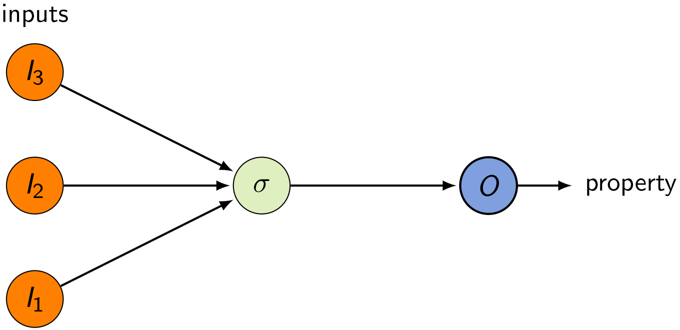
The neuron



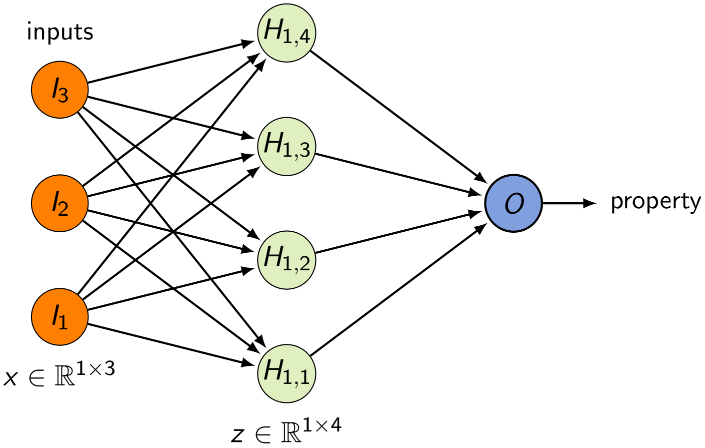
This unit is differentiable as long as σ is.

$$\implies \frac{\partial O}{\partial w} = x\sigma'(xw)$$

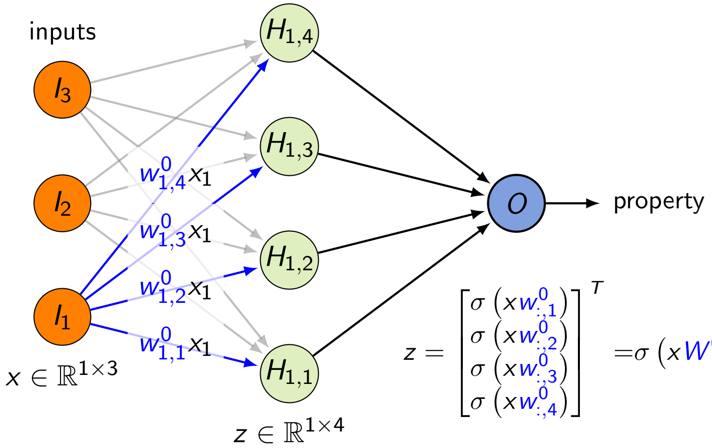
The perceptron



The perceptron

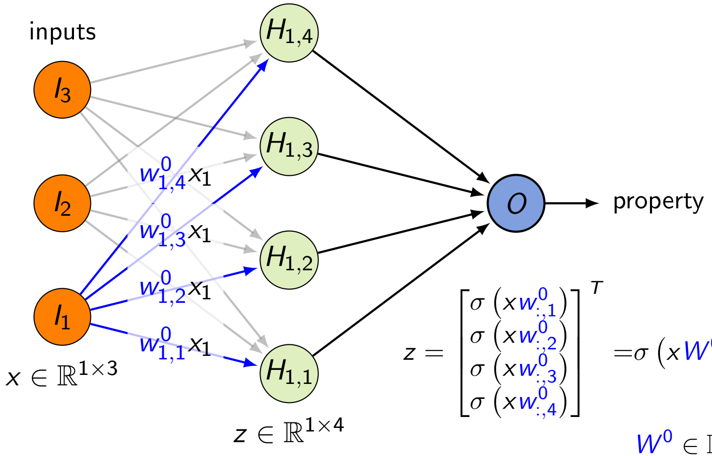


The perceptron



$$z = \begin{bmatrix} \sigma\left(x w^0_{:,1}\right) \\ \sigma\left(x w^0_{:,2}\right) \\ \sigma\left(x w^0_{:,3}\right) \\ \sigma\left(x w^0_{:,4}\right) \end{bmatrix}^T = \sigma\left(x W^0\right)$$

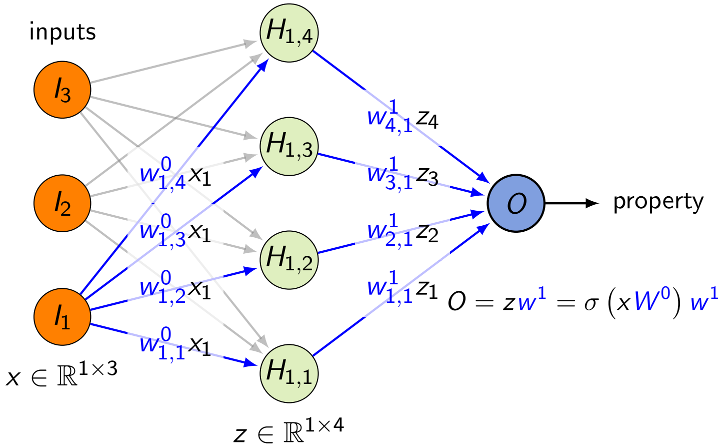
The perceptron



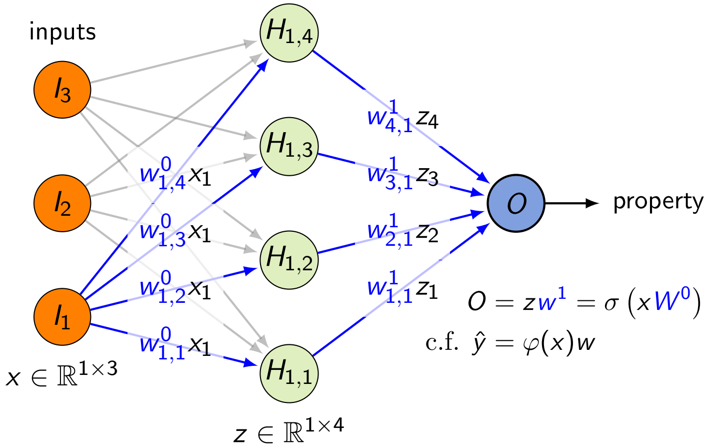
$$z = \begin{bmatrix} \sigma\left(x w^0_{:,1}\right) \\ \sigma\left(x w^0_{:,2}\right) \\ \sigma\left(x w^0_{:,3}\right) \\ \sigma\left(x w^0_{:,4}\right) \end{bmatrix}^T = \sigma\left(x W^0\right)$$

$$W^0 \in \mathbb{R}^{3\times 4}$$

The perceptron



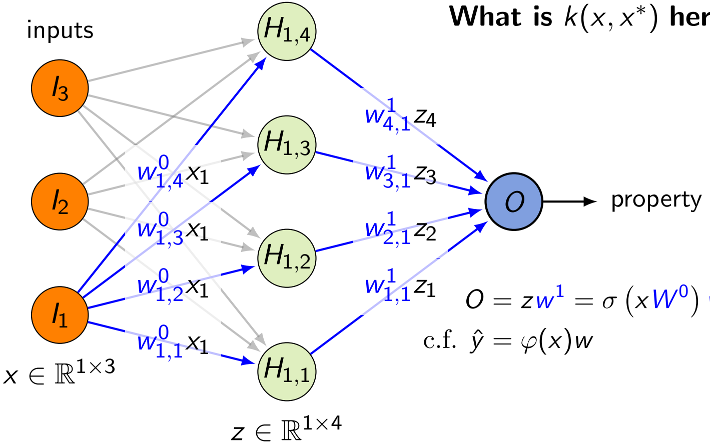
The perceptron



$$O = zw^1 = \sigma\left(xW^0\right)w^1$$

c.f. $\hat{y} = \varphi(x)w$

The perceptron

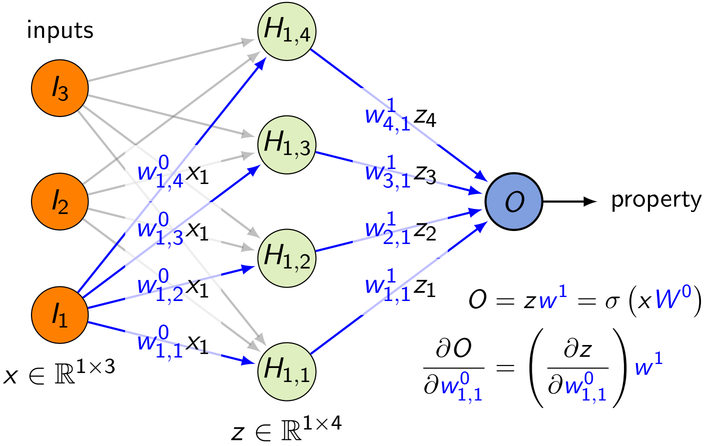


What is $k(x, x^*)$ here?

$$O = zw^1 = \sigma\left(xW^0\right)w^1$$

c.f. $\hat{y} = \varphi(x)w$

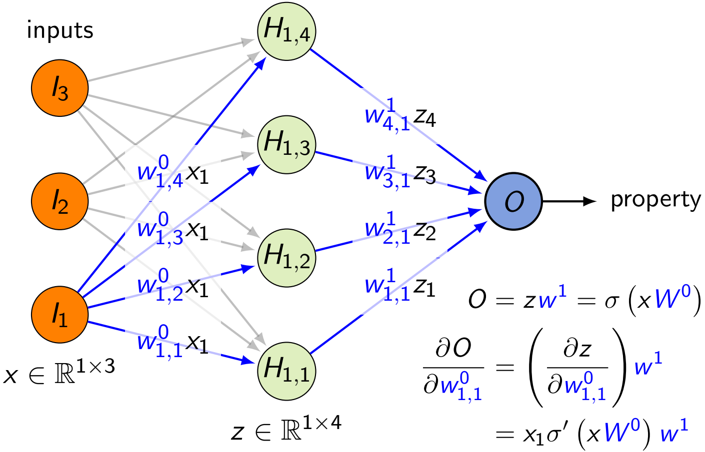
The perceptron



$$O = zw^1 = \sigma\left(xW^0\right)w^1$$

$$\frac{\partial O}{\partial w^0_{1,1}} = \left(\frac{\partial z}{\partial w^0_{1,1}}\right)w^1$$

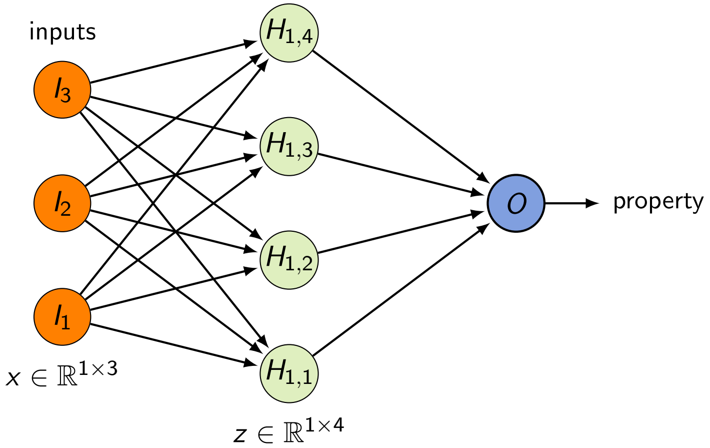
The perceptron



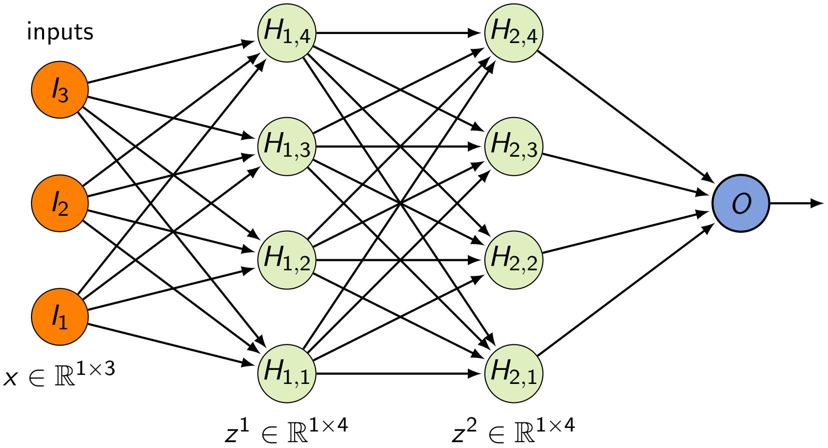
$$O = zw^1 = \sigma\left(xW^0\right)w^1$$

$$\frac{\partial O}{\partial w^0_{1,1}} = \left(\frac{\partial z}{\partial w^0_{1,1}}\right)w^1 = x_1\sigma'\left(xW^0\right)w^1$$

