Jhelum Chakravorty

Published Aug 6
Fork of Function Fitