

Fairness, bias, and transparency in Machine Learning

Module 4: Interpretable and explainable ML

This module

- 4.1: Introduction to interpretability and explainability
- Case study: the mythos of model interpretability
- 4.2: Explainability techniques for linear models
- 4.3: A short discussion of statistical significance
- 4.4: Sparse models
- 4.5: Variable selection
- 4.6: Model agnostic methods
- Case-study: Concept bottleneck models
- 4.7: How should explanations be evaluated?
- 4.8: Explainability of image classifiers
- 4.9: Explainability of language models

(approx. 2 weeks)

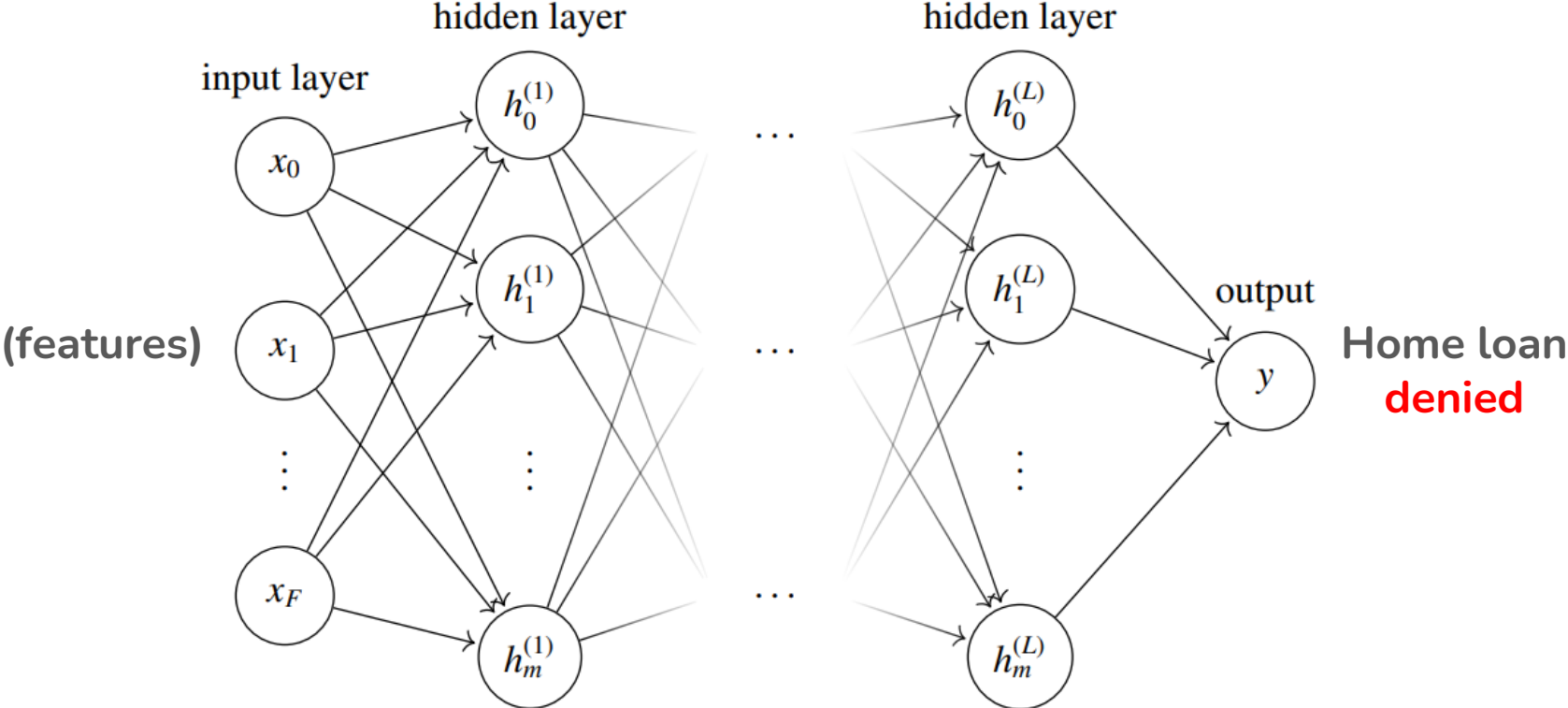
Interpretable and explainable ML

4.1: Introduction to interpretability and explainability

This section

- Introduce "interpretability" in terms of basic desiderata of interpretable models

“Black box” machine learning models



Why do we want models that offer explanations?

Why might we want to know why our home loan was rejected?

- Is the decision "fair"?
- Is there anything the user can do about it?
- Do we need to "fix" the model?

Categorization of interpretability techniques

- Global vs local
- Inherent vs post-hoc
- Model-based vs model agnostic

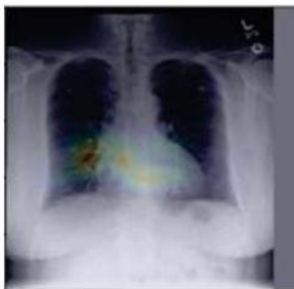
Categorization of interpretability techniques

Global vs local:

Local interpretations explain individual predictions of the model

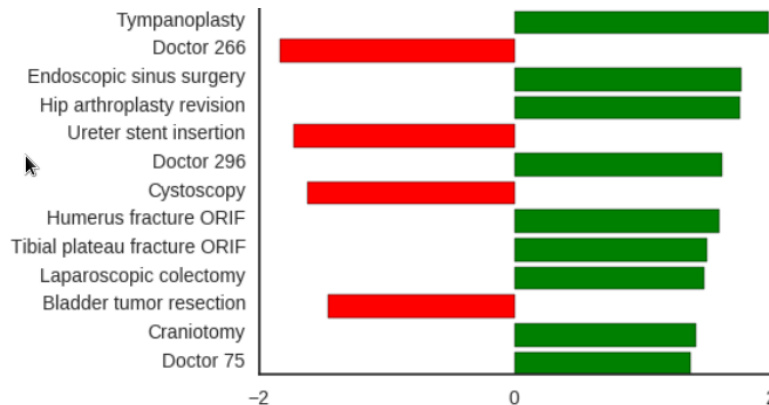
Global predictions interpret the model itself (e.g. via its parameters)

Heatmap
(global)



Real: heart size is mildly enlarged
Prediction: interval increase in heart size.

VS



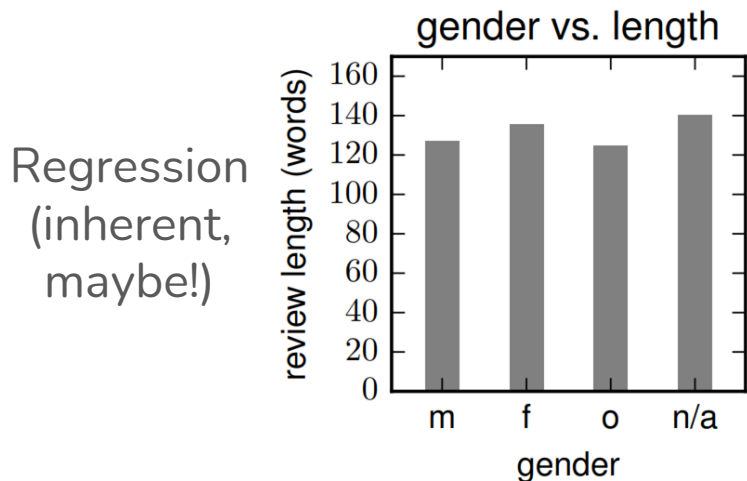
Model
weights
(local)

Categorization of interpretability techniques

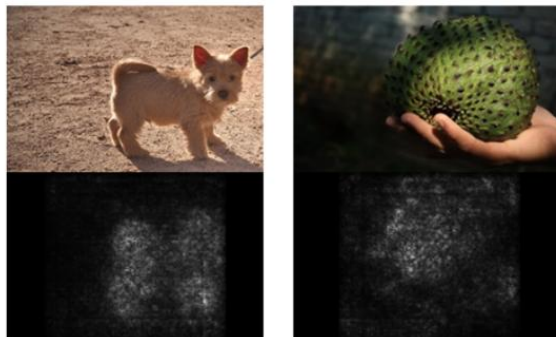
Inherent vs post-hoc

Inherent explanations are built into the design of the model

Post-hoc explanations are used to explain “black-box” methods



vs



Saliency maps
(post-hoc)

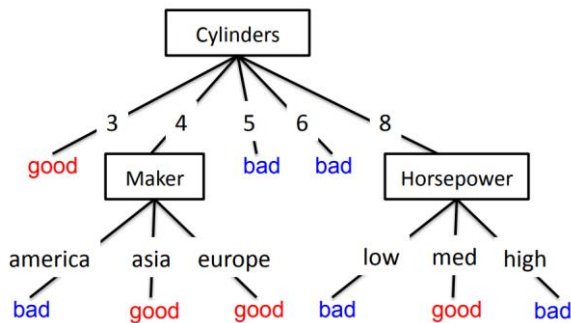
Categorization of interpretability techniques

Model based vs model agnostic

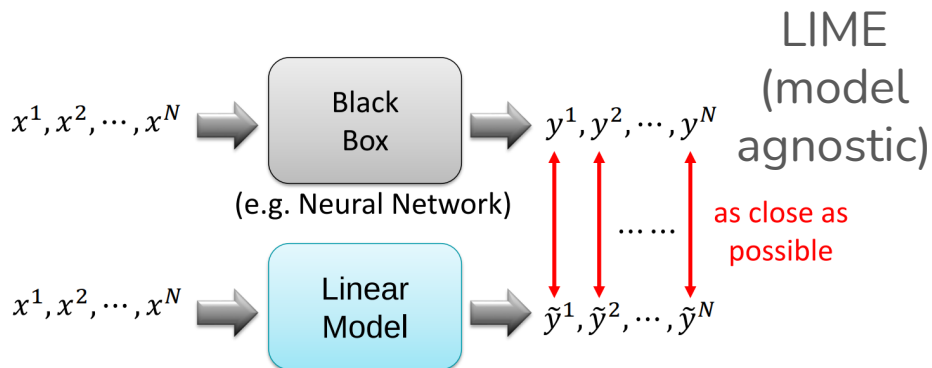
Model-based methods can explain only one class (or a few classes) of model

Model agnostic methods can explain the predictions of any model

Decision tree
(model-based)



VS



What are desirable properties of interpretable models?

Trust

- Humans are reluctant to use ML models in critical tasks
- People are generally unwilling to trust new technologies

(Q: To what extent does interpretability itself make models more "trustworthy"?)

What are desirable properties of interpretable models?

Safety

- Understand what features models are using to make their decisions
- Ensure that the model won't make decisions outside of desirable operating characteristics
- Ensure that there's a way to "fix" the model if it's not making reasonable decisions

What are desirable properties of interpretable models?

Contestability

Illustrate the reasoning used by the model, i.e., so that the model must "justify" its decisions

Give model users a chance to get different outcomes (e.g. if their homeloan was denied)

(Q: To what extent does a model like a decision tree fulfil these conditions?)

What are desirable properties of interpretable models?

Simulatability

- Would a human be able to repeat the steps taken by the model in order to understand whether those steps are "reasonable" to them?
 - I.e., is the model "simple" enough to understand?
- (Q: Does this automatically exclude e.g. image classifiers? Are humans' own decisions generally "simulatable"?)

What are desirable properties of interpretable models?

Decomposability

- Can model decisions be decomposed into simpler "steps" such that we can understand what the model is doing at each step
- (see e.g. decision trees)

What are desirable properties of interpretable models?

Algorithmic transparency

- More general term: is the model easy to "understand"; is it "informative"; etc.

What are desirable properties of interpretable models?

Functionally grounded: do explanations accurately reflect what the model is doing (sometimes known as "faithfulness" or "fidelity")?

Application grounded: does deploying an interpretable model actually result in better outcomes (more accurate, fairer, etc.)?

Human grounded: are interpretations actually "useful" to humans?

Summary

Lots of potential goals from interpretable algorithms, many of which are quite "fluffy"!

In this module we'll explore a range of "interpretable" models that exhibit each of the above criteria

Main resources

- Interpretable Machine Learning: <https://christophm.github.io/interpretable-ml-book/> (free online textbook!)
- Post-hoc Interpretability for Neural NLP: <https://arxiv.org/pdf/2108.04840> (specifically for NLP, but a good resource in general)

Interpretable and explainable ML

Case study: The mythos of model interpretability

The mythos of model interpretability

This paper (<https://arxiv.org/pdf/1606.03490>) discusses (mostly qualitatively) various aspects of interpretability research, mostly arguing that:

- *Interpretability* has no agreed upon meaning, and is used in a quasi-mathematical way
- Many of the interpretability techniques used at the time don't meet the desiderata of an interpretable method

Desiderata

Trust: A user should *trust* that a model's predictions are accurate

- Shouldn't a user reasonably have the most "trust" in a model that's the most accurate (in which case "interpretability" serves no purpose)?
- We care not just that a model is *right* but also for *which examples it is right*
- Would a model be considered "trustworthy" if it is right when humans are right (but mistaken when humans are mistaken)?

Desiderata

Transferability: “Trust” might also mean convincing a user that a model will continue to work when deployed in a new scenario that differs from its training objective.

E.g. a model predicting probability of death from pneumonia may associate “asthma” with a *low* probability of death – but this is due to the more aggressive treatment such patients receive. We would not want “has asthma” to be associated with low risk in a system that was deployed for the purposes of patient triage.

Properties of interpretable models – transparency

Simulatability: A model might be called “transparent” if a person can contemplate the entire model at once, i.e., if they could reasonably be expected to “simulate” its decisions in their head.

This is a fairly subjective definition!

Food for thought: are (wide) linear models, or (deep) decision trees any better than neural networks in this regard?

Properties of interpretable models – transparency

Decomposability: A model might be considered transparent if each part of the model (input, parameter, calculation) has an intuitive explanation, e.g. a branch of a decision tree might correspond to “*all patients with diastolic blood pressure over 150*”; or a parameter of a linear model may represent the strength associated with a particular feature.

Food for thought: do these definitions qualify linear models or decision trees as being more “interpretable”, or are potentially misleading given the complex interactions between model components (see e.g. paradoxes associated with parameter interpretation from early module)?

Properties of interpretable models – transparency

Algorithmic transparency: Can guarantees be made about the model itself, e.g. will it converge to the optimal solution of the objective it seeks to optimize? This would be true of models like least-squares regression but generally be true of (e.g.) neural networks.

Food for thought: is there a fundamental tradeoff between algorithms that are powerful and those that we can understand?

Discussion

Are linear models more/less interpretable than neural networks?

- Are the ways in which they are normally interpreted meaningful, given the potentially complex correlations among features?
- If they are much less accurate, does their better "interpretability" matter at all? Would we ever want "interpretable" but inaccurate results?
- Why not instead rely on *post-hoc* explanations of more powerful models?
- **To what standards are humans held in this regard?**

Discussion

How can claims regarding interpretability be qualified?

- One can assert that a linear model is "more interpretable" than a deep network, though this statement is not usually backed by a measurable objective
- Likewise, post-hoc, or other interpretability techniques that claim interpretability should offer evidence that it has been achieved

Discussion

Is transparency at odds with the broader objectives of AI?

- Institutions may be biased against new models (and want "interpretability" in order to trust them), but will (e.g.) lower-accuracy models ultimately be considered more "trustworthy"?

Can interpretations potentially mislead?

- Interpretations can be plausible (and subjectively trustworthy) but ultimately wrong; one could potentially develop explanations that were *deliberately* wrong with a goal of misleading

Properties of interpretable models

(lots more in paper but I've probably covered enough...)

The paper is mainly interesting just to help you think about the nuances involved in the problem, and is very readable (though certainly has a negative take on interpretability in general!)

Interpretable and explainable ML

4.2: Intro to explainability techniques for linear models

This section

- Very brief introduction to interpretability in linear models
- A few more evaluation metrics

A little about regression diagnostics

So far, we've evaluated regression models using the Mean Squared Error (MSE):

$$\frac{1}{N} \sum_i (y_i - x_i \cdot \theta)^2$$

How low should the MSE be before it is “good enough”?

Mean, Variance, and MSE

(want to get through these fairly quickly so please revise if needed!)

Mean:

$$\bar{y} = \frac{1}{N} \sum_i y_i$$

Variance:

$$\text{var}(y) = \frac{1}{N} \sum_i (y_i - \bar{y})^2$$

MSE:

$$\text{MSE}_{\theta}(y|X) = \frac{1}{N} \sum_i (y_i - x_i \cdot \theta)^2$$

R^2 statistic

$$FVU(f) = \frac{MSE(f)}{\text{Var}(y)}$$

$$(R^2 = 1 - FVU)$$

$R^2 = 0 \rightarrow$ trivial predictor

$R^2 = 1 \rightarrow$ perfect predictor

Interpreting the weights of a linear model

(1) **Code example:** see examples from 158/258, or from Module 1

(2) What do the weights “mean”

$$\text{e.g. rating} = \theta_0 + \sum_{w \in \text{text}} \theta_w \text{count}_w$$

Interpreting the weights of a linear model

$\mathcal{O}_k = w$ means that
a unit change in x_k corresponds
to a change in y of w ,
assuming all other feature values
remain the same

Interpreting the weights of a linear model

$$\text{category} \sim \Theta^1 \cdot f(\text{image}) + \Theta^2(\text{text})$$

suppose $|\Theta^1| \gg |\Theta^2|$; can we
conclude that image features are
"more important"?

Interpreting the weights of a linear model

Linear models are often thought of as being “interpretable” compared to neural networks, but these “interpretations” are so fragile that they’re fairly useless!

Even for these simple models, it seems that we need more sophisticated interpretation techniques – this is mostly what we’ll cover in this module

Interpretable and explainable ML

4.3: A short (?) discussion of statistical significance

This section

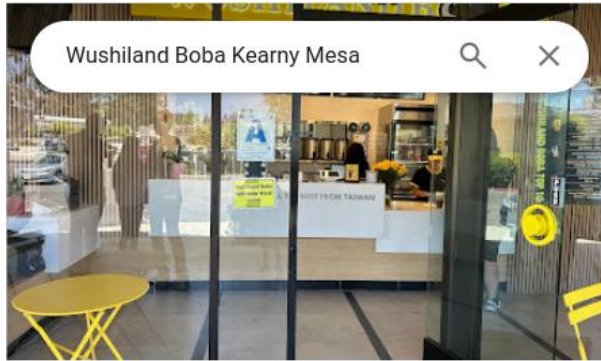
- Introduction to significance testing
- p-values and t-tests
- Significance tests for model coefficients
- Weight plots and effect plots
- Odds ratio (logistic regression)

Significance testing

- How can we determine whether a model's predictions are better than random?
- How can we determine whether a particular feature meaningfully contributes to the model's predictions?
- This is somewhat of a “classical statistics” view of the problem, which is mostly here for completeness, though (in my opinion) not all that relevant to large-scale machine learning systems

Motivating example

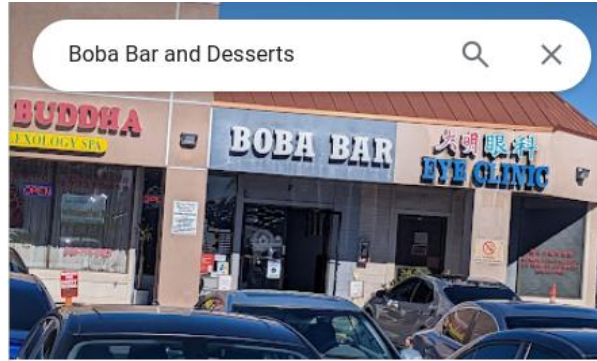
Which of these restaurants would you choose?



Wushiland Boba Kearny Mesa

4.7 ★★★★★ (7)

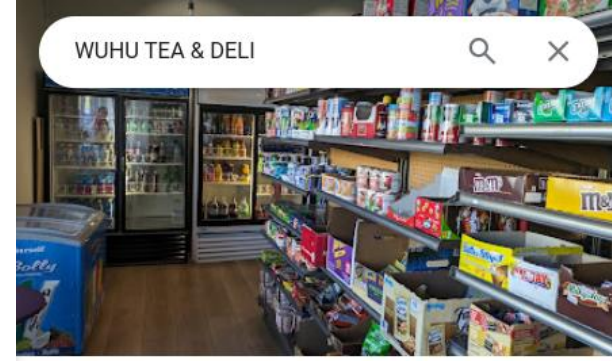
Tea house · 📍



Boba Bar and Desserts

4.6 ★★★★★ (623) · \$

Dessert shop · 📍



WUHU TEA & DELI

4.9 ★★★★★ (23) · \$

Bubble tea store · 📍

Basically, which one *really* has the higher rating?

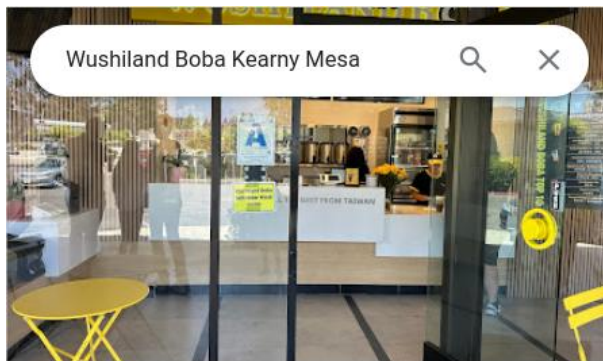
Motivating example

Basically, which one really has the higher rating?

- We don't observe the "real" average rating (or the real *distribution* of ratings)
- Instead we observe a *sample from a population*
- We'd like to determine (e.g.) *how likely one restaurant is to have a "real" higher average, based on the samples*

Motivating example

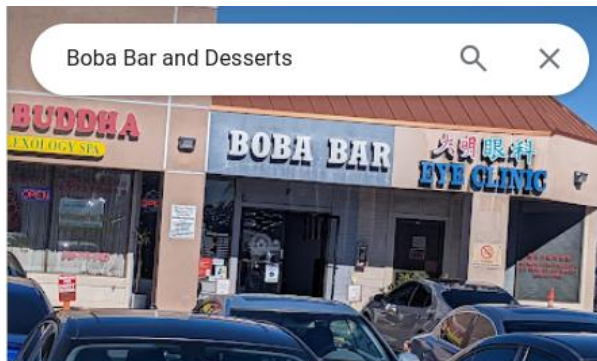
Basically, which one really has the higher rating?



Wushiland Boba Kearny Mesa

4.7 ★★★★★ (7)

Tea house · [📍](#)



Boba Bar and Desserts

4.6 ★★★★★ (623) · \$

Dessert shop · [📍](#)

How likely is it that Boba Bar “really” has better reviews? In other words: *If reviewers are a random sample from the true distribution of review scores, how likely are we to observe a difference in averages of > 0.1 ?*

Motivating example

Basically, which one really has the higher rating?

Answering this formally depends on a lot of things!

- What is the *magnitude* of the difference between the two averages?
- What is the *size* of the two samples?
- What is the *variance* of the two samples?

“Stats textbook” (or wikipedia!) example

- I can't think of any simpler example than the one on wikipedia:

Imagine you flip a coin 20 times and it lands on heads 14 times. How likely is it that the coin is fair?

- **Formally:** what is the probability that a *fair* coin would land on heads 14 or more times?

“Stats textbook” (or wikipedia!) example

Formally: what is the probability that a *fair* coin, flipped 20 times, would land on heads 14 or more times?

$$\begin{aligned} & P(14) + P(15) + \dots + P(20) \\ &= \frac{1}{2^{20}} \left[\binom{20}{14} + \binom{20}{15} + \dots + \binom{20}{20} \right] \approx 0.058 \end{aligned}$$

“Stats textbook” (or wikipedia!) example

Null hypothesis: Coin is fair; i.e, $P(\text{heads}) = 0.5$

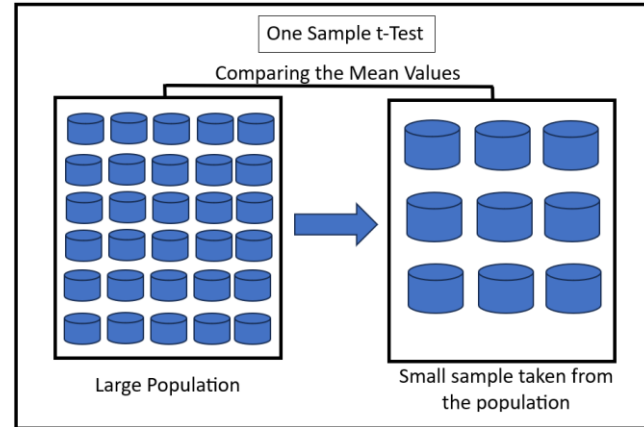
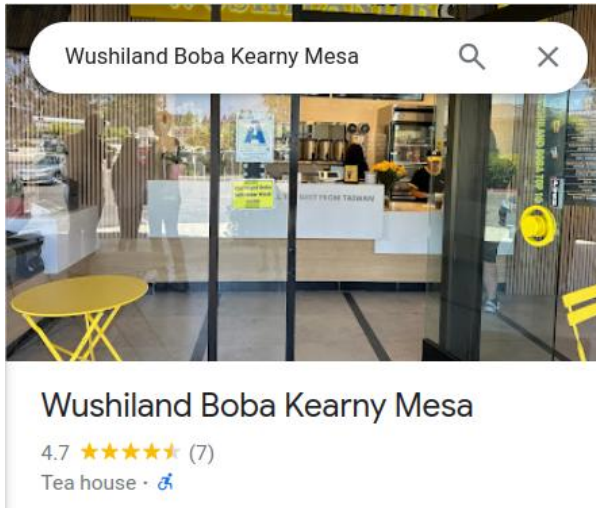
Observation: 14 out of 20 heads

(two-tailed) p-value:

$$2 \min(P(\geq 14 H), P(\leq 14 H)) \\ = 2 \min(0.058, 0.98) = 0.115$$

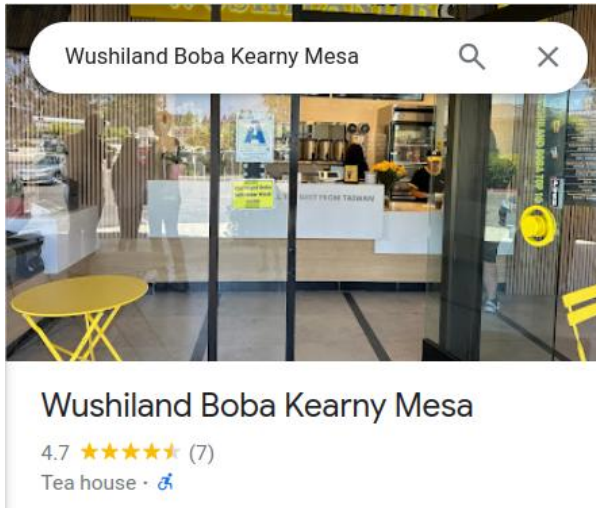
More complex example

What is the probability that Wushiland Boba Kearny Mesa has an average rating of 4.6?



More complex example

What is the probability that Wushiland Boba Kearny Mesa has an average rating of 4.6?



$$t = \frac{\bar{x} - \mu_0}{s / \sqrt{n}}$$

More complex example

The value of t will be normally distributed with mean 0 and variance 1 (see: a stats class!)

$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

standard
error

\bar{x} = av. rating

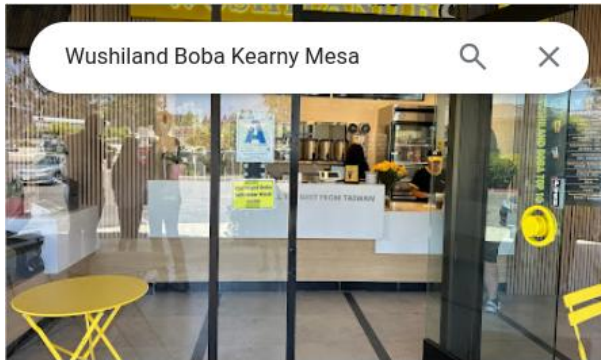
$\mu_0 = 4.6$

s = std. dev

n = # of samples

More complex example

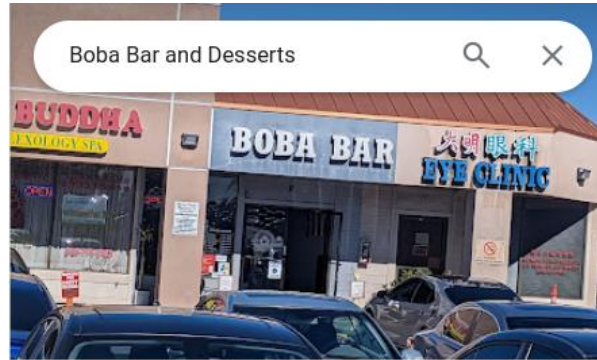
What is the probability that Wushiland Boba Kearny Mesa has a different average rating than Boba Bar and Desserts?



Wushiland Boba Kearny Mesa

4.7 ★★★★★ (7)

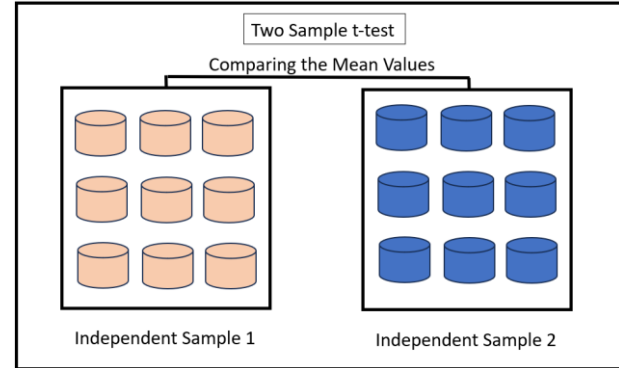
Tea house · 📍



Boba Bar and Desserts

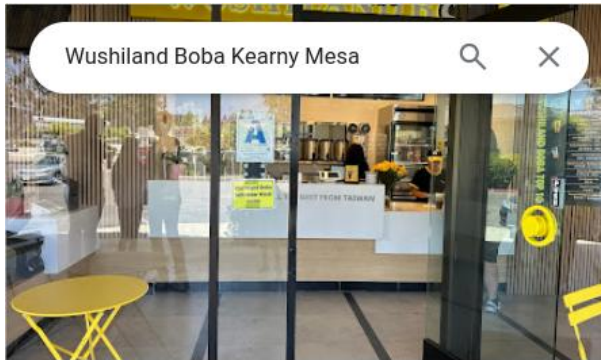
4.6 ★★★★★ (623) · \$

Dessert shop · 📍



More complex example

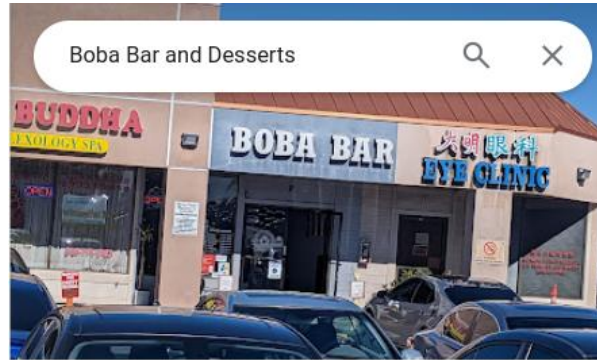
What is the probability that Wushiland Boba Kearny Mesa has a different average rating than Boba Bar and Desserts?



Wushiland Boba Kearny Mesa

4.7 ★★★★★ (7)

Tea house · 📍



Boba Bar and Desserts

4.6 ★★★★★ (623) · \$

Dessert shop · 📍

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

T-test

These particular tests are called *t*-tests, where the goal is to compare averages across small samples; several variants of these tests are used for different conditions.

History of the test is interesting; these tests were designed by a statistician at the Guinness Brewery (William Sealy Gosset), who wanted to assess the chemical properties of barley with small sample sizes (see wikipedia for history)

What about for *models*?

So far we've compared *population averages*; but in this class we're interested in *model predictions*

To assess models, we'll generally assess the "population" of model *residuals*:

$$e_i = y_i - f_{\theta}(x_i)$$

What about for *models*?

So far we've compared *population averages*. But in this class we're interested in *model predictions*.

To assess models, we'll generally assess the “population” of model *residuals*:

- Are the residuals greater than zero?
- Are the residuals of one model closer to zero (i.e., smaller in magnitude) than the residuals of another model?

See textbook (Section 3.5.1) for an example

What about for model coefficients?

(this is actually the relevant bit!)

Suppose we'd like to evaluate whether some features have "large" coefficients ("important") and others have "small" coefficients ("not important")

Note: we can't compare coefficient values directly (why?)

$$\text{Weight} = \theta_0 + \theta_1 \underset{\substack{\uparrow \\ \text{ft}}}{\text{height}} + \theta_2 \underset{\substack{\uparrow \\ \text{yes}}}{\text{age}}$$

θ_2 will be small because "units" are small

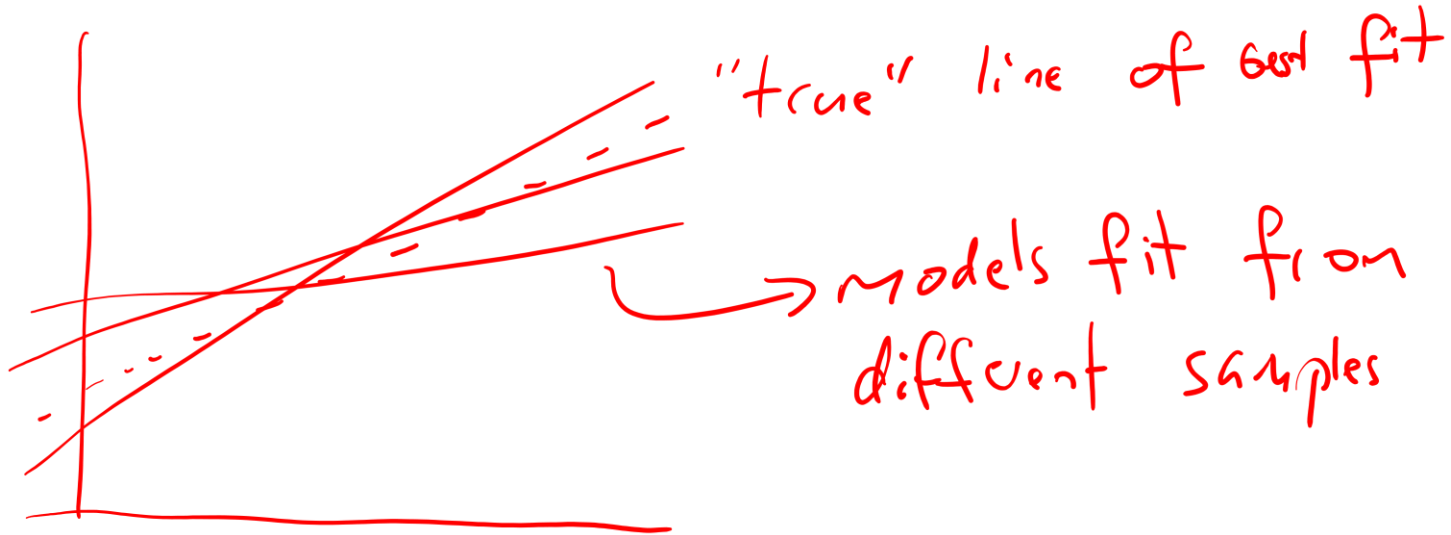
What about for model *coefficients*?

If we can't compare coefficients directly, we need to get a sense of “variance” of a coefficient (or something like that); e.g., is a coefficient “big enough” that we can confidently say that it is greater than zero, accounting for the fact that there may be more variability for larger coefficients

Again, this is critical if we want to **interpret** our model: raw feature weights are (generally) not meaningful

What about for model coefficients?

Different *samples* will generate different coefficients



What about for model *coefficients*?

Our question is generally whether we can conclude that the *true* coefficient is greater than zero, based on the coefficient estimated from the sample

That is, can we conclude that the feature is meaningfully related to the output variable?

$$\hat{\theta}_1 \stackrel{?}{>} 0$$

(though the same techniques can be used for other tests about coefficients too)

What about for model coefficients?

Put differently, is the coefficient *far enough away from zero* that a value of zero is not within the range of plausible errors due to sampling variation / randomness

For the test statistic we need to compute the standard error of a model coefficient (again from Wikipedia):

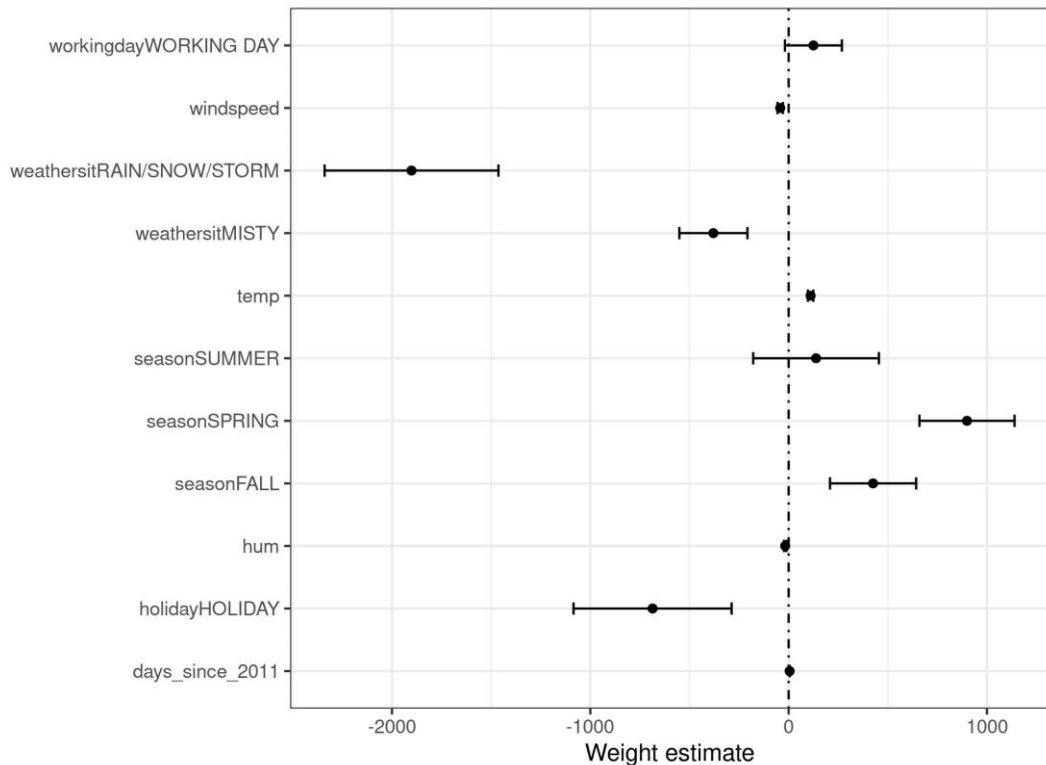
$$S_{\beta} = \sqrt{\frac{\sum_i e_i^2}{(n-2) \sum_i (x_i - \bar{x})^2}}$$

Explaining predictions using weight plots & effect plots

Weight plots: now, we can report e.g. 95% confidence intervals of feature importances

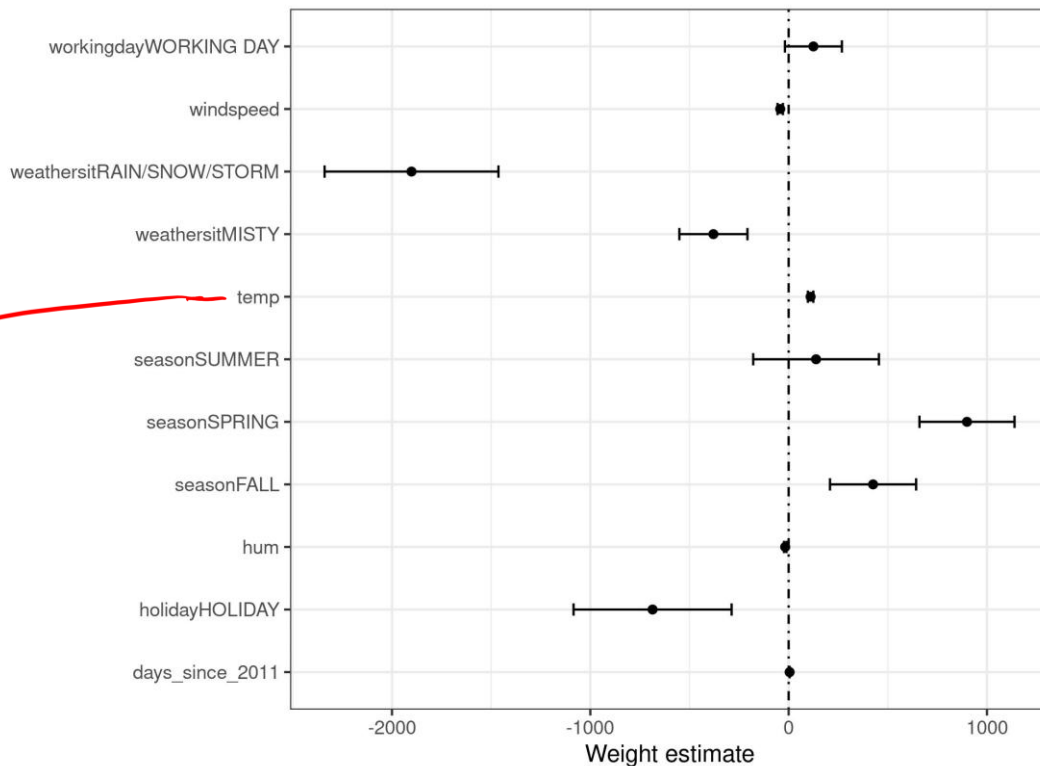
Points are parameter values (θ_i); bars indicate 95% confidence intervals

(plot based on bike rental data)



Explaining predictions using weight plots & effect plots

Small but significant!
Due to feature scales



Explaining predictions using weight plots & effect plots

Effect plots: the effect that a particular feature value has on a prediction is equal to:

$$\text{effect}_j^{(i)} = w_j x_j^{(i)}$$

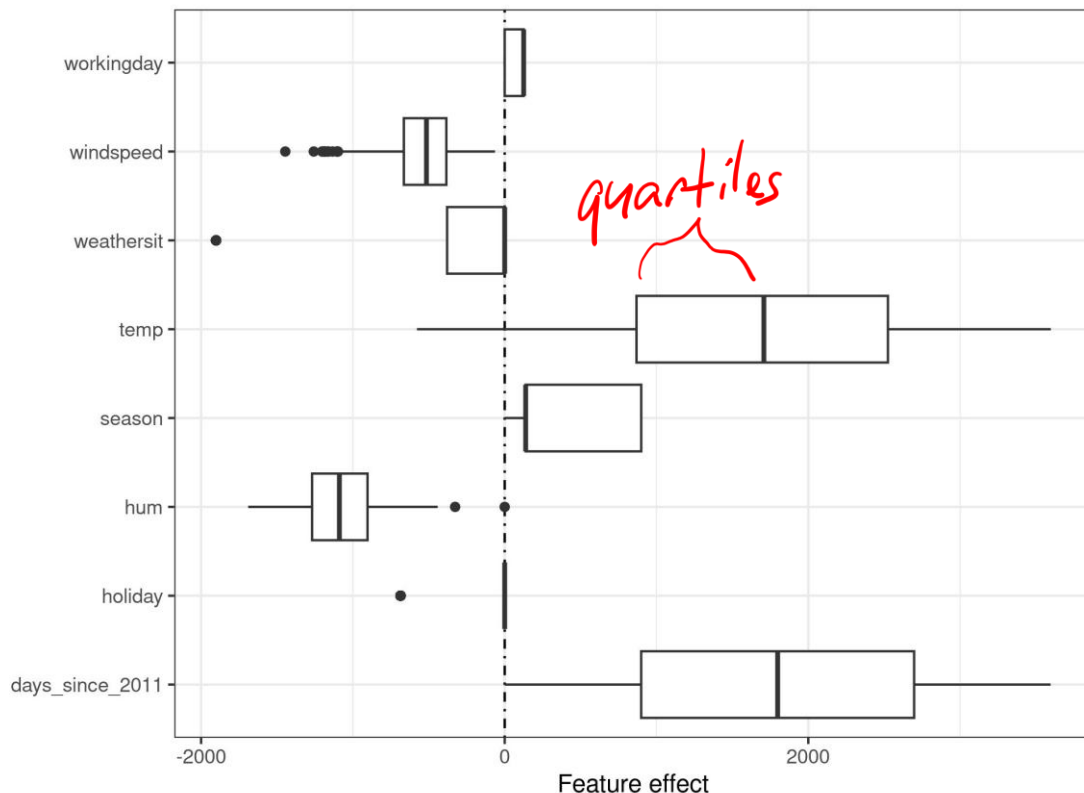
Handwritten annotations:
- $x_j^{(i)}$ is labeled as "ith data point"
- w_j is labeled as "jth feature dim."

Note: this is invariant to the particular scaling of any feature (e.g. inches vs cm)

Explaining predictions using weight plots & effect plots

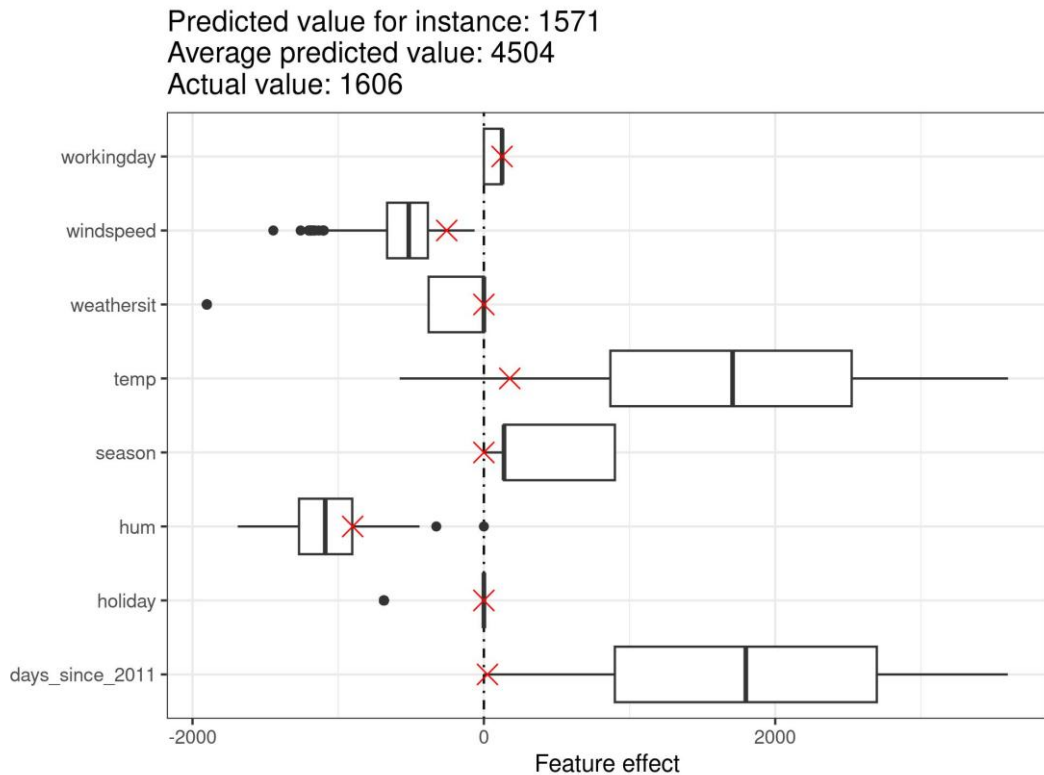
Effect plots show the distribution of these effect values to give a sense of how much *effect* different features have on the prediction

From this plot, it is clear that **temperature** and **days since 2011** have the greatest effect



Explaining predictions using weight plots & effect plots

We can use the same idea to compare the effects of features from a particular sample to the overall distribution



Do linear models make for “good” explanations?

We've formalized measuring whether features are "important", but:

- "Importance" may change or disappear depending on what other features are present (**see:** feature selection)
- "Interpretation" of a feature in a linear model is restrictive and struggles to account for correlation among features
- Interpretation is only reasonable to the extent that a linear model is actually a good model of the data in the first place
- (lots more to say in link)

Odds ratio (logistic regression)

How can we apply this type of interpretation to logistic regression?

What does the (linear) term in logistic regression correspond to?

$$p_1(y=1|x) = \frac{1}{1+e^{-x \cdot \theta}} \rightarrow \text{solve for } x \cdot \theta$$

$$p_0(y=0|x) = \frac{e^{-x \cdot \theta}}{1+e^{-x \cdot \theta}}, \quad \text{so}$$

$$\frac{p(y=1)}{p(y=0)} = \frac{1}{e^{-x \cdot \theta}} = e^{x \cdot \theta}$$

Odds ratio (logistic regression)

$$\frac{P(y=1)}{P(y=0)} = e^{x \cdot \theta} \quad \left. \vphantom{\frac{P(y=1)}{P(y=0)}} \right\} \text{odds}$$

$$x \cdot \theta = \log \left(\frac{P(y=1)}{P(y=0)} \right) \quad \left. \vphantom{\log \left(\frac{P(y=1)}{P(y=0)} \right)} \right\} \log(\text{odds})$$

Odds ratio (logistic regression)

We showed that *logistic regression is a linear model for the log odds*

(So?) how do the odds change given a unit change in a feature?

$$\frac{\text{odds}_{x_{j+1}}}{\text{odds}_{x_j}} = \frac{e^{\cancel{\theta_0 \cdot x} - \theta_j x_j + \theta_j (x_j + 1)}}{e^{\cancel{\theta_0 \cdot x}}} = e^{\theta_j}$$

$x_j \rightarrow x_{j+1}$

Odds ratio (logistic regression)

A unit change in a feature changes the odds ratio by a multiplicative factor of $\exp(\text{beta}_j)$

Example: Odds of 2 means $y=1$ twice as likely as $y=0$

A weight of 0.7 means that a unit increase multiplies the odds by $e^{0.7} \approx 2$ so the odds would change to 4

Odds ratio (logistic regression)

Not much more to say about this... there are some more specific examples in the link

Otherwise, pitfalls to interpreting weights in this way are much the same as what we saw for other types of linear models

Study points & take-homes

- Even models we thought were "inherently" interpretable (e.g. linear models) turn out to have much more complicated evaluations
- Even armed with more "sophisticated" statistical techniques, it is still hard to determine "importance" and "significance" due to e.g. correlation among features

Interpretable and explainable ML

4.4: Sparse models

This section

- Introduce the notion of model "sparsity"
- Introduce the broader problem of *variable selection*
- Motivate sparsity (making models smaller) as a form of interpretability

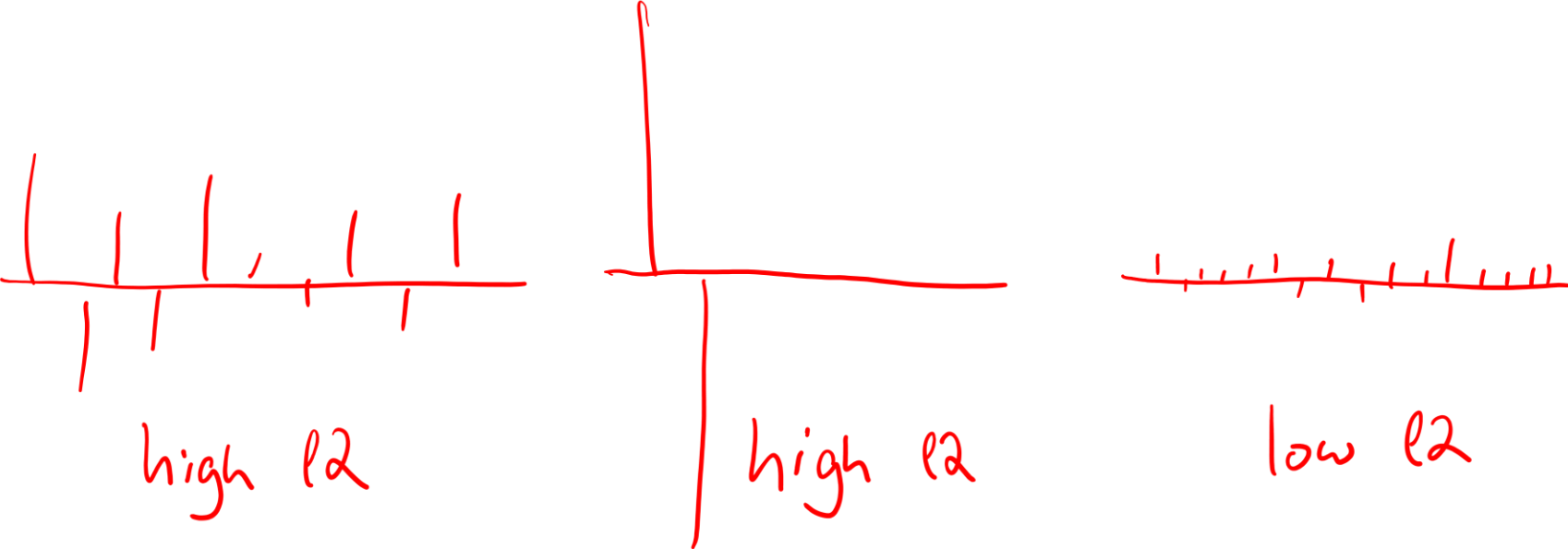
Reminder: regularization

Recall that our (regularized) linear model looks like:

$$\text{Min}_{\theta} \underbrace{\frac{1}{N} \sum_i (x_i \cdot \theta - y_i)^2}_{\text{error}} + \underbrace{\lambda \sum_k \theta_k^2}_{(\text{l2}) \text{ regularizer}}$$

Reminder: regularization

What kind of model parameters will this lead to?



Reminder: regularization

What about an l1 regularized model?

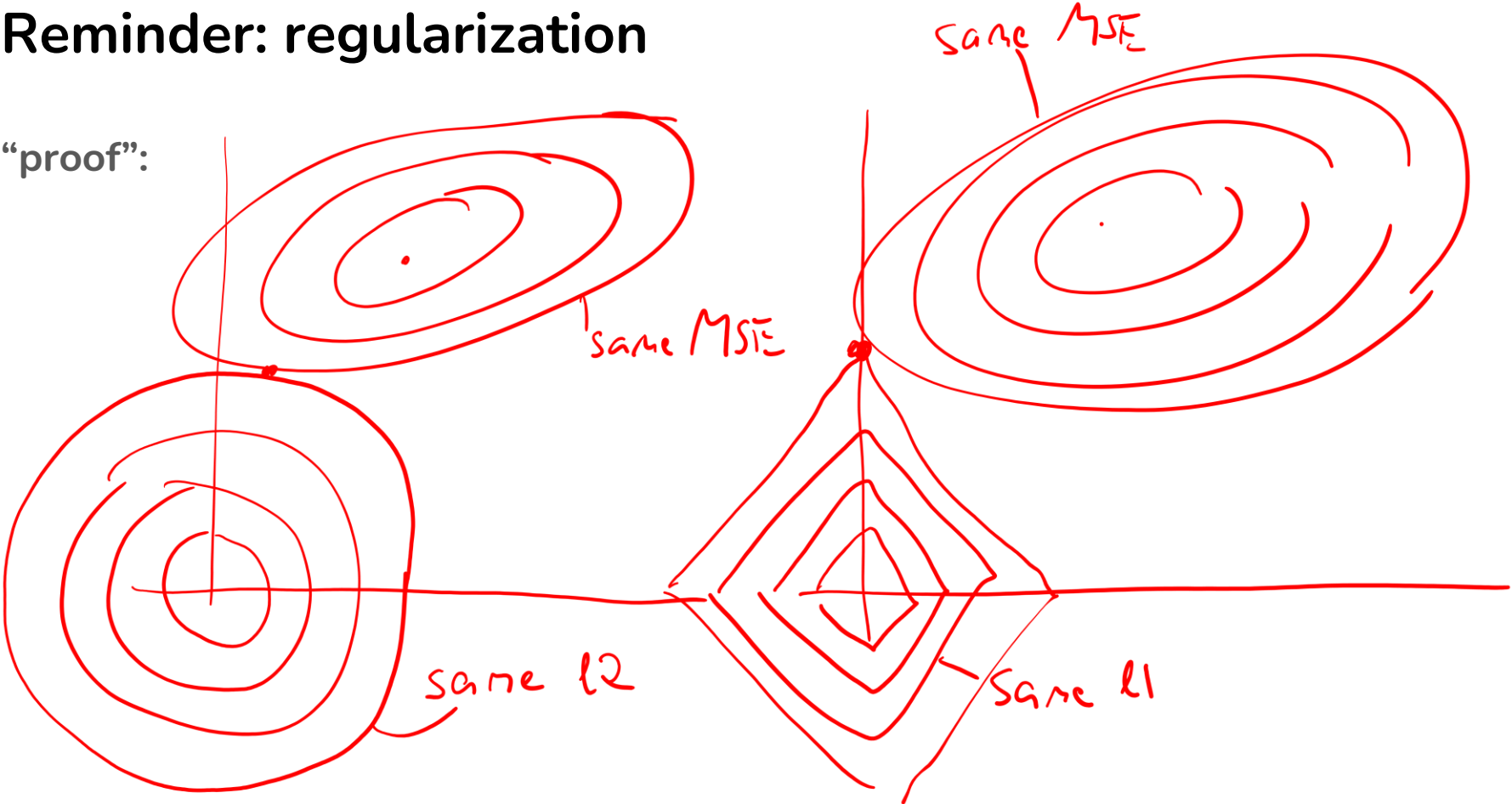


Reminder: regularization

Claim: l_2 regularized models have parameters that are roughly uniform (in magnitude); l_1 regularized models have **sparse** parameters

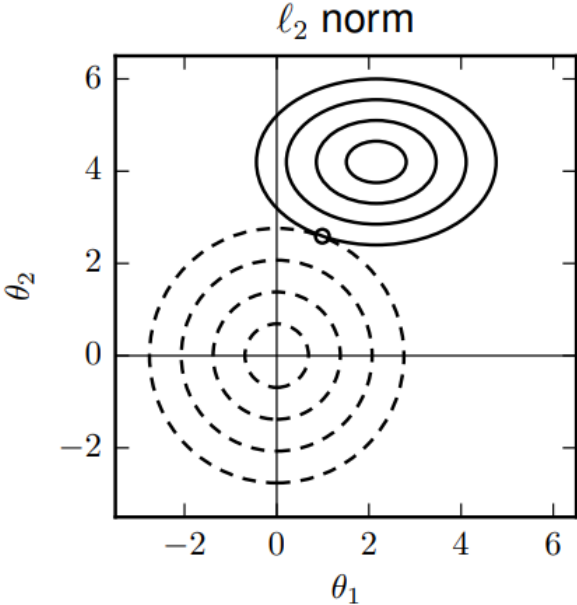
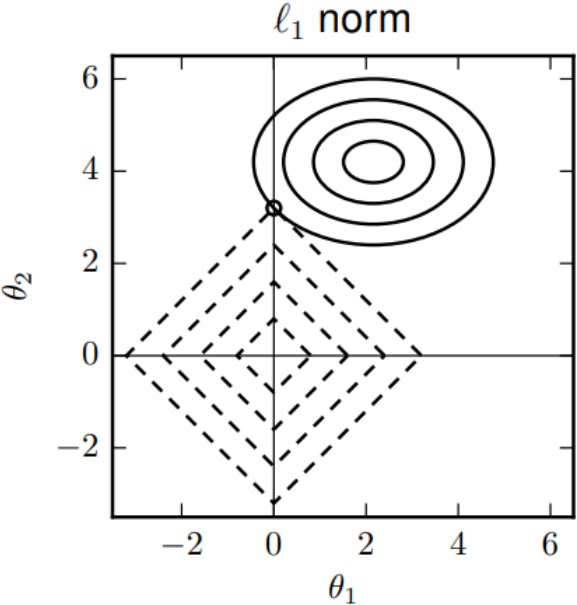
Reminder: regularization

“proof”:



Reminder: regularization

“proof” (from textbook)



Code example: housing prices

Idea: let's perform feature selection via regularization

Based on example from

https://courses.cs.washington.edu/courses/cse416/18sp/slides/L4b_lasso-regression.pdf

Use housing dataset from: <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/>

Code available in **workbook4.ipynb** workbook

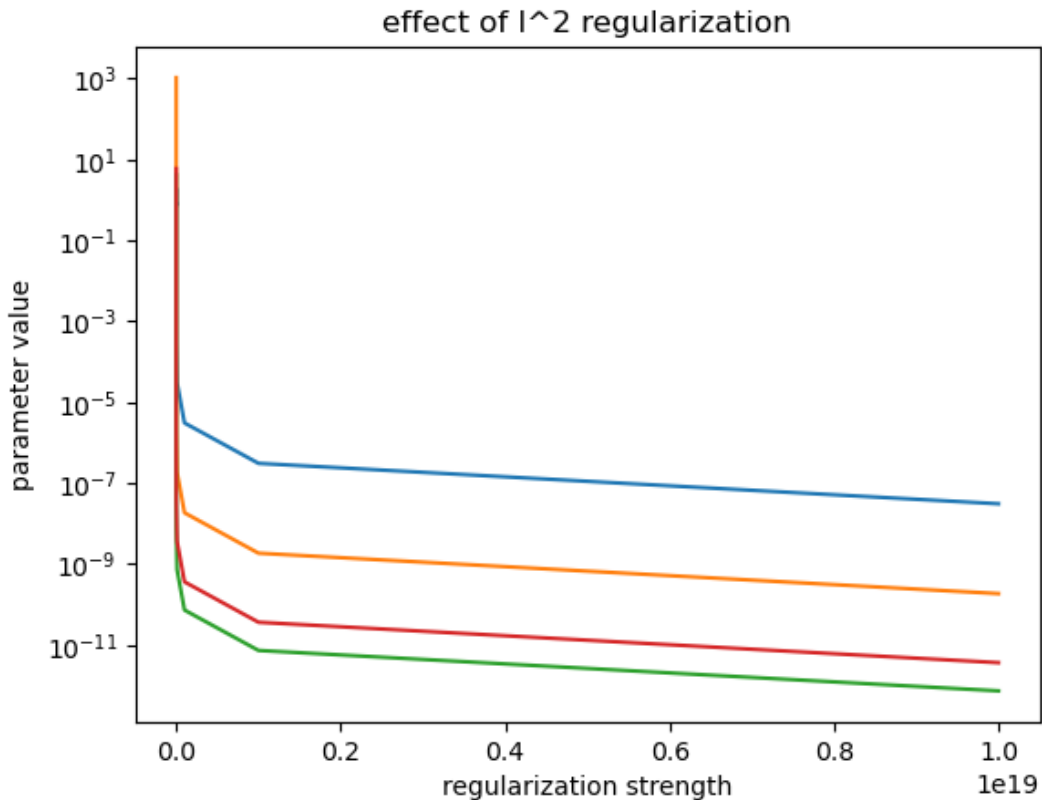
Code example: housing prices

Regression task: predict **housing sale price** using following features:

```
["LotArea", "YearBuilt", "CentralAir", "FullBath", "OverallCond", "GrLivArea",  
"BedroomAbvGr", "Fireplaces"]
```

How do feature weights change as we increase the regularization strength?