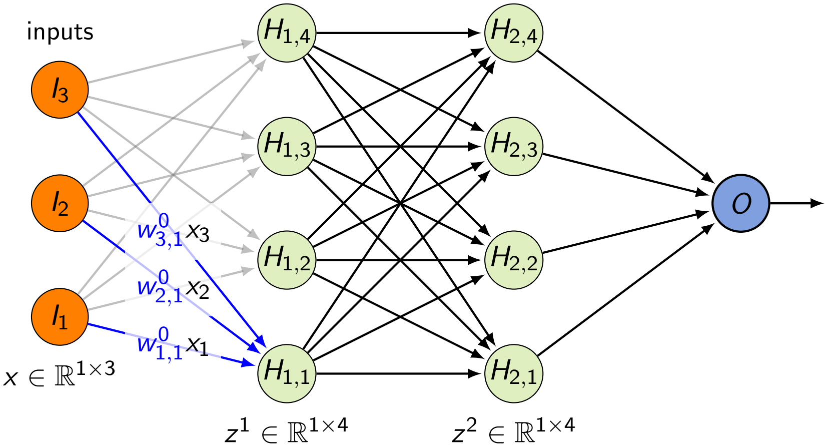
The *multilayer* perceptron



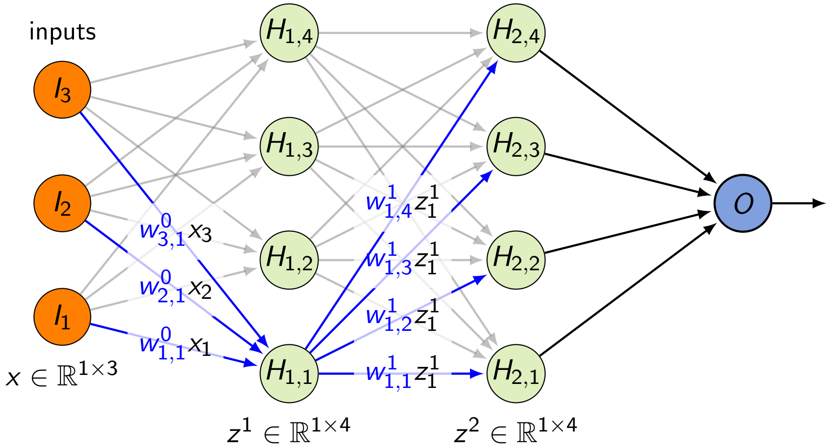
The *multilayer* perceptron



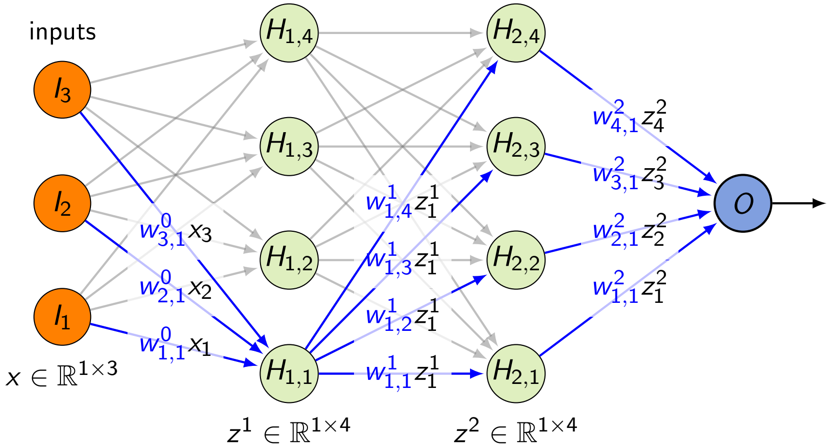
The *multilayer* perceptron



The *multilayer* perceptron

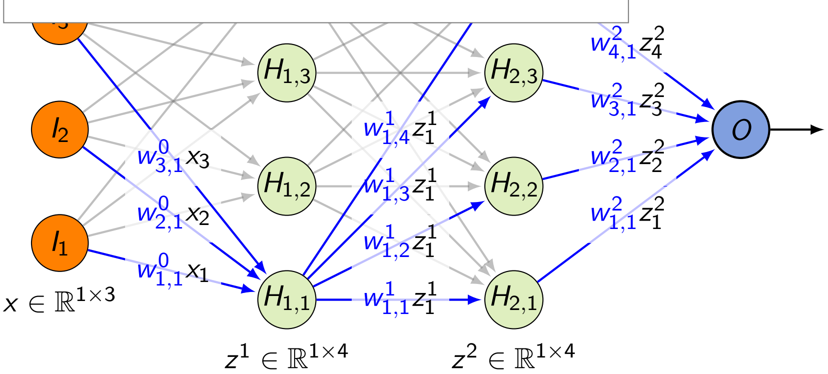


The *multilayer* perceptron



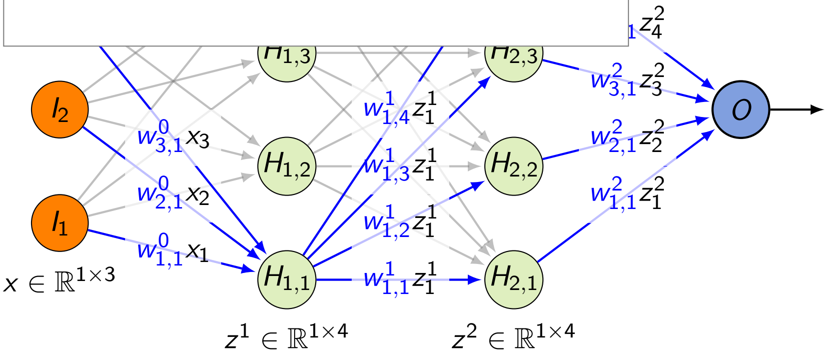
The *multilayer* perceptron

$$O = z^2 w^2 = \sigma\left(z^1 W^1\right) w^2$$



The *multilayer* perceptron

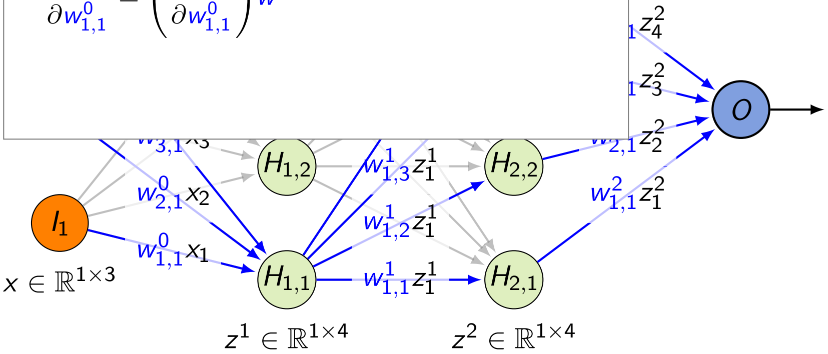
$$
\begin{aligned}
O &= z^2 w^2 = \sigma\left(z^1 W^1\right) w^2 \\
&= \sigma\left(\left[\sigma\left(x W^0\right)\right] W^1\right) w^2
\end{aligned}
$$



The *multilayer* perceptron

$$
\begin{aligned}
O &= z^2 w^2 = \sigma\left(z^1 W^1\right) w^2 \\
&= \sigma\left(\left[\sigma\left(x W^0\right)\right] W^1\right) w^2
\end{aligned}
$$

$$
\frac{\partial O}{\partial w^0_{1,1}} = \left(\frac{\partial z^2}{\partial w^0_{1,1}}\right) w^2
$$