Code example: housing prices



Parameters *tend toward* zero as we increase the regularization strength, but never reach zero precisely

Code example: housing prices

Parameters *tend toward* zero as we increase the regularization strength, but never reach zero precisely

Difficult to use this model for variable selection; perhaps we could try setting some minimal parameter threshold (i.e., setting small parameters to zero), but:

- Thresholding parameters (and removing those with low weights) would be a different model than the one we actually fit
- Parameter values aren't comparable anyway, since they depend on feature scales, e.g. features with large scale will tend to have small parameters but can still be important

Code example: housing prices

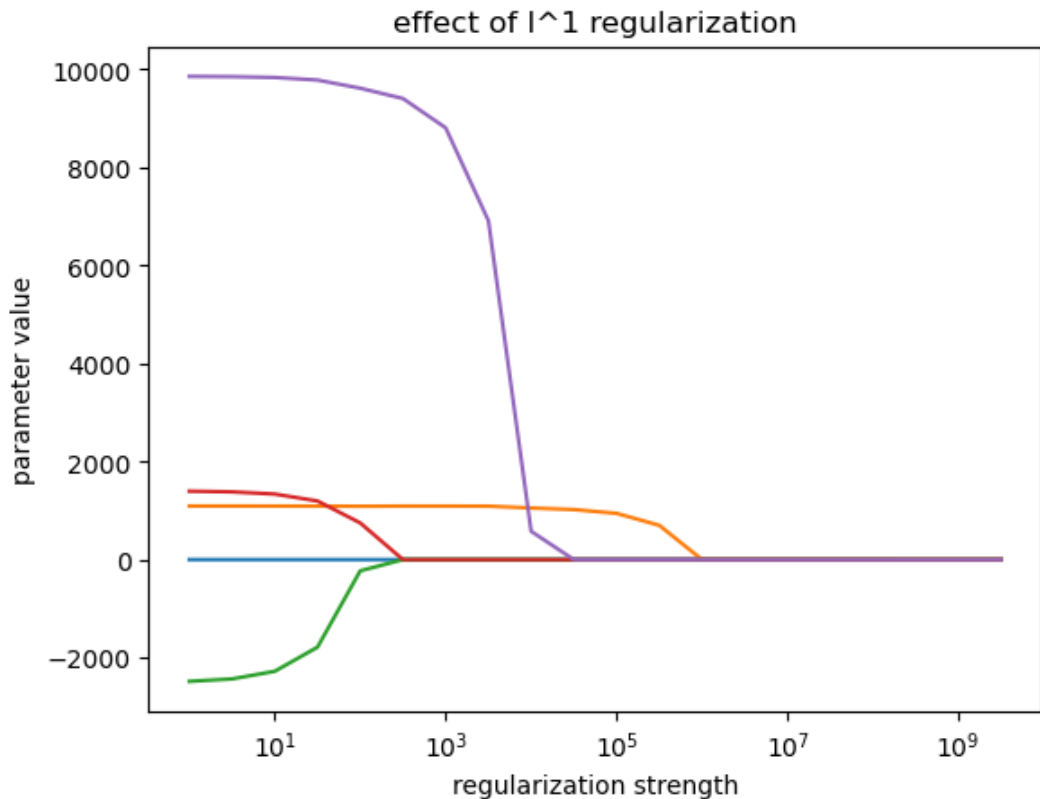
Regression task: predict **housing sale price** using following features:

["LotArea", "YearBuilt", "CentralAir", "FullBath", "OverallCond", "GrLivArea", "BedroomAbvGr", "Fireplaces"]

What if we use an l1 regularizer?

$$\sum_i (x_i \cdot \theta - y_i)^2 + \lambda \sum_k |\theta_k|$$

Code example: housing prices



Features reach *exactly zero* as we increase the regularization strength; the fitted models are *sparse*

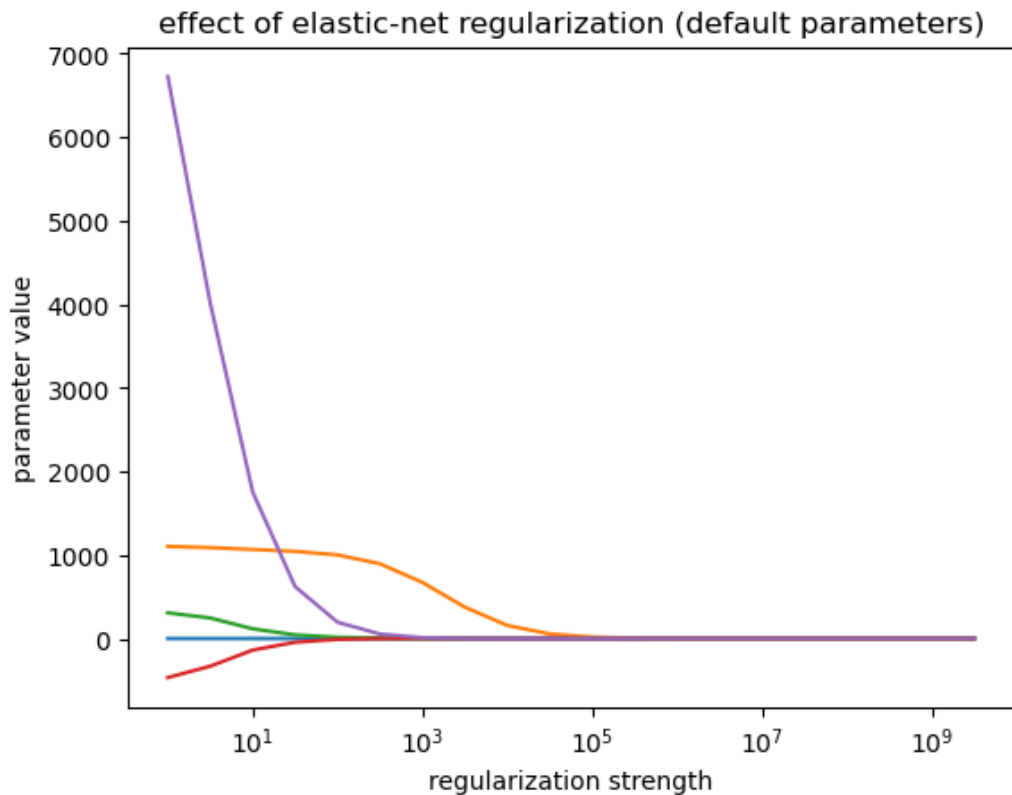
Elastic net regularization

“Limitations” of Lasso (l1 regularization):

- If the number of observations (n) exceeds the number of features (k), only at most k features will be selected (who cares?)
- If many features are highly correlated, only one will be “selected” (isn't this a good thing?)

See also “elastic net” regularization, which combines both an l1 and an l2 penalty simultaneously (though maybe not so relevant to this class)

Code example: housing prices



l1 (lasso) and l2 (ridge)
penalties combined

Sparse models

So, we saw how the use of an l1 penalty can help to induce *sparsity* in a model; arguably sparse models are more interpretable in the sense that they reduce the feature space to a small, manageable number of features (usually called simply “variable selection” rather than “interpretability”)

The way we interpret parameters doesn't change, but note that where **two correlated features might have had large parameters in an l2 model, one of them will generally “disappear” in the l1 model**; as such, our assessment of the importance of a particular feature could significantly change

Also worth considering the accuracy tradeoff, though neither the l1 nor l2 model is guaranteed to be better in any particular context

Study points & take-homes

- We saw empirically that l_1 regularizers induce sparsity, and discussed a bit about why

Interpretable and explainable ML

4.5: Variable selection

This section

- Look at the problem of *variable selection*
- Discuss a few specific variable selection strategies
- Discuss the use of variable selection as a form of interpretation

Today

- How can we experimentally determine which features (or sets of features) of a model contribute most to its performance?
- How can we use these tests to build models that only include “useful” features
- Instead of testing hypotheses etc., we’ll look at approaches that aim to maximize model performance

Why do variable selection?

- Selecting important variables (features) in a model is a form of model interpretation / explanation: it allows us to assess the contribution of model features in terms of their predictive capacity
- Such techniques are also used to demonstrate empirically that a particular feature is a useful addition to a model
- **Note:** interpretation could be different! “Significant” model parameters may not contribute much to model performance

Feature selection versus parameter significance

- **Note:** parameters that are most “statistically significantly” non-zero are not necessarily those that most contribute to a model’s performance
- Why?

Consider e.g. an extremely rare
but highly predictive attribute

Ablation tests

“Ablation” just means to remove something. An “ablation” of a model is just a model with some component (or just some of its features) removed

Comparing a model with an ablated version measures *how much that component contributes to the model (assuming all other components are still present)*

Output is similar to what it was when measuring significance: scores (e.g. change in MSE) per features, but we are measuring importance based on **predictive capacity** rather than “significance”

Ablation tests – code example

(exercise!)

Variable selection

Again note the difference between “predictive capacity” and “significance”: a rare but extremely predictive feature will have little predictive importance as measured by an ablation test

Another limitation of an ablation experiment is that features which are “important” but highly correlated **will not appear important when individually ablated**

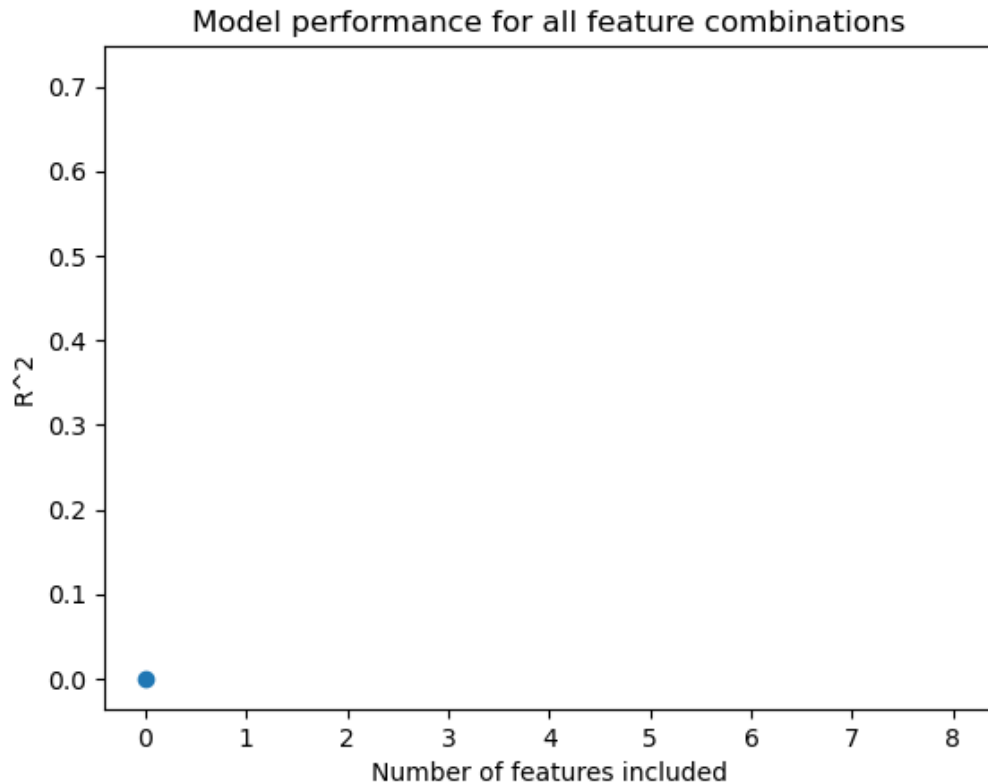
To understand the importance of features (or components) even when they are correlated, we might consider *iteratively* adding or removing them

Variable selection – code example

See `workbook4.ipynb`, based on an example from:

https://courses.cs.washington.edu/courses/cse416/18sp/slides/L4b_lasso-regression.pdf

Code example: housing prices

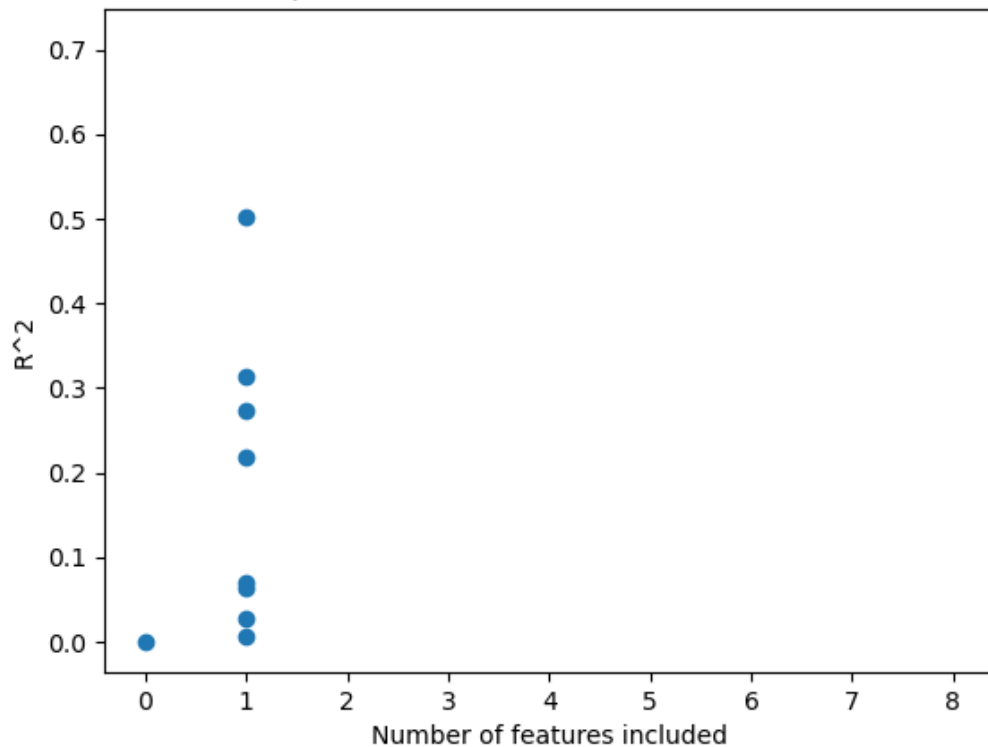


Best model with **zero** features

(will have $R^2 = 0$ by definition)

Code example: housing prices

Model performance for all feature combinations

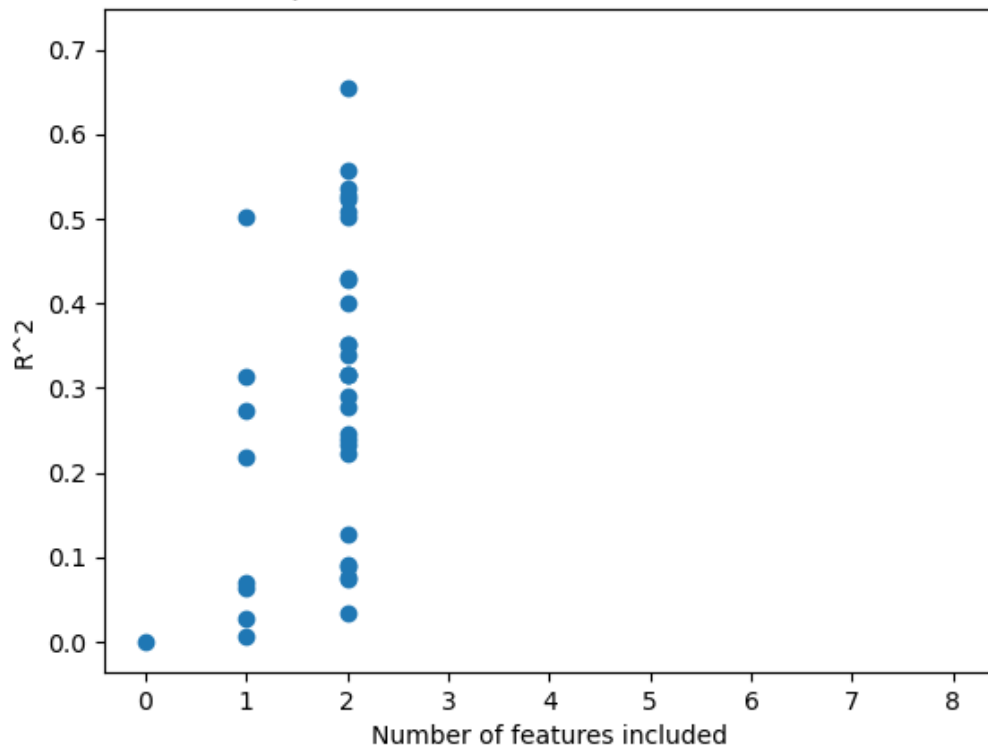


Best model with **one**
feature:

`['GrLivArea']`

Code example: housing prices

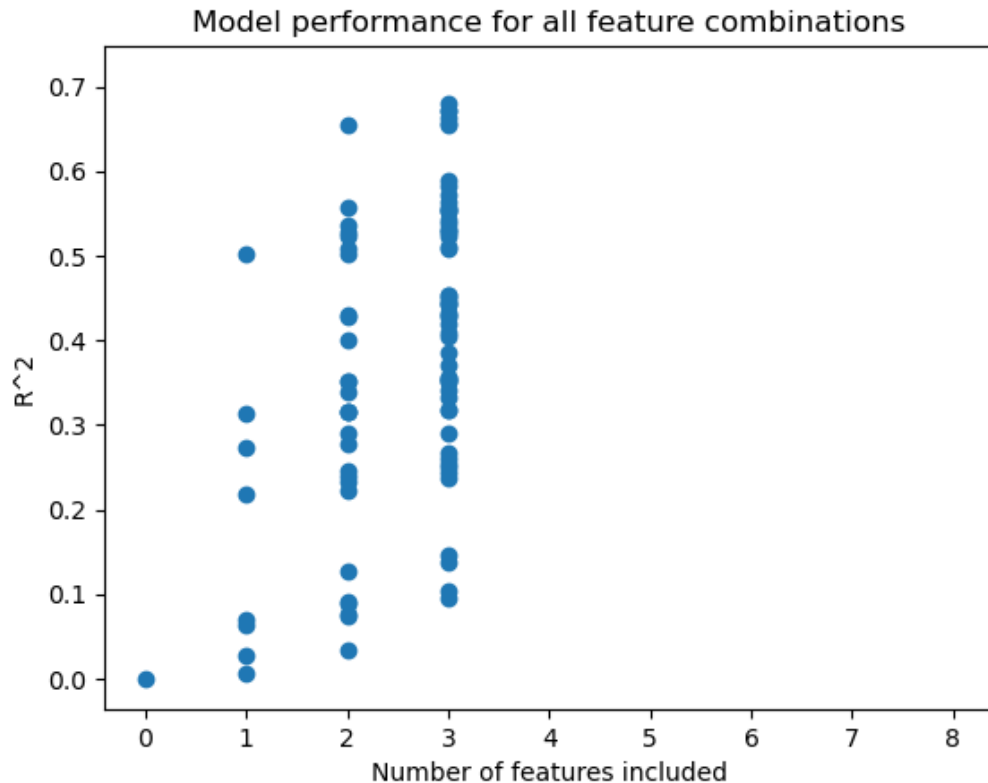
Model performance for all feature combinations



Best model with **two**
features:

`['YearBuilt', 'GrLivArea']`

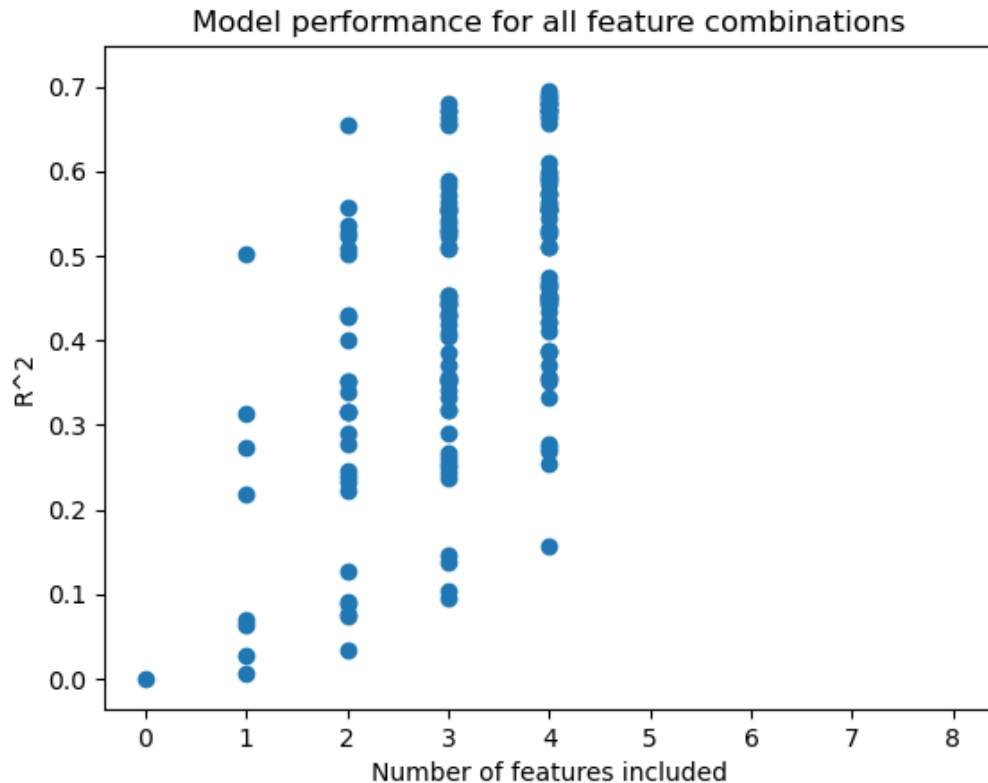
Code example: housing prices



Best model with **three** features:

```
['YearBuilt', 'GrLivArea',  
'BedroomAbvGr']
```

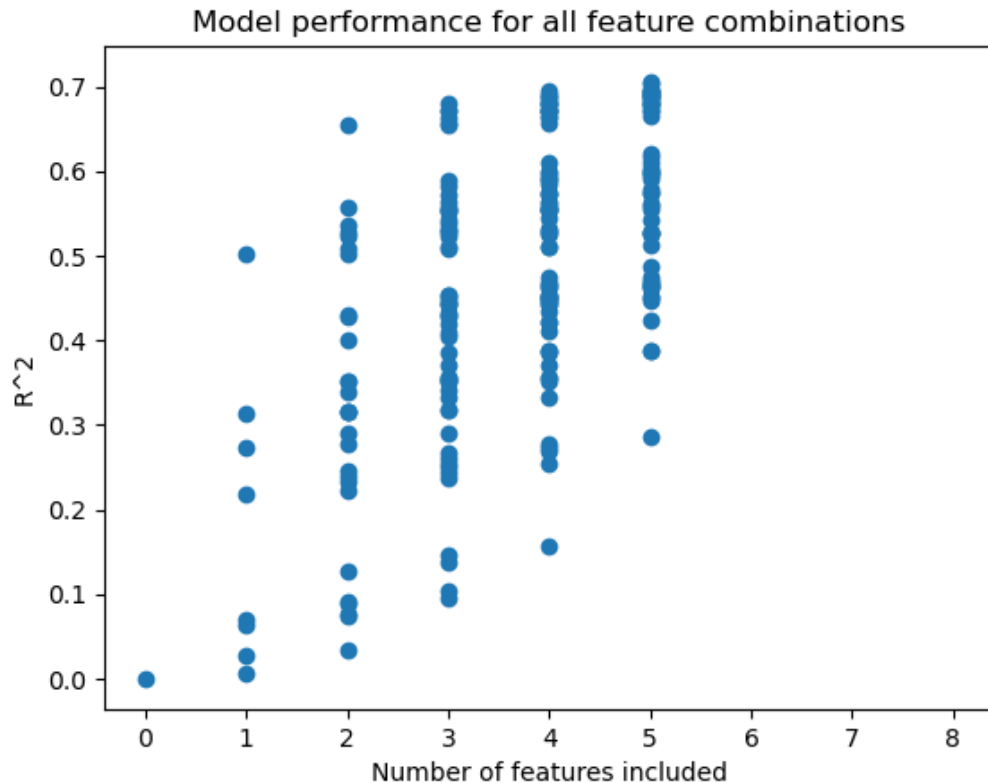
Code example: housing prices



Best model with **four** features:

```
['YearBuilt', 'OverallCond',  
'GrLivArea',  
'BedroomAbvGr']
```

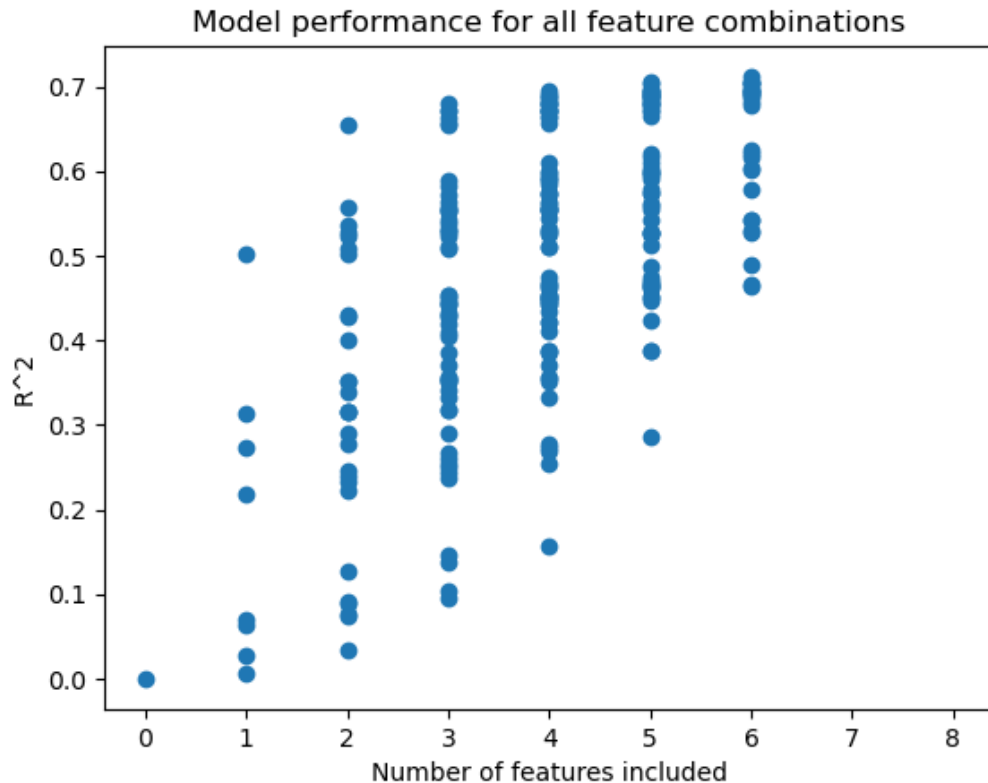
Code example: housing prices



Best model with **five** features:

['YearBuilt', 'OverallCond',
'GrLivArea',
'BedroomAbvGr',
'Fireplaces']

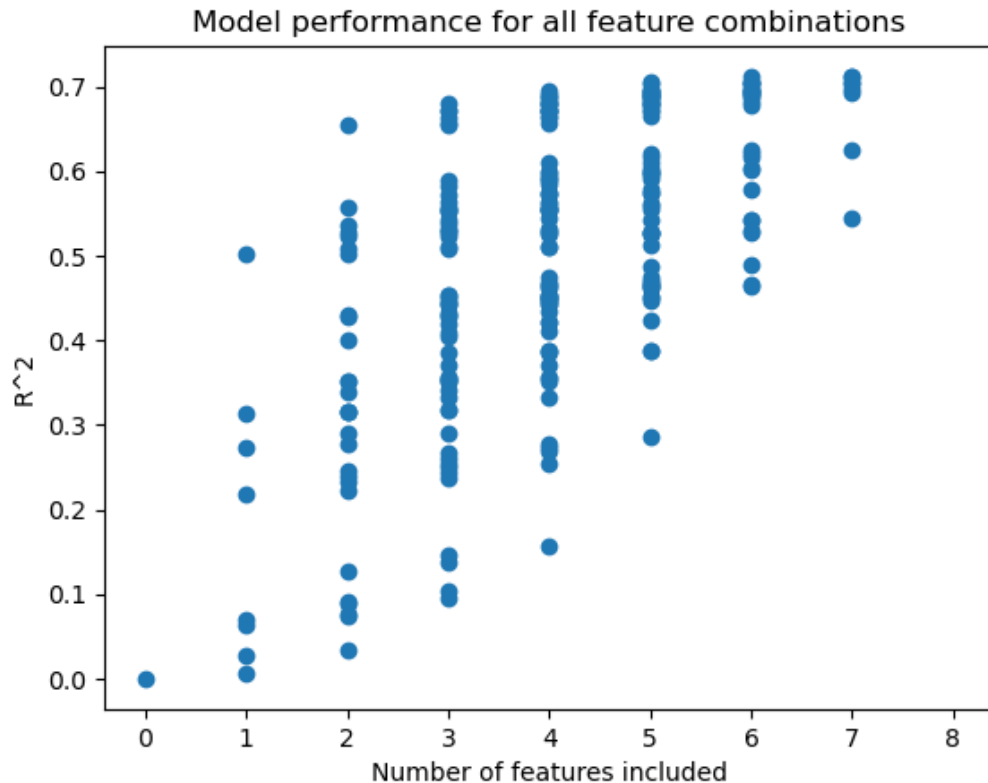
Code example: housing prices



Best model with **six** features:

['LotArea', 'YearBuilt', 'OverallCond', 'GrLivArea', 'BedroomAbvGr', 'Fireplaces']

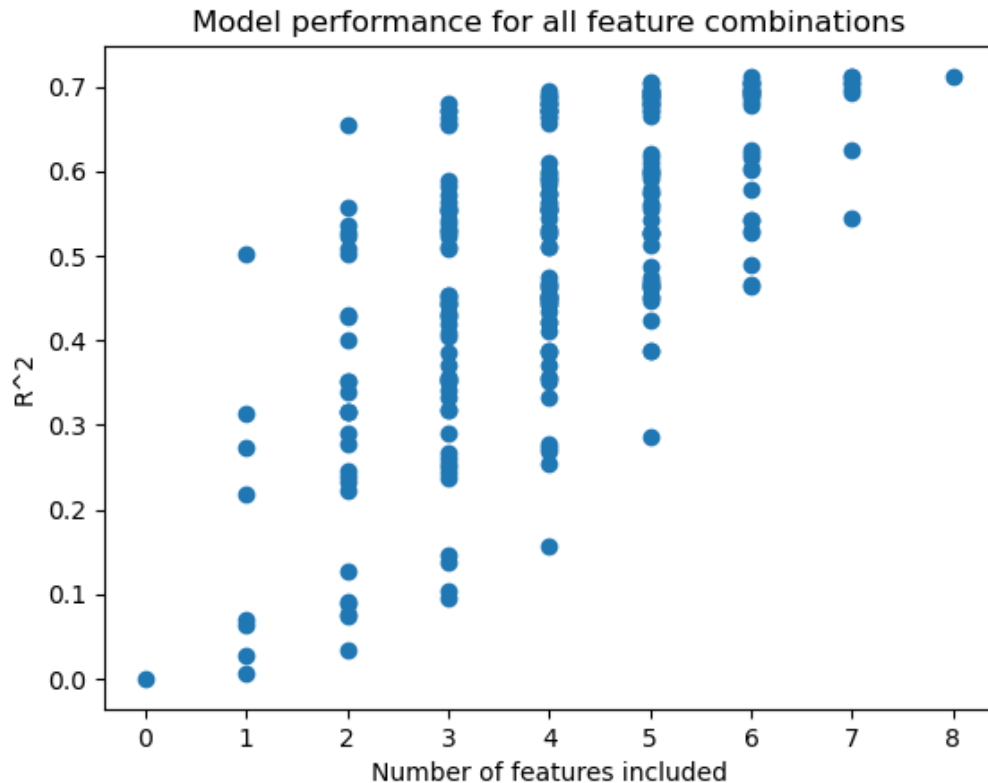
Code example: housing prices



Best model with **six** features:

['LotArea', 'YearBuilt', 'CentralAir', 'OverallCond', 'GrLivArea', 'BedroomAbvGr', 'Fireplaces']

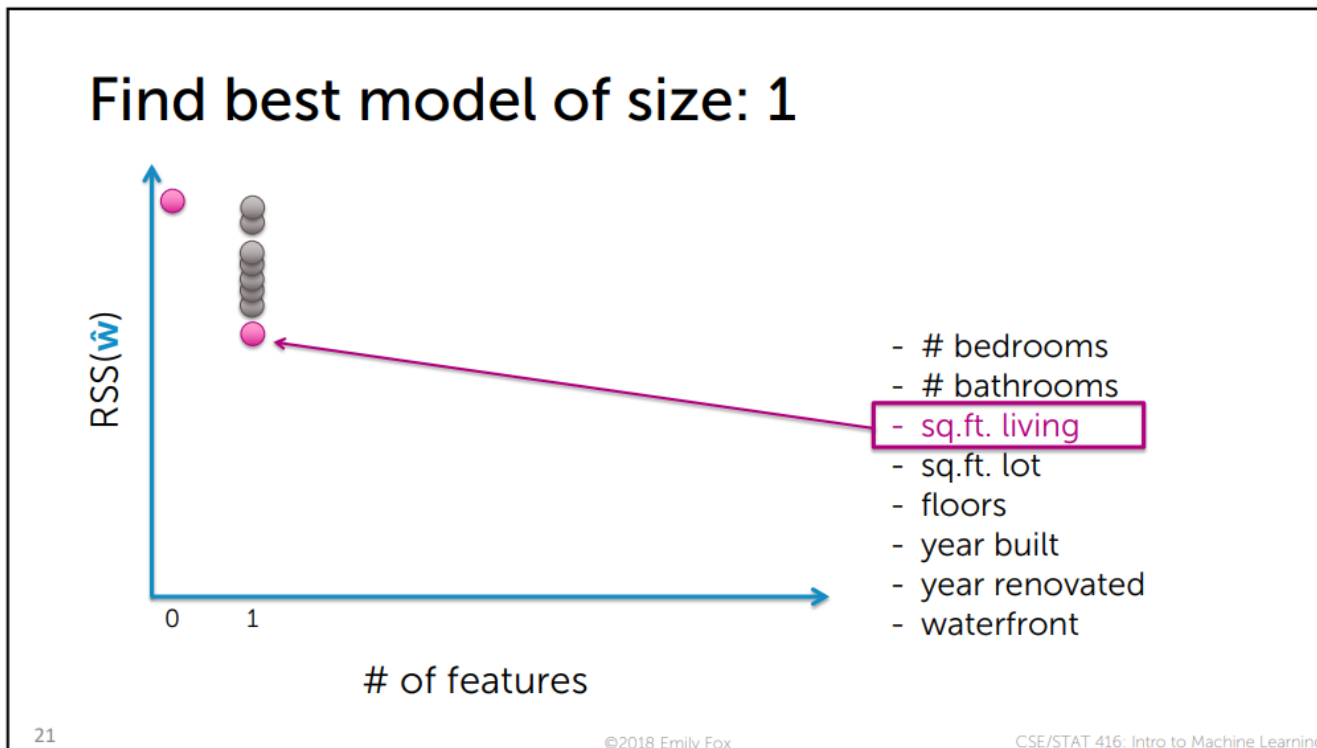
Code example: housing prices



Best model with **all** features:

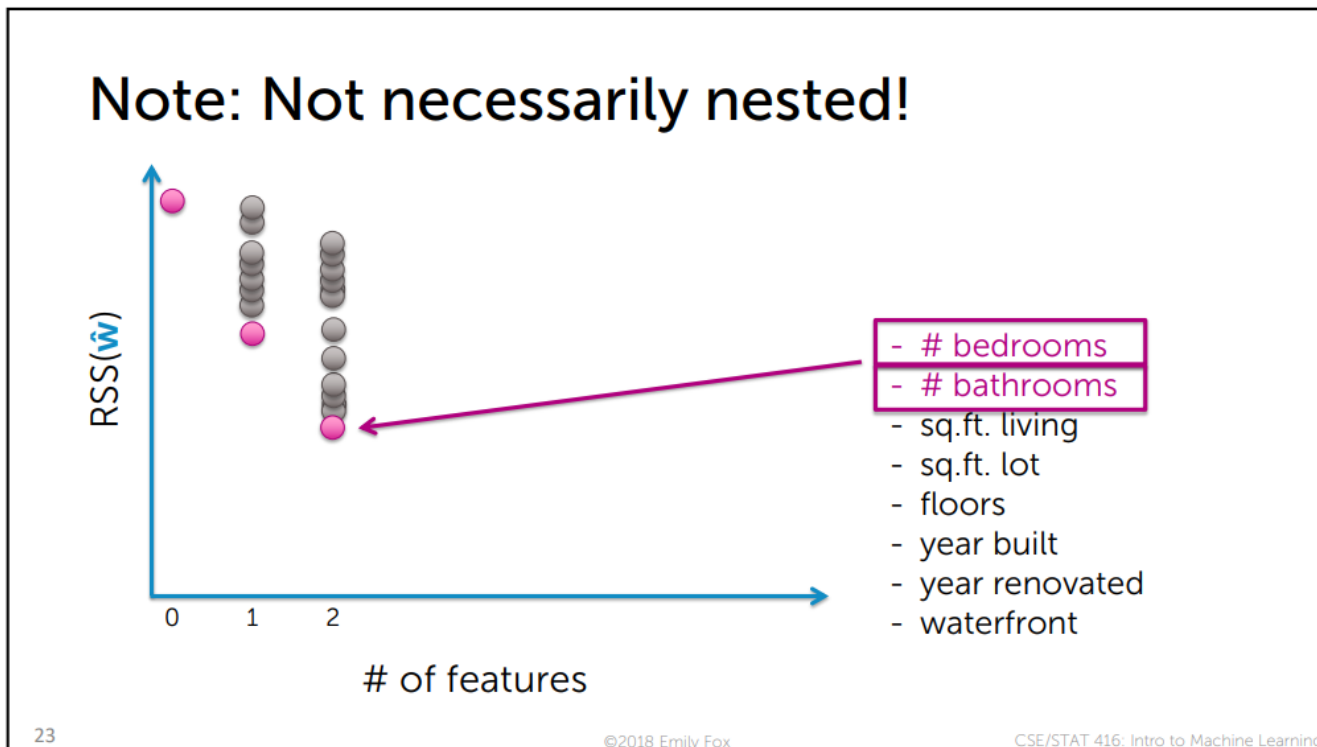
`['LotArea', 'YearBuilt', 'CentralAir', 'FullBath', 'OverallCond', 'GrLivArea', 'BedroomAbvGr', 'Fireplaces']`

(better example that I couldn't reproduce!)



see https://courses.cs.washington.edu/courses/cse416/18sp/slides/L4b_lasso-regression.pdf

(better example that I couldn't reproduce!)