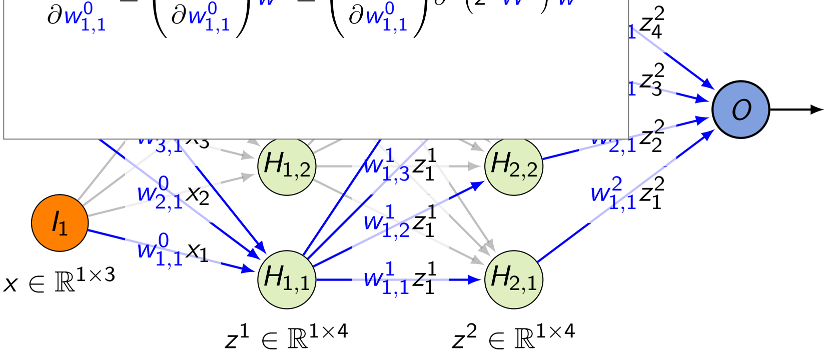


The *multilayer* perceptron

$$O = z^2 w^2 = \sigma\left(z^1 W^1\right) w^2$$

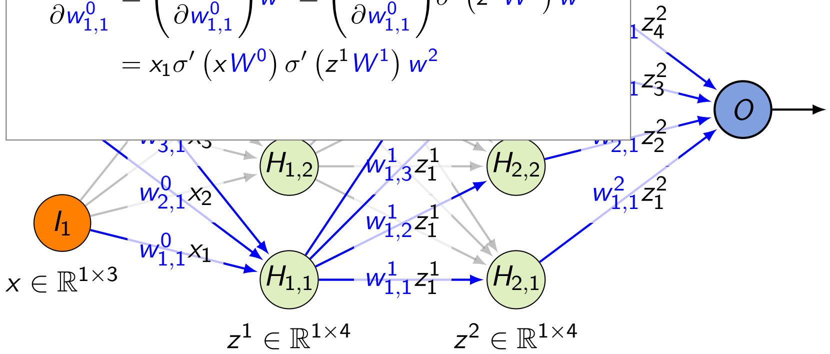
$$= \sigma\left(\left[\sigma\left(x W^0\right)\right] W^1\right) w^2$$

$$\frac{\partial O}{\partial w^0_{1,1}} = \left(\frac{\partial z^2}{\partial w^0_{1,1}}\right) w^2 = \left(\frac{\partial z^1}{\partial w^0_{1,1}}\right) \sigma'\left(z^1 W^1\right) w^2$$



The *multilayer* perceptron

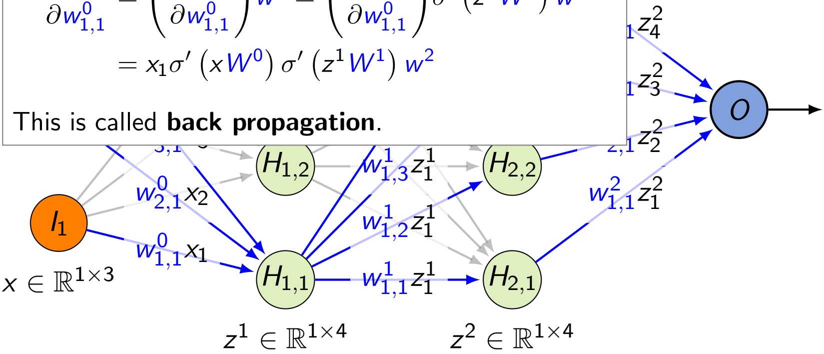
$$
\begin{aligned}
O &= z^2 w^2 = \sigma\left(z^1 W^1\right) w^2 \\
&= \sigma\left(\left[\sigma\left(x W^0\right)\right] W^1\right) w^2 \\
\frac{\partial O}{\partial w^0_{1,1}} &= \left(\frac{\partial z^2}{\partial w^0_{1,1}}\right) w^2 = \left(\frac{\partial z^1}{\partial w^0_{1,1}}\right) \sigma'\left(z^1 W^1\right) w^2 \\
&= x_1 \sigma'\left(x W^0\right) \sigma'\left(z^1 W^1\right) w^2
\end{aligned}
$$



The *multilayer* perceptron

$$
\begin{aligned}
O &= z^2 w^2 = \sigma\left(z^1 W^1\right) w^2 \\
&= \sigma\left(\left[\sigma\left(x W^0\right)\right] W^1\right) w^2 \\
\frac{\partial O}{\partial w^0_{1,1}} &= \left(\frac{\partial z^2}{\partial w^0_{1,1}}\right) w^2 = \left(\frac{\partial z^1}{\partial w^0_{1,1}}\right) \sigma'\left(z^1 W^1\right) w^2 \\
&= x_1 \sigma'\left(x W^0\right) \sigma'\left(z^1 W^1\right) w^2
\end{aligned}
$$

This is called **back propagation**.

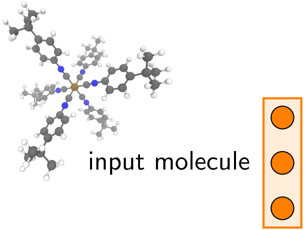


How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:

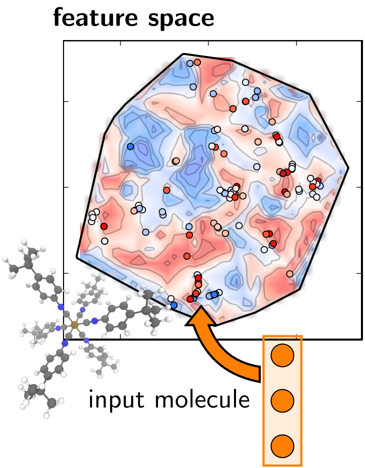
How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



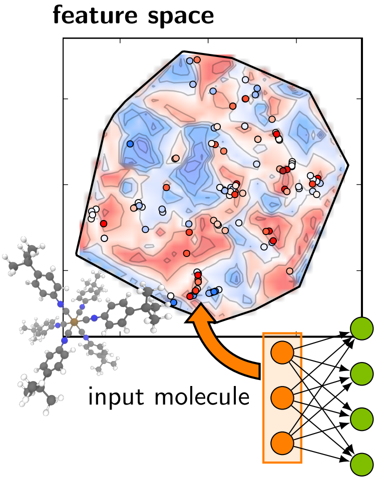
How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



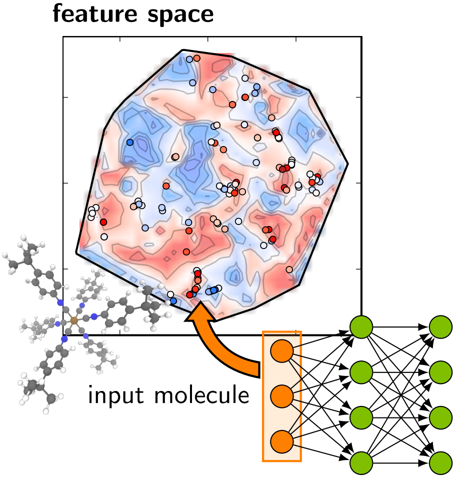
How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



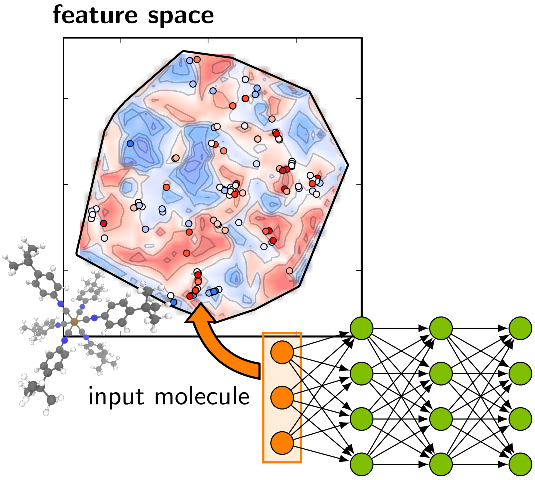
How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



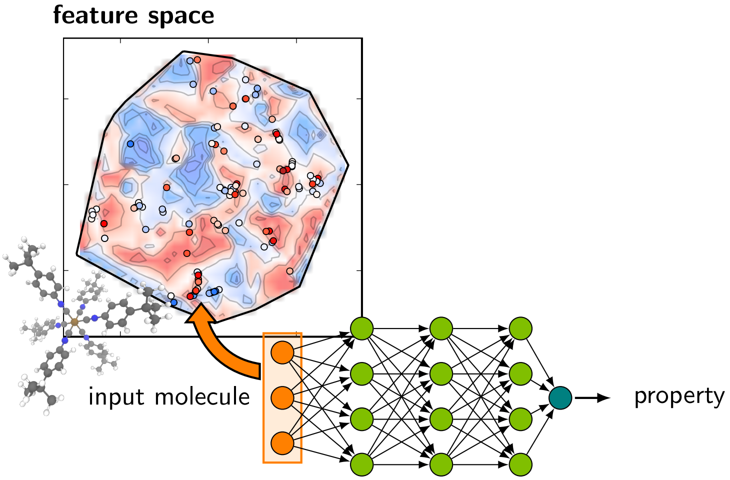
How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



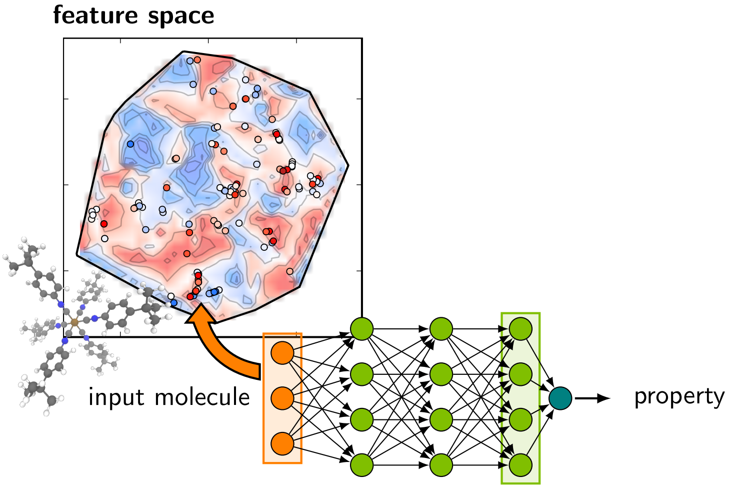
How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