Variable selection – code example

Notes:

- This type of approach would not be scalable!
- Need something more efficient (or “greedy”)

Code example: housing prices

Note: optimal feature combinations need not be nested, e.g. the best model containing two features may include features that don't appear in the best model containing three features

(this didn't happen in our example, though happens in this one:

https://courses.cs.washington.edu/courses/cse416/18sp/slides/L4b_lasso-regression.pdf)

Also: this type of approach would not be scalable! We need something more efficient (or “greedy”)

Forward selection

Forward selection applies the same type of approach, but in a greedy fashion: rather than considering all possible combinations of variables, we consider adding one variable at a time

Note: based on the nesting issue we just saw, this may not select the best possible model!

Forward selection

Start with a model **M** with no features

While (not every feature is included):

 For every feature **f** that hasn't been included yet:

 Compute the performance (e.g. accuracy) of **M + f**

 Add the feature with the best performance delta to **M**

 (keep track of the order in which features were added)

Backward selection

Backward selection applies the process in reverse: start with all features included, and iteratively discard whichever adds the least predictive capacity

Backward selection

Start with a model **M** with **all** features

While (the model still has some remaining features):

For every feature **f** in **M**:

 Compute the performance (e.g. accuracy) of **M - f**

Remove the feature with the lowest delta from **M**

(keep track of the order in which features were removed)

Forward and backward selection

Both methods produce an **ordered list of which features are the most important** (in forward selection, the most important features get added first, in backward selection the most important features are the last to be removed)

Q: *Will these two lists be the same?* Can you get a sense of any **qualitative** differences they may have?

Forward and backward selection

- **Note:** Requires fitting $O(N^2)$ models (N is the number of features)
- For a few specific models there may be efficient ways of implementing this
- Ideally we might like to try out every possible combination of features to find the best model with a certain number of features, but this would be prohibitively expensive; this is just a heuristic
- More complex heuristics exist, see e.g. Forward-Backward selection

- **Note:** This is a form of **interpretation** (which features are important?) but also a tool for **model selection** (what is a good model that includes a limited set of features?)

Study points & take-homes

- Introduced ablation tests
- Introduced forward and backward selection strategies

Interpretable and explainable ML

4.6: Model agnostic methods

This section

- Introduce the idea of "model agnostic" interpretation
- Explaining the predictions of any classifier (LIME)
- SHapley Additive exPlanations (SHAP)

Model agnostic methods

A "model agnostic" interpretability technique is a form of interpretability that doesn't depend on the specific details of the underlying model

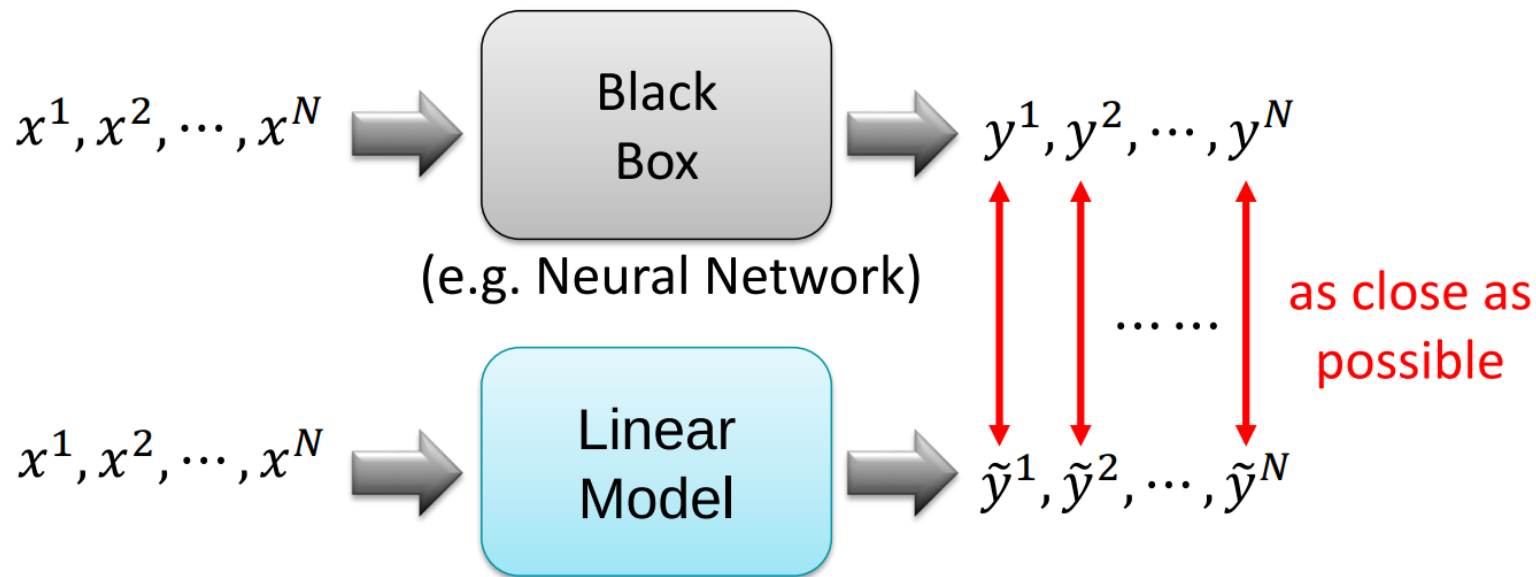
Put differently, rather than relying on a model being interpretable, we separate the task of modeling and interpretation

Explaining the predictions of any classifier (LIME)

LIME: Local interpretable model-agnostic explanations

- Some models are difficult to interpret (e.g. black-box neural network models) whereas others are easier (e.g. linear models)
- Perhaps an interpretable model can be used to approximate the uninterpretable model, and used to provide an explanation

Explaining the predictions of any classifier (LIME)

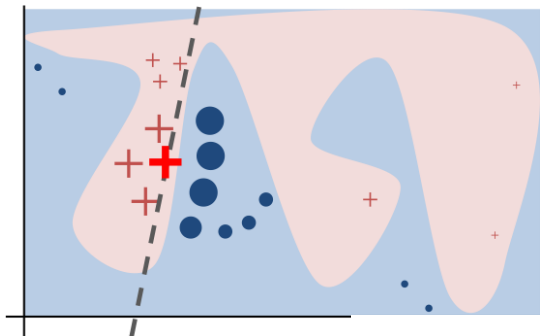


Explaining the predictions of any classifier (LIME)

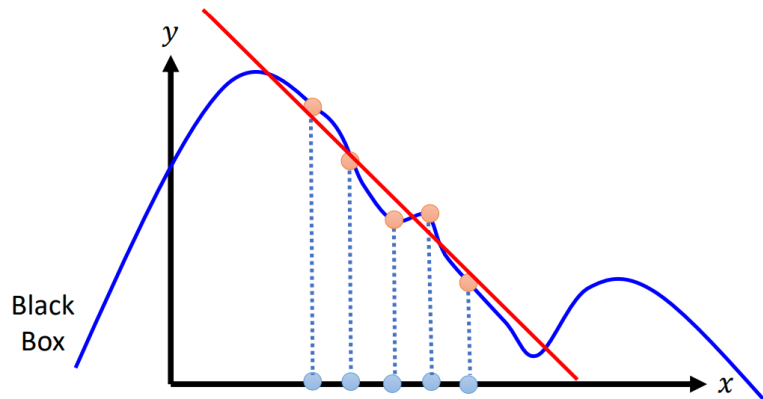
Problem: the (simple) linear model cannot approximate the neural network everywhere

However, it can possibly approximate *part* of the neural network *locally*
(such models are called "locally interpretable")

Explaining the predictions of any classifier (LIME)



- Take any data point (sample) you want to explain
- Sample nearby data points
- Fit a linear model (or some other interpretable model)
- Interpret the linear model



In theory, the linear model should “explain” part of the complex decision boundary

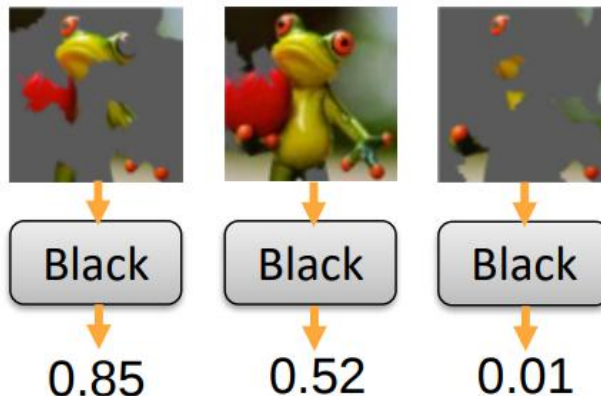
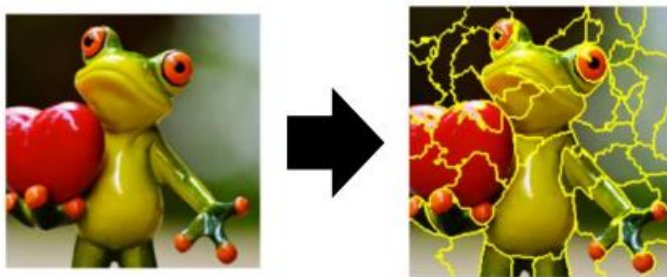
see <https://homes.cs.washington.edu/~marcotcr/blog/lime/>

see [https://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/XAI%20\(v7\).pdf](https://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/XAI%20(v7).pdf)

Explaining the predictions of any classifier (LIME)

Image example:

- Split image into superpixels (segments)
- Randomly delete some segments
- Use black box to compute $p(\text{frog})$

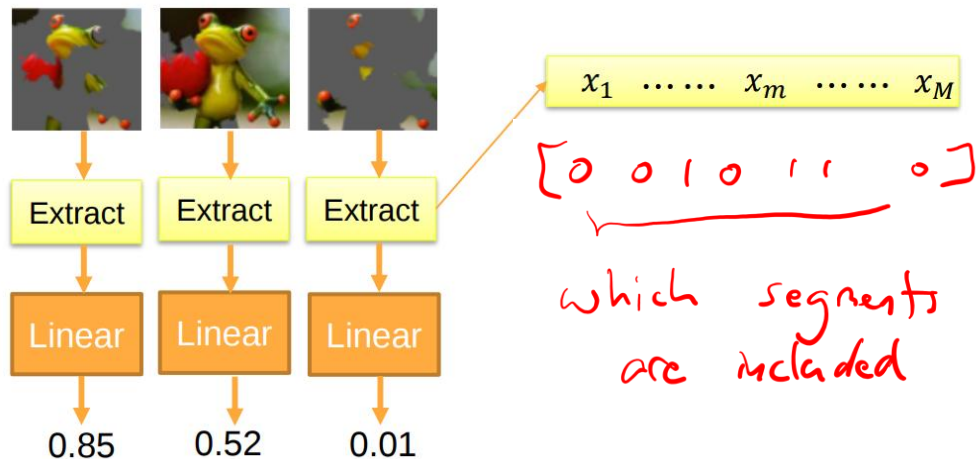


Explaining the predictions of any classifier (LIME)

Fit with linear model:

$$y = x \cdot \theta$$

$\theta_i \rightarrow$ how much is the i^{th} segment relevant to "frog"



Explaining the predictions of any classifier (LIME)

Above is perhaps unsophisticated (depends on using image classifiers based on segments!) but gives a general sense of how one might design an interpretation scheme based on local perturbations

We'll see other explainability techniques specific to images later

Explaining the predictions of any classifier (LIME)

Code example: <https://marcotcr.github.io/lime/tutorials/Lime%20-%20basic%20usage%2C%20two%20class%20case.html>

SHapley Additive exPlanations (SHAP)

LIME can have issues in terms of **stability**, and is also not “**efficient**”: The contribution of the individual features, if added together, will not correspond to the original prediction (it is, after all, just an approximation of the original model)

SHapley Additive exPlanations (SHAP)

Shapley values are an idea from game theory to distribute the profits from a game among a coalition of (cooperative) participants in a way that is proportional to their individual contribution.

SHapley Additive exPlanations (SHAP)

Example: You (Player 1) and your friend (Player 2) enter a Kaggle competition, and win \$10,000. Your friend wants to split the money evenly, but you believe you have superior ML skills and thus deserve a larger share. How should the winnings be divided “fairly”?

SHapley Additive exPlanations (SHAP)

Suppose you could go back in time and determine what your winnings *would have been if either of you had not been on the team*. You determine the following coalition values for different possible teams:

$$C_{12} = 10000$$

$$C_1 = 7500$$

$$C_2 = 5000$$

$$C_0 = 0$$

Player 1 deserves more. But how to split?

SHapley Additive exPlanations (SHAP)

Player 1 deserves more. But how to split?

Idea: compute the **expected marginal contribution** of each player. That is, compute the weighted average of each player's contribution to all coalitions that they could join. (This is a lot of words, but not much going on)

SHapley Additive exPlanations (SHAP)

Player 1 deserves more. But how to split?

Example: Player 1 could join a coalition with Player 2, or Player 1 could join a “coalition” with the empty set:

$$C_{12} = 10,000$$

$$C_1 = 7,500$$

$$C_2 = 5,000$$

$$C_0 = 0$$

$$C_{12} - C_2 = 5000$$

$$C_1 - C_0 = 7500$$

$$av. = \$6250$$

SHapley Additive exPlanations (SHAP)

Player 1 deserves more. But how to split?

Example: Same for Player 2:

$$C_{12} = 10,000$$

$$C_1 = 7,500$$

$$C_2 = 5,000$$

$$C_0 = 0$$

$$C_2 - C_1 = 2500$$

$$C_2 - C_0 = 5000$$

$$\text{av. } \$3750$$

Note: sum of players' contributions adds to the total (10,000)

SHapley Additive exPlanations (SHAP)

Can generalize this to any number of players:

$$\phi_i = \sum_{S \subseteq \{1, \dots, p\} \setminus \{i\}} \underbrace{\frac{|S|!(p-|S|-1)!}{p!}}_{\text{Weight}} \underbrace{[val(S \cup \{i\}) - val(S)]}_{\text{Marginal contribution of player } i \text{ to coalition } S}$$

$$\varphi_i(v) = \frac{1}{\text{number of players}} \sum_{\text{coalitions excluding } i} \frac{\text{marginal contribution of } i \text{ to coalition}}{\text{number of coalitions excluding } i \text{ of this size}} \quad (\text{wikipedia})$$

SHapley Additive exPlanations (SHAP)

Example for three players: suppose we have the following coalition values:

$$\begin{aligned} \mathbf{C}_{123} &= 10,000 \\ \mathbf{C}_0 &= 0 \end{aligned}$$

$$\begin{aligned} \mathbf{C}_{12} &= 7,500 \\ \mathbf{C}_{13} &= 7,500 \\ \mathbf{C}_{23} &= 5,000 \end{aligned}$$

$$\begin{aligned} \mathbf{C}_1 &= 5,000 \\ \mathbf{C}_2 &= 5,000 \\ \mathbf{C}_3 &= 0 \end{aligned}$$

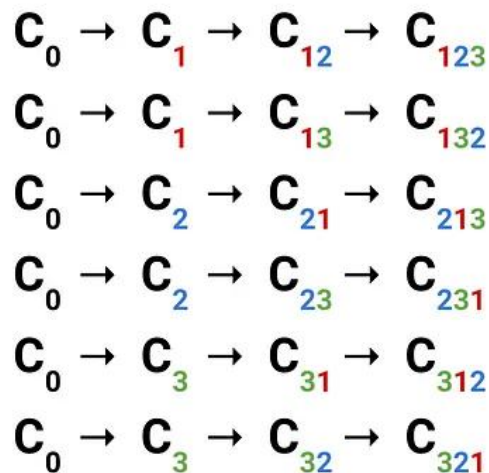
SHapley Additive exPlanations (SHAP)

What is the marginal contribution of Player 1?

$$\begin{aligned} C_{123} - C_{23} &= 5,000 && 5,000 * (1/3) + 2,500 * (1/6) \\ C_{12} - C_2 &= 2,500 && + 7,500 * (1/6) + 5,000 * (1/3) \\ C_{13} - C_3 &= 7,500 && = \mathbf{\$5,000} \\ C_1 - C_0 &= 5,000 \end{aligned}$$

SHapley Additive exPlanations (SHAP)

Different ways of forming a three-player coalition:



SHapley Additive exPlanations (SHAP)

Summary: The marginal contributions of Player i are weighted by the probability that they make that contribution. The contributions are then weighted by this probability. In this way, we're computing the **expected marginal contribution** of each player.

SHapley Additive exPlanations (SHAP)

Axioms: The Shapley value is the **only** “efficient” value that satisfies the following axioms:

Symmetry: Two players are considered “interchangeable” if they make the same contributions to all coalitions. *If two players are interchangeable they should be given an equal share of the game’s total value.*

Null player property: If a player makes zero marginal contribution to all coalitions then they get none of the total value.

Additivity: If we combine two (independent) games, then a player’s contribution is the sum of contributions for the two individual games.

From Shapley to SHAP

How does this apply to ML?

Value of a game → model prediction

Players → feature values

(we'll refer to “feature values” as “features” but note that it is the *values* and not the *features* that contribute to the outcome)

From Shapley to SHAP

Example: suppose we want to predict income from age (feature 1) and degree (feature 2):

$$f(x_1, x_2) = 200x_1 + 1000x_2$$

From Shapley to SHAP

Example: suppose we want to predict income from age (feature 1) and degree (feature 2).

Also need to assume a *distribution* over feature values:

$$f(x_1, x_2) = 200x_1 + 1000x_2$$

$$\text{age} \rightarrow x_1 \in [18, 60]$$

$$\text{degree} \rightarrow x_2 \in \{0, 1\}$$

From Shapley to SHAP

Example: suppose we want to predict income from age (feature 1) and degree (feature 2).

Q: What is the value of a “coalition” containing *both* feature (values)?

$$f(x_1, x_2) = 200x_1 + 1000x_2$$

$$\begin{aligned} \text{val}_x(\{1, 2\}) &= f(20, 1) \\ &= 200(20) + 1000(1) \\ &= 5000 \end{aligned}$$

From Shapley to SHAP

Q: What is the value of a “coalition” containing just the first feature?

A: Need to marginalize over the second feature:

$$\begin{aligned}val_x(\{1\}) &= \int f(20, x_2) dP_{x_2} \\ &= \sum_{i=0}^1 f(20, i) P(x_2 = i) \\ &= (200(20) + 1000(0))(0.5) + \\ &\quad (200(20) + 1000(1))(0.5) \\ &= 4500\end{aligned}$$

From Shapley to SHAP

Q: What is the marginal contribution of feature 2 to a coalition containing feature 1 (i.e., the contribution of “degree=1” to the prediction)?

$$\begin{aligned} \text{val}_x(\{1, 2\}) - \text{val}_x(\{1\}) &= 5000 - 4500 \\ &= 500 \end{aligned}$$

From Shapley to SHAP

(to compute the Shapley value, still need to compute the contribution of feature 2 to a coalition containing *no* features, which would require marginalizing twice;
but let's not!)