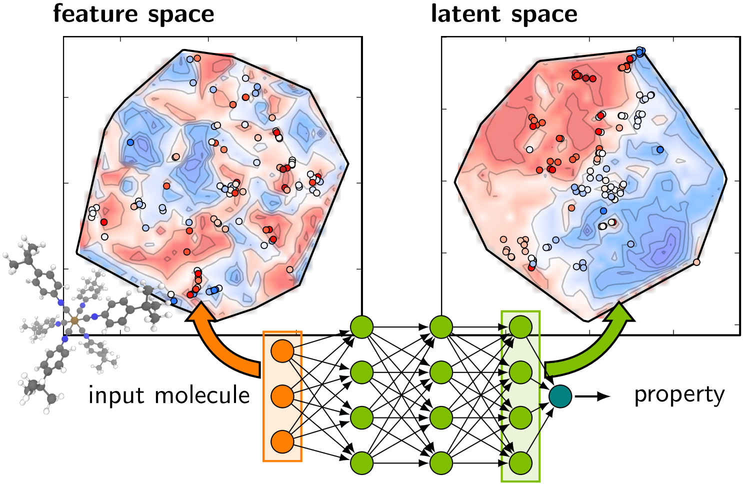
How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



How ANNs work

ANNs learn nonlinear reorganizations of the input to make prediciton easier:



SGD and training process

Backpropagation is normally used to find the gradients. Since all of the gradients are made easy to find with backpropagation, they can be updated with gradient descent:

SGD and training process

Backpropagation is normally used to find the gradients. Since all of the gradients are made easy to find with backpropagation, they can be updated with gradient descent:

$$W = W - \eta * \left(\frac{\partial Loss}{\partial W}\right)$$

SGD and training process

Backpropagation is normally used to find the gradients. Since all of the gradients are made easy to find with backpropagation, they can be updated with gradient descent:

$$W = W - \eta * \left(\frac{\partial Loss}{\partial W}\right)$$

In gradient descent, loss term calculated from training data $\rightarrow$ if you train for the same number of epochs (number of rounds your NN sees the training data), you will get the same result. **Gets stuck in local minima!**

SGD and training process

Backpropagation is normally used to find the gradients. Since all of the gradients are made easy to find with backpropagation, they can be updated with gradient descent:

$$W = W - \eta * \left(\frac{\partial Loss}{\partial W}\right)$$

In gradient descent, loss term calculated from training data $\rightarrow$ if you train for the same number of epochs (number of rounds your NN sees the training data), you will get the same result. **Gets stuck in local minima!**

$$GD \rightarrow \left(\frac{\partial Loss}{\partial W}\right) \rightarrow \text{all training data}$$

SGD and training process

Backpropagation is normally used to find the gradients. Since all of the gradients are made easy to find with backpropagation, they can be updated with gradient descent:

$$W = W - \eta * \left(\frac{\partial Loss}{\partial W}\right)$$

In gradient descent, loss term calculated from training data $\rightarrow$ if you train for the same number of epochs (number of rounds your NN sees the training data), you will get the same result. **Gets stuck in local minima!**

$$GD \rightarrow \left(\frac{\partial Loss}{\partial W}\right) \rightarrow \text{all training data}$$

$$SGD \rightarrow \left(\frac{\partial Loss}{\partial W}\right) \rightarrow \text{one/few examples} \rightarrow \text{MUCH noisier!} \rightarrow \text{less stuck}$$

Other types of layers

Input$\rightarrow$ layers on the NN where inputs are fed

Other types of layers

Input$\rightarrow$ layers on the NN where inputs are fed
Hidden$\rightarrow$ layers between the input and output nodes, helps to create the latent space (rearrangement of initial input points in space)

Other types of layers

Input$\rightarrow$ layers on the NN where inputs are fed
Hidden$\rightarrow$ layers between the input and output nodes, helps to create the latent space (rearrangement of initial input points in space)
Dropout$\rightarrow$ zero out nodes to reduce specific node dependence
Output$\rightarrow$ Node right before result, linear for regression, sigmoid for classification. Maps latent space to final answer

Other types of layers

Input→ layers on the NN where inputs are fed
Hidden→ layers between the input and output nodes, helps to create the latent space (rearrangement of initial input points in space)
Dropout→ zero out nodes to reduce specific node dependence
Output→ Node right before result, linear for regression, sigmoid for classification. Maps latent space to final answer

Other types of layers

Input$\rightarrow$ layers on the NN where inputs are fed
Hidden$\rightarrow$ layers between the input and output nodes, helps to create the latent space (rearrangement of initial input points in space)
Dropout$\rightarrow$ zero out nodes to reduce specific node dependence
Output$\rightarrow$ Node right before result, linear for regression, sigmoid for classification. Maps latent space to final answer
Convolutional$\rightarrow$ Layer used to extract information or "focus on" certain features. Use a sliding "feature detector" called a filter, which creates a map

Other types of layers

Input→ layers on the NN where inputs are fed
Hidden→ layers between the input and output nodes, helps to create the latent space (rearrangement of initial input points in space)
Dropout→ zero out nodes to reduce specific node dependence
Output→ Node right before result, linear for regression, sigmoid for classification. Maps latent space to final answer
Convolutional→ Layer used to extract information or "focus on" certain features. Use a sliding "feature detector" called a filter, which creates a map
Recurrent→ Layers that "store" information about previous times, thus commonly used in speech or handwriting recognition