From Shapley to SHAP

Axioms: In the context of ML, the Shapley value satisfies the following properties/axioms:

Efficiency: (previously: the value of a game is divided among its players) For ML, the prediction is divided among its features. Specifically, the sum of all Shapley values and the average predicted value is equal to the observation's prediction:

$$f(x) = \sum_{i=1}^p \phi_i + E_X[f(X)]$$

From Shapley to SHAP

Axioms: In the context of ML, the Shapley value satisfies the following properties/axioms:

Symmetry: Two features will have the same Shapley value if they make the same contributions to all coalitions.

Dummy: A feature will have a Shapley value of 0 if it never changes the prediction. In other words, features that are not used in a model will not have a Shapley value.

Additivity: (for ensemble models) The overall Shapley value can be calculated by taking the weighted average of Shapley values for each model in the ensemble.

Consistency: If, in a new model, a feature increases a prediction more than before, then its Shapley value will increase. Thus we can compare feature contributions across different models.

Approximation of Shapley values

Calculating exact Shapley values is computationally expensive! Complexity increases combinatorially as we add more features. So we'd like some way to approximate them.

One way we could approximate them is via **sampling**:

- Sample the feature values at random (recall that we know their distribution)
- Compare the prediction with the given feature value (+i) to predictions with other (random) feature values:

$$\hat{\phi}_i = \frac{1}{M} \sum_{m=1}^M (f(x_{+i}^m) - f(x_{-i}^m))$$

Approximation of Shapley values

This type of approximation scheme is still impractical

The original SHAP paper (“A Unified Approach to Interpreting Model Predictions”) describes various other approximation schemes (arguably, this is the paper’s main contribution), though these are significantly more involved

The popularity of the above techniques owes largely to the success of these approximations, and their efficient implementation via the SHAP Python package.

Code example (SHAP Python package): <https://github.com/shap/shap>

SHAP

Note: the above section covers most of what's in the original SHAP paper (“A Unified Approach to Interpreting Model Predictions”, i.e., the paper which applies these ideas to ML, rather than the original paper describing Shapley values). But the treatment there is not so easy, so the presentation is based on the article linked below.

Adversarial attacks on LIME and SHAP

LIME and SHAP depend on perturbation of the input; it is possible to create adversarial models that *appear* ethical when explained using perturbed inputs but which are not ethical when evaluated without perturbation

As such, these models are only functionally-grounded as long as the model is trained without malicious intent

If curious see: <https://arxiv.org/pdf/2108.04840>, <https://arxiv.org/pdf/1911.02508>

Study points & take-homes

- Introduced LIME and SHAP
- Briefly discussed some of their limitations

Interpretable and explainable ML

Case-study: Concept bottleneck models

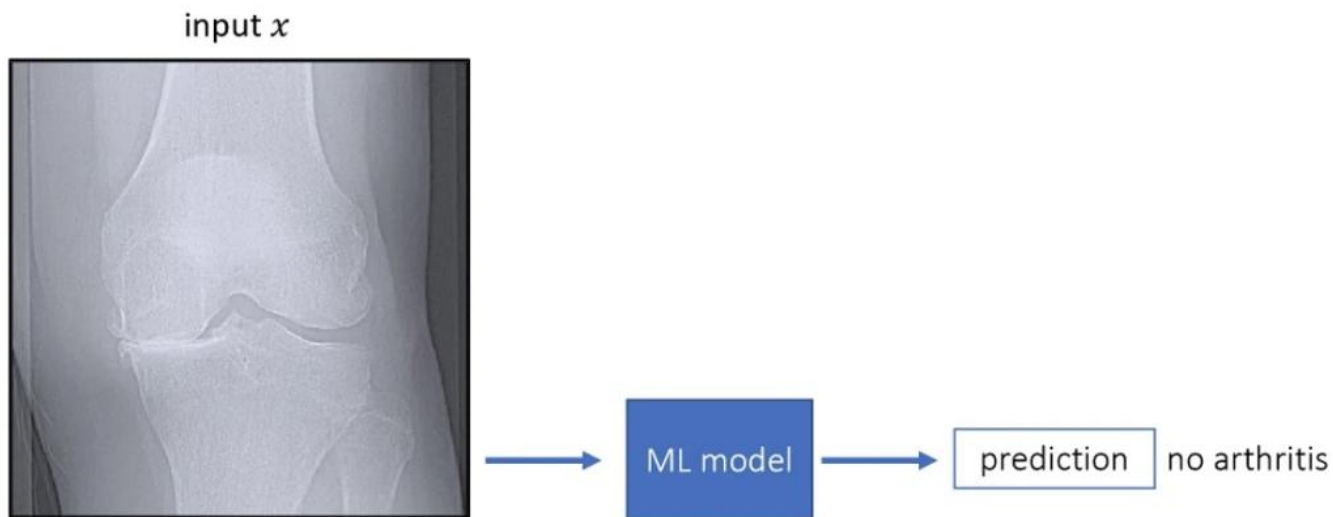
Concept bottleneck models

Rather than training a model to directly predict some outcome (e.g. "does this x-ray indicate arthritis?"), force the model to predict via intermediate human-provided "concepts" (e.g. "has bone spurs")

The goal is not only to make the model more inherently interpretable, but also supports manipulation: e.g. we can query the model about what it *would have predicted* if bone spurs had been identified

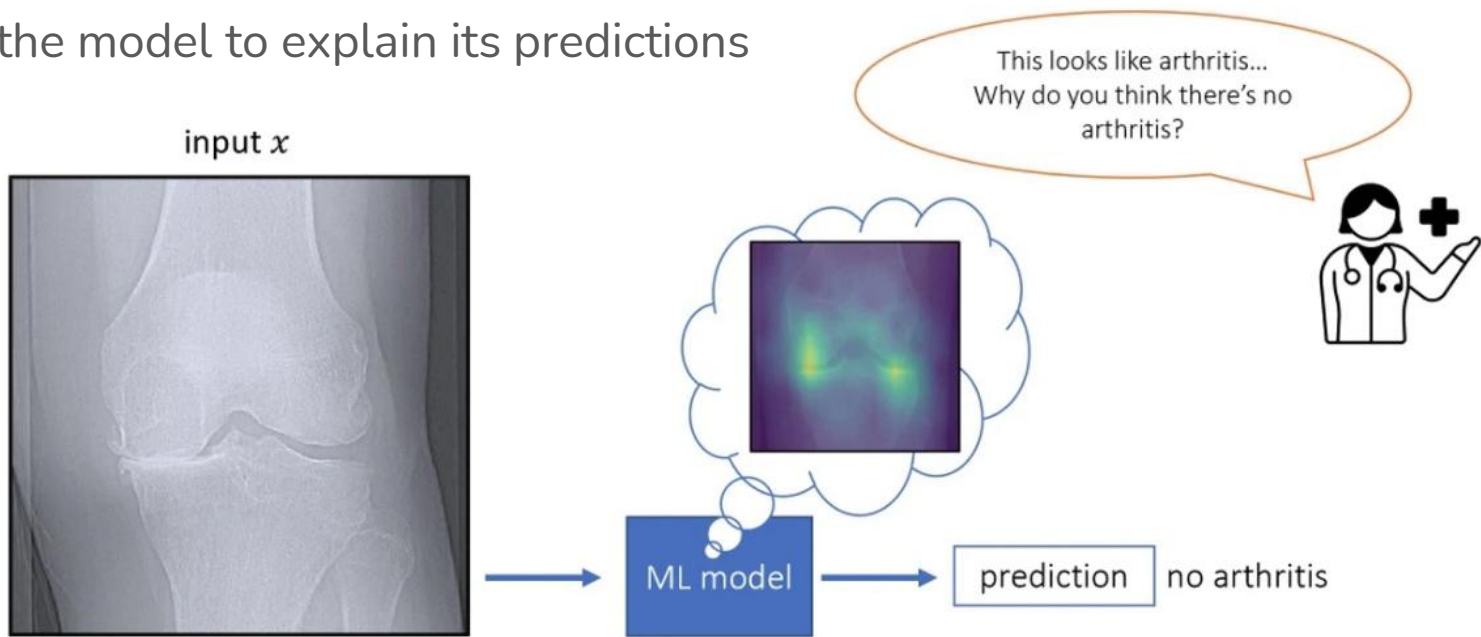
Concept bottleneck models

Black-box model (image \rightarrow prediction): not interpretable



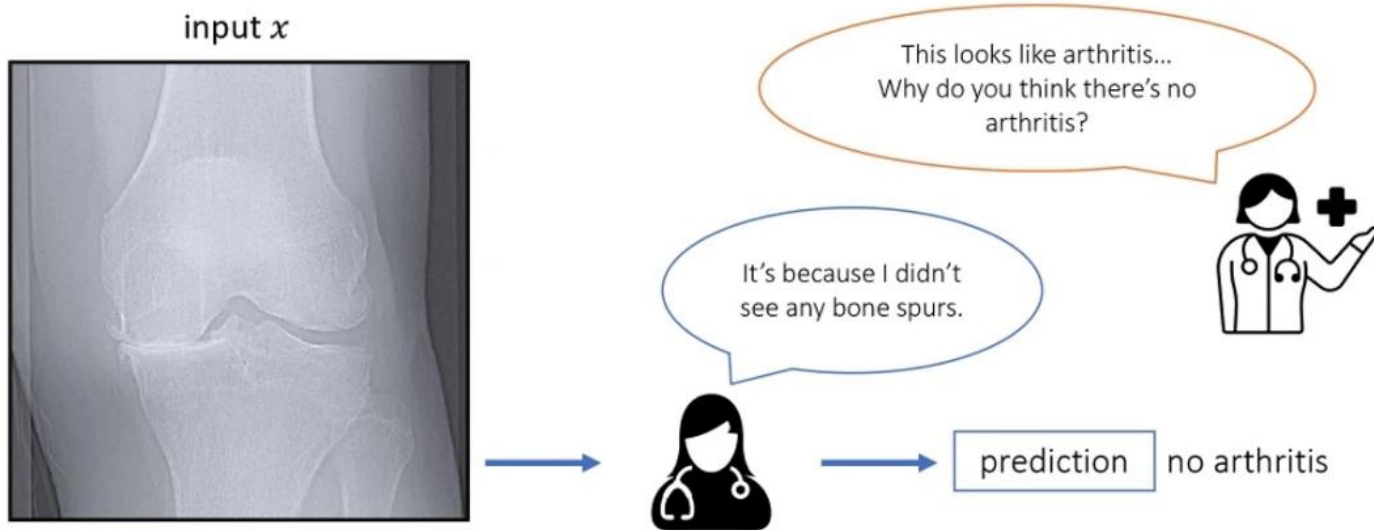
Concept bottleneck models

Want the model to explain its predictions



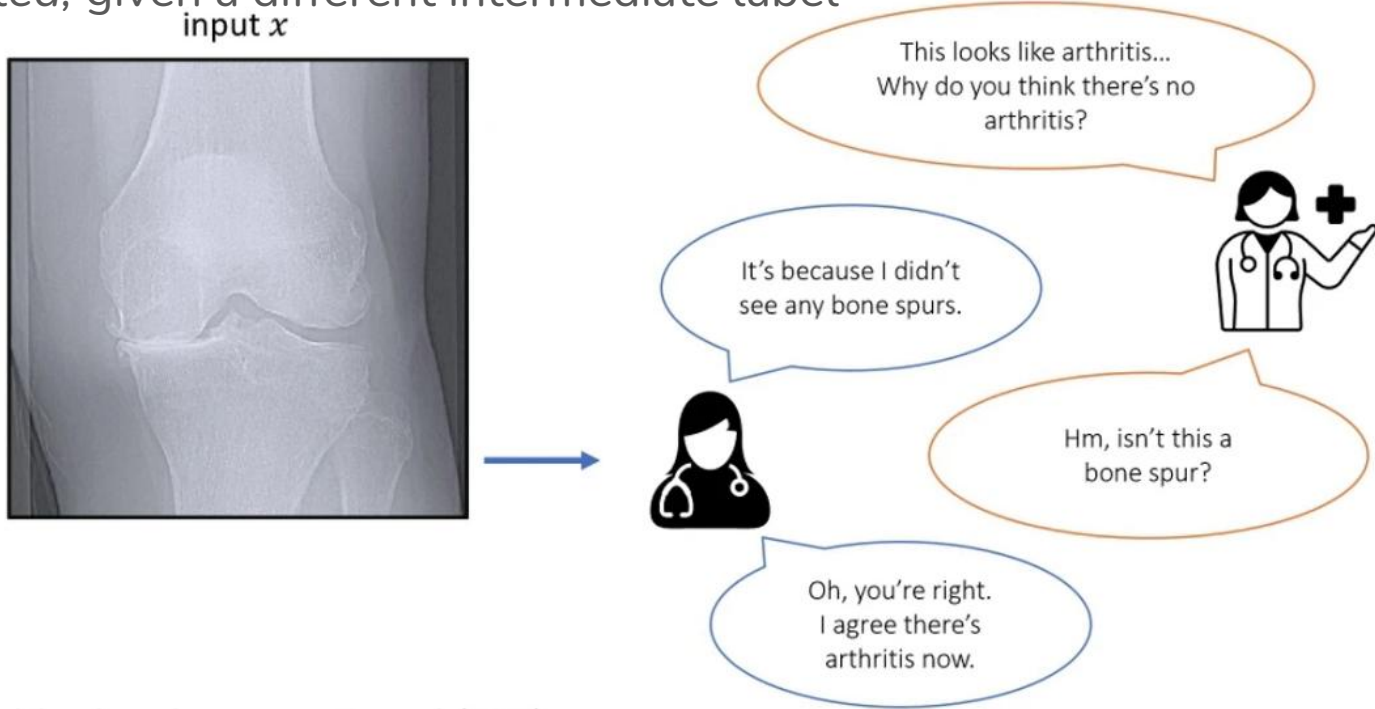
Concept bottleneck models

Idea: explain predictions in terms of human-provided concepts



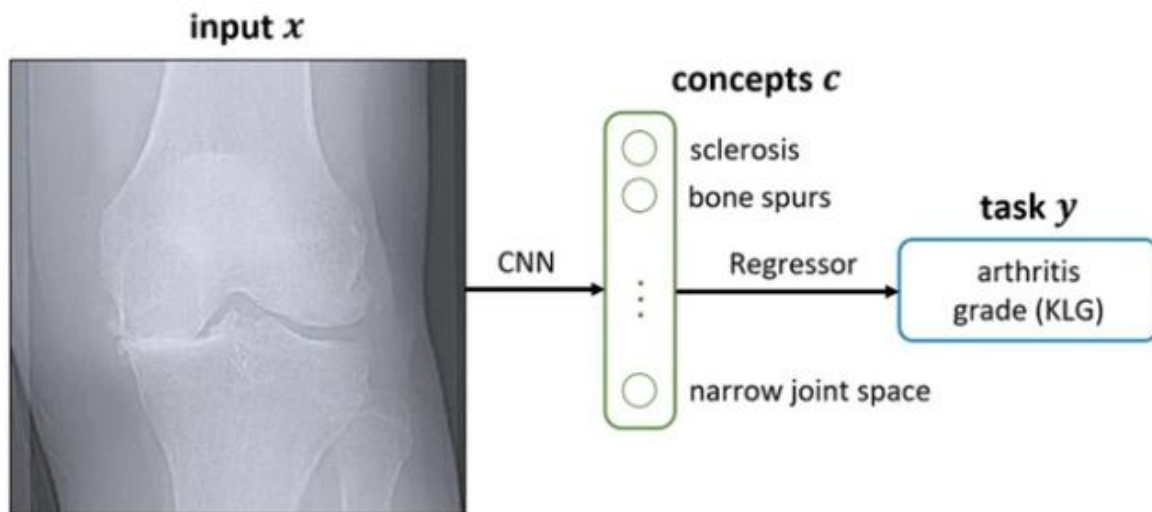
Concept bottleneck models

Want to be able to “manipulate” the model, and ask what it *would have predicted*, given a different intermediate label

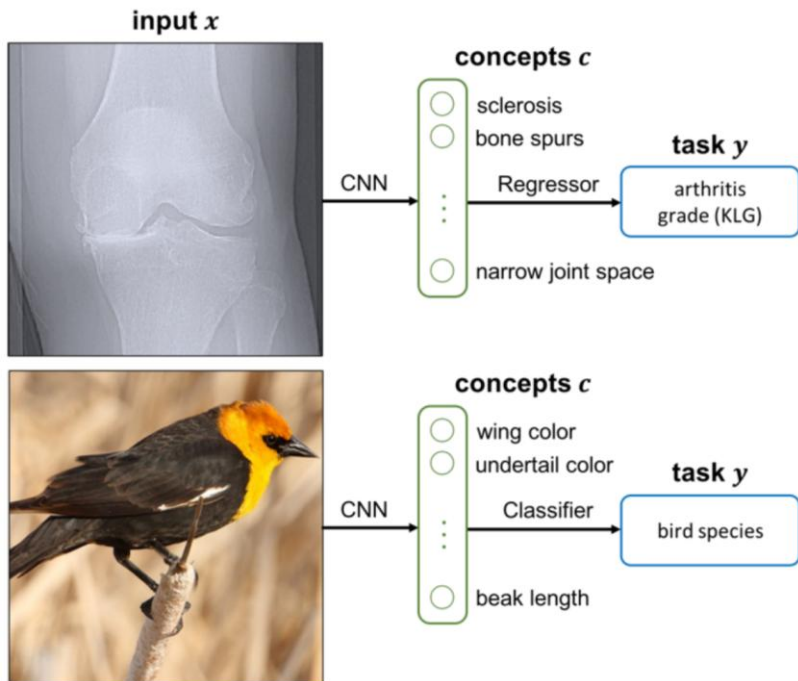


Concept bottleneck models

Idea: force predictions through a “bottleneck” of intermediate concepts



Concept bottleneck models



Ways to train:

Independent:

$$x \rightarrow c \quad \text{and} \quad c \rightarrow y$$

Sequential:

$$x \rightarrow c \quad \text{then} \quad \hat{c} \rightarrow y$$

Joint:

$$x \rightarrow c \rightarrow y$$

Standard (no concepts):

$$x \rightarrow y$$

Concept bottleneck models

Some findings:

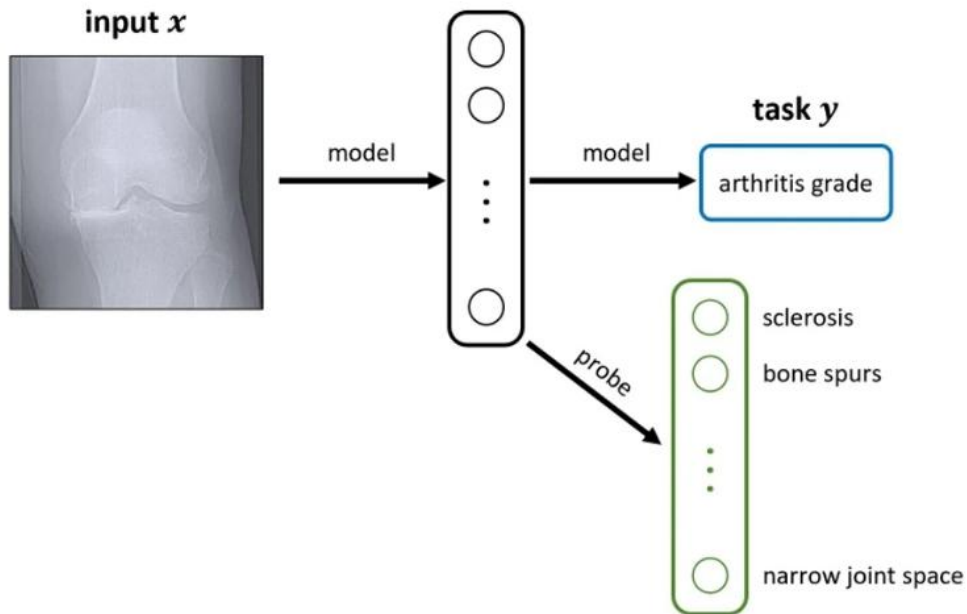
- Fairly competitive with standard models (though this is dependent on a dataset with sufficiently rich concepts)
- Not much trade-off between label accuracy and concept error (i.e., tuning the model such that accuracy is high doesn't make concept error low and vice versa) (is this surprising?)