Interpretation as representation learning

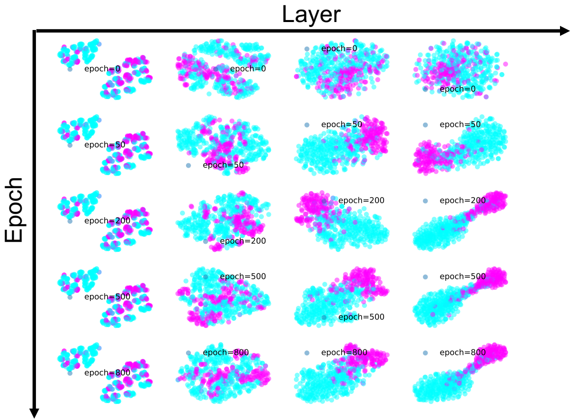
A neural network (NN) takes an initial representation as an input.

Interpretation as representation learning

A neural network (NN) takes an initial representation as an input. Hidden layers in the NN utilize distort the representation in a high dimensional space, thus learning a new representation. This separation allows ease of classification or regression

Interpretation as representation learning

A neural network (NN) takes an initial representation as an input. Hidden layers in the NN utilize distort the representation in a high dimensional space, thus learning a new representation. This separation allows ease of classification or regression



Conclusions

In summary:

Conclusions

In summary:

1. Neural network models provide model complexity 'on tap'

Conclusions

In summary:

1. Neural network models provide model complexity ‘on tap’
2. Backpropagation allows easy access to derivatives

Conclusions

In summary:

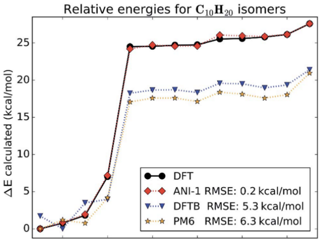
1. Neural network models provide model complexity 'on tap'
2. Backpropagation allows easy access to derivatives
3. We can understand neural networks as automatic feature selection/transformation, followed by linear regression

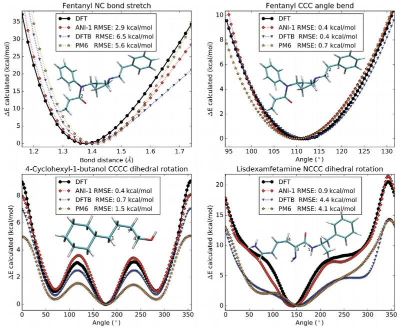
ANN example

jupyter notebook: `github.com/jpjanet/ML-chem-workshop/blob/master/notebooks/ANN.ipynb`

Neural network potentials

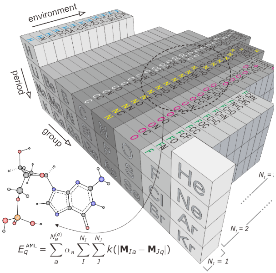
Smith, J. S., Isayev, O., and Roitberg, A. E. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.*, **2017**, *8*, 3192-3203.

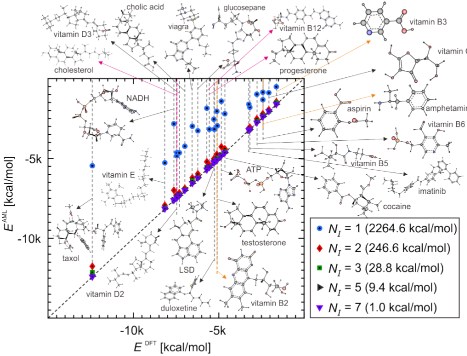




Property predictions

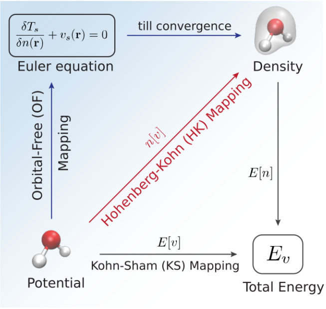
Huang, B. & von Lilienfeld, O.A. *arXiv* 1707.04146, "The 'DNA'of chemistry: Scalable quantum machine learning with amons", 2017.

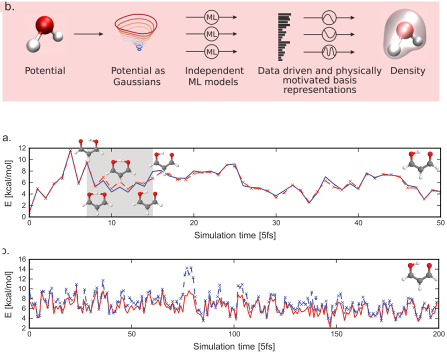




Accelerating quantum chemistry

Bogojeski, M. *el al.*, Burke, K. and Müller, K.R. *arXiv* 1811.06255, Efficient prediction of 3D electron densities using machine learning, 2018.





Controlling calculations on the fly

Chenru Duan *et. al.* *ChemRxiv* .7616009, "Learning from Failure: Predicting Electronic Structure Calculation Outcomes with Machine Learning Models", 2019.