Concept bottleneck models

Some findings:

- Training an end-to-end model, and trying to recover concepts from the trained model leads to low concept accuracy

food for thought: would this be true if we had much larger unlabeled (i.e., no concepts) datasets?



Concept bottleneck models

Limitations:

- Need to have datasets with useful concepts; on the one hand this may be required to meaningfully interpret models anyway; on the other hand, accuracy may be limited compared to those possible with bigger unlabeled (no concepts) datasets

Interpretable and explainable ML

4.7: How should explanations be evaluated?

This section

- A few words about how explanations should be evaluated in practice
- Somewhat outside the scope of this class (e.g. human subject experiments); mostly just a reference to Doshi-Valez, 2017:
<https://arxiv.org/pdf/1702.08608>

Application-grounded evaluation: real humans, real tasks

If researchers have a concrete application in mind, such as diagnosing patients with a disease, the best way to show that the model works is to evaluate it with respect to the task: *doctors performing diagnoses*

This reasoning aligns with evaluation in human-computer interaction and visualization; e.g. a homework-hint system is evaluated on whether the student achieves better post-test performance (Williams et al., 2016)

Examples of experiments include:

- Domain expert experiment with the exact application task
- Domain expert experiment with a simpler or partial task to shorten experiment time and increase the pool of potentially-willing subjects

Human-grounded metrics: real humans, simplified tasks

Human-grounded evaluation is about conducting simpler human-subject experiments that maintain the essence of the target application; this is appealing when experiments are challenging/expensive, since evaluations can be completed with laypeople

The key question is how to evaluate the quality of an explanation without a specific end-goal: ideally, our evaluation approach will depend only on the quality of the explanation, regardless of (e.g.) the correctness of the associated prediction

Human-grounded metrics: real humans, simplified tasks

Example experiments include:

- **Binary forced choice:** humans are presented with pairs of explanations and must choose the one that they find of higher quality
- **Forward simulation/prediction:** humans are presented with an explanation and an input, and must correctly simulate the model's output
- **Counterfactual simulation:** humans are presented with an explanation, an input, and an output, and are asked what must be changed to change the method's prediction to a desired output

E.g. intrusion-detection tests (Chang et al., 2009) in topic models ask humans to find the difference between the model's true output and some corrupted output as a way to determine whether the human has correctly understood what the model's true output is

Functionally-grounded evaluation: no humans, proxy tasks

Functionally-grounded evaluation requires no human experiments; instead, it uses some formal definition of interpretability as a proxy for explanation quality

Functionally-grounded evaluations are most appropriate if we have a class of models that have already been validated, e.g. via human-grounded experiments; they may also be appropriate when a method is not yet mature or when human subject experiments are unethical

The challenge is to determine what proxies to use; for example, decision trees have been considered interpretable in many situations (Freitas, 2014)

Functionally-grounded evaluation: no humans, proxy tasks

Examples of experiments include

- Show the improvement of prediction performance of a model that is already proven to be interpretable (assumes that someone has run human experiments to show that the model class is interpretable).
- Show that one's method performs better with respect to certain regularizers—for example, is more sparse—compared to other baselines (assumes someone has run human experiments to show that the regularizer is appropriate).

Interpretable and explainable ML

4.8: Explainability of image classifiers

This section

- A few words about explainability of image classifiers in terms of individual pixels
- Also somewhat outside the scope of this class; mostly just pointing to related references

Saliency maps – “vanilla” gradients

Which pixels in an image contribute to our prediction that it belongs to a certain class?

- SHAP and LIME generate such explanations by manipulating *parts of the image* (e.g. occlusion)
- They are *model-agnostic*: explanations are based on manipulating the *image*, rather than analysis of the model

Saliency maps

Here we'll look at **gradient-based methods**:

- Roughly, gradient-based methods tell us *how relevant is a particular pixel to the classification of an image*
- Specifically, how much will a *change* in a pixel cause a *change* in a prediction?

Saliency maps

Motivating example:

Consider a linear classifier which given a class c assigns a score $S_c(I)$ to an image I (or image features):

$$S_c(I) = w_c^T I + b_c$$

(i.e., a regular old linear model). Here, the weights w_c would directly correspond to the importance of the corresponding features.

Saliency maps

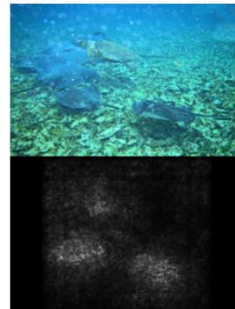
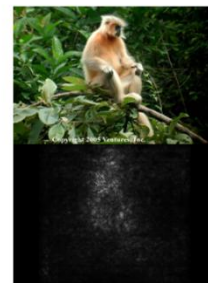
But in practice, our score function $S_c(I)$ will be highly non-linear. Instead, let's compute the first-order Taylor expansion with respect to a particular point (i.e., image) I_0 , i.e.,

$$w = \left. \frac{\partial S_c}{\partial I} \right|_{I_0}$$

Saliency maps

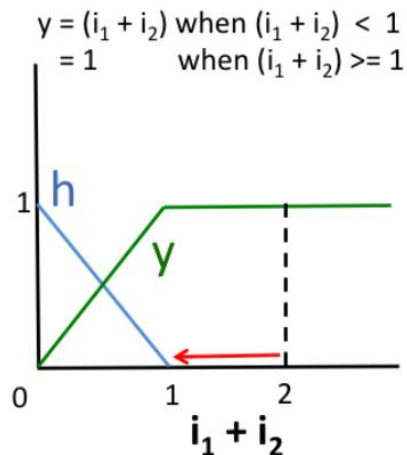
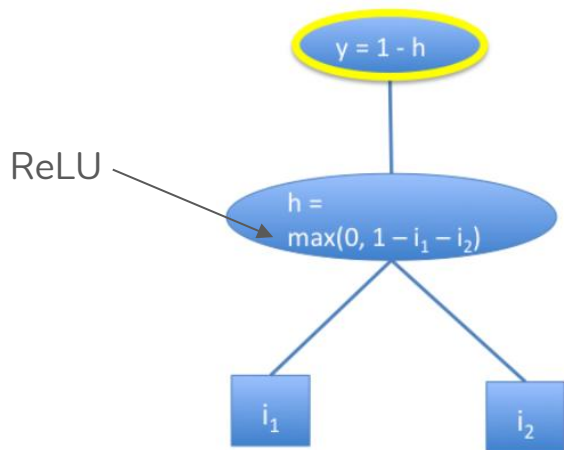
That's about it! Other minor details, e.g. should we visualize the absolute value or should we visualize positive and negative values differently?

images and their saliency maps for top-1 predicted class:



Saliency maps

Problem: certain commonly-used activation functions in neural networks will not respond to perturbations (i.e., they'll have zero gradient)



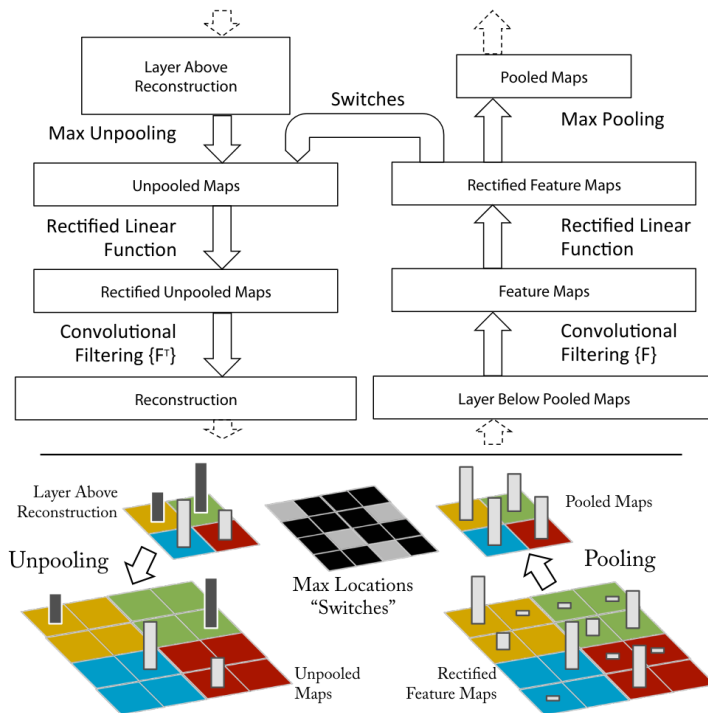
Once $i_1 = 1$ and $i_2 = 1$, perturbing the other value to zero won't change the output (so we'd erroneously conclude that those values aren't related to the label)

Saliency maps – DeconvNet

Very roughly speaking, generalizes the “vanilla” gradient-based method by using a different choice to backpropagate the gradient through ReLU.

(if anyone can adequately distill the idea of this paper into one slide, let me know!)

See also e.g. GradCAM



Interpretable and explainable ML

4.9: Explainability of language models

This section

- Discuss how language models might be evaluated (mostly not *generative* language models though)
- Mostly an extended case-study of a specific survey paper

Explainability of language models

We'll mostly cover the paper “Post-hoc Interpretability for Neural NLP” (<https://arxiv.org/pdf/2108.04840>), which covers only post-hoc interpretability, and *mostly* covers local methods

Very brief introduction to NLP concepts

- Bag-of-words models
- Cosine distance

I'll spend almost no time on these so please revise if needed!

(but they're only needed for a few examples so please keep paying attention even if you miss those parts!)

Bag-of-words models

Q: How should I represent the following piece of text using a **fixed-length feature vector**?

Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.

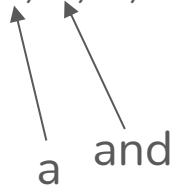
Bag-of-words models

Q: How should I represent the following piece of text using a **fixed-length feature vector**?

A: Vector counting which dictionary words occur:

[3, 6, 0, 0,]

a and



Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.

Bag-of-words models

Note: randomly shuffling words preserves the bag-of-words representation (roughly, this is why it's called a "bag of words"):

Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.

yeast and minimal red body thick light a Flavor sugar strong quad. grape over is molasses lace the low and caramel fruit Minimal start and toffee. dark plum, dark brown Actually, alcohol Dark oak, nice vanilla, has brown of a with presence. light carbonation. bready from retention. with finish. with and this and plum and head, fruit, low a Excellent raisin aroma Medium tan

Bag-of-words models

Lots of details to get right to actually implement this:

- How to keep the dictionary to a manageable size
- How to handle capitalization, punctuation, stopwords, stemming, etc.
- How to capture any notion of syntax (e.g. bags-of-ngrams), and when is this necessary

For more detail refer to my other class notes (158/258), or textbook (or pretty much any NLP class)

Motivating example

Predict the sentiment (positive or negative) of the sentence \mathbf{x} :

		$p(y \mathbf{x}; \theta)$	y
\mathbf{x}	<u>the</u> <u>year</u> <u>'s</u> <u>best</u> <u>and</u> <u>most</u> <u>unpredictable</u> <u>comedy</u>	0.91	pos
\mathbf{x}	<u>we</u> <u>never</u> <u>feel</u> <u>anything</u> <u>for</u> <u>these</u> <u>characters</u>	0.95	neg
\mathbf{x}	<u>handsome</u> <u>but</u> <u>unfulfilling</u> <u>suspense</u> <u>drama</u>	0.18	neg

input → prediction → label

Local explanations

Predict the sentiment (positive or negative) of the sentence \mathbf{x} :

- Which tokens (words) are most important for the prediction [input features]?
- What would break the model's predictions [adversarial examples]?
- What training examples influenced the prediction [influential examples]?
- What does the model consider a valid opposite example [counterfactuals]?
- What would a generated natural language explanation be?

Local explanations – input features

Before getting sophisticated: for this task we could straightforwardly visualize the features of a linear model

$$\text{e.g. rating} = \theta_0 + \sum_{w \in \text{text}} \theta_w \text{count}(w)$$

Or use gradients as for images

Local explanations – input features

Which tokens (words) are most important for the prediction (input features)?

- Similar to gradient-based explanations of images (saliency maps)
- One complication: “pixels” in an image don’t easily map to “words” in a document

the year 's best and most

0	0	0	1	0	0
0	0	0	0	1	0
0	1	0	0	0	0
0	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	0	0
0	0	1	0	0	0
0	0	0	0	0	0

Local explanations

Which tokens (words) are most important for the prediction [input features]?

- Think of the bag-of-words representation like a 2-d image
- Take the norm across the vocabulary dimension

		$p(y \mathbf{x}; \theta)$	y	c
x	<u>the</u> <u>year</u> <u>'s</u> <u>best</u> <u>and</u> <u>most</u> <u>unpredictable</u> <u>comedy</u>	0.91	pos	pos
x	<u>we</u> <u>never</u> <u>feel</u> <u>anything</u> <u>for</u> <u>these</u> <u>characters</u>	0.95	neg	neg
x	<u>handsome</u> <u>but</u> <u>unfulfilling</u> <u>suspense</u> <u>drama</u>	0.18	neg	pos

Local explanations

This will work, though has some of the same problems as with image saliency maps:

- Important areas may have zero gradients; as with image models different techniques can be used to resolve issues with ReLU activation etc.
- Since we took the norm across the vocabulary dimension, gradients no longer have a direction (as compared to image models)

Local explanations – LIME

We can also use LIME (or SHAP) to explain predictions of language models. As before we need to:

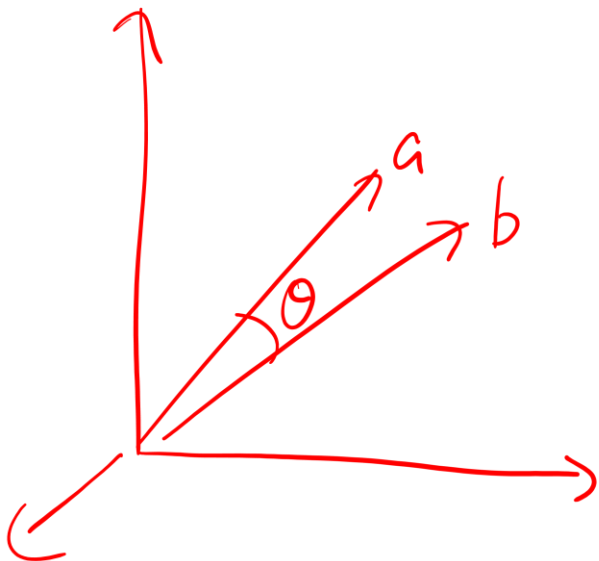
- Sample nearby observations to x
- Train a linear model on those observations
- Visualize

Q: How should we sample “nearby” observations for language data?

Local explanations – LIME

Q: How should we sample “nearby” observations for language data?

A: Cosine similarity



$$\theta = \cos^{-1} \left(\frac{a \cdot b}{\|a\| \cdot \|b\|} \right)$$
$$\cos(\theta) = \frac{a \cdot b}{\|a\| \cdot \|b\|}$$

Local explanations – LIME

		$p(y \mathbf{x}; \theta)$	y	c
x	<u>the</u> <u>year</u> 's <u>best</u> <u>and</u> <u>most</u> <u>unpredictable</u> <u>comedy</u>	0.91	pos	pos
x	<u>we</u> <u>never</u> <u>feel</u> <u>anything</u> <u>for</u> <u>these</u> <u>characters</u>	0.95	neg	neg
x	<u>handsome</u> <u>but</u> <u>unfulfilling</u> <u>suspense</u> <u>drama</u>	0.18	neg	pos

Local explanations – LIME

Notes:

- Cosine similarity of Bag-of-Words vectors (or variants) aren't great measures of semantic similarity (though can use alternatives)
- Can also *mask* tokens in \mathbf{x} , for models which support masking
- Same ideas can mostly be applied to SHAP

Local explanations – similar examples

Briefly: another simple local explanation method consists of retrieving similar examples from the training set; one can retrieve similar examples with both the same and different labels:

	\mathbf{x}	$p(y \mathbf{x}; \theta)$	y	Δ
\mathbf{x}	<u>the</u> <u>year</u> <u>'s</u> <u>best</u> <u>and</u> <u>most</u> <u>unpredictable</u> <u>comedy</u>	0.91	pos	0.21
$\tilde{\mathbf{x}}$	<u>a</u> <u>delightfully</u> <u>unpredictable</u> <u>,</u> <u>hilarious</u> <u>comedy</u>	0.95	pos	3.82
$\tilde{\mathbf{x}}$	<u>loud</u> <u>and</u> <u>thoroughly</u> <u>obnoxious</u> <u>comedy</u>	0.98	neg	-1.51

(see paper for several implementations with specific influence functions)

↑
“influence”

Adversarial examples

An *adversarial example* is an input that causes the model to produce a wrong prediction due to limitations of the model

These are often generated by perturbing an existing example for which the model produces a correct prediction

These can be relevant as explanations in the sense that they reveal the underlying “logic” and limitations of the model

Adversarial examples – HotFlip

Basic idea: replace words in a sentence with “similar” words (high cosine similarity) that significantly change the prediction

In practice this is implemented by computing gradients:

$$\mathcal{L}(y, \tilde{\mathbf{x}}_{t:v \rightarrow \tilde{v}}) - \mathcal{L}(y, \mathbf{x}; \theta) \approx \frac{\partial \mathcal{L}(y, \mathbf{x}; \theta)}{\partial x_{t, \tilde{v}}} - \frac{\partial \mathcal{L}(y, \mathbf{x}; \theta)}{\partial x_{t, v}}$$

perturbed input with single token
($v \rightarrow \hat{v}$) change

derivative of loss w.r.t. single
token

Adversarial examples – HotFlip

This would trivially end up just replacing positive words by very negative ones (for positively labeled sentences) and *vice versa*

To make sure the model selects *paraphrases* (i.e., words that will *change the prediction* but leave the *meaning* unchanged), the model constrains the cosine similarity between original and replaced tokens to be > 0.8

Adversarial examples – HotFlip

		$p(y \mathbf{x}; \theta)$	y
\mathbf{x}	<u>the</u> <u>year</u> 's <u>best</u> <u>and</u> <u>most</u> <u>unpredictable</u> <u>comedy</u>	0.91	pos
	<u>the</u> <u>year</u> 's <u>finest</u> <u>and</u> <u>most</u> <u>unpredictable</u> <u>comedy</u>	0.30	-
$\tilde{\mathbf{x}}$	<u>the</u> <u>year</u> 's <u>finest</u> <u>and</u> <u>most</u> <u>unforeseeable</u> <u>comedy</u>	0.08	-
\mathbf{x}	<u>we</u> <u>never</u> <u>feel</u> <u>anything</u> <u>for</u> <u>these</u> <u>characters</u>	0.95	neg
$\tilde{\mathbf{x}}$	<u>we</u> <u>never</u> <u>feel</u> <u>anything</u> <u>for</u> <u>these</u> <u>people</u>	0.03	-

adversarial
example

That's about enough! *Lots more in paper*

- **Anchors:** find short lists of words most relevant for making predictions (fairly similar to LIME etc., but not gradient-based)
- **Semantically equivalent adversaries:** similar to HotFlip, but uses a “paraphrasing model” to produce paraphrases rather than cosine similarity
- **Counterfactual explanations:**

References for Module 4

- Interpretable Machine Learning: <https://christophm.github.io/interpretable-ml-book/>
- A survey on XAI and natural language explanations: <https://sentic.net/explainable-artificial-intelligence.pdf>
- The mythos of model interpretability: <https://arxiv.org/abs/1606.03490>
- Introduction to Machine Learning: <https://courses.cs.washington.edu/courses/cse416/>
- From Shapley to SHAP: <https://towardsdatascience.com/from-shapley-to-shap-understanding-the-math-e7155414213b>
- Concept Bottleneck Models: <https://slideslive.com/38928546/concept-bottleneck-models>
- Post-hoc Interpretability for Neural NLP: <https://arxiv.org/pdf/2108.04840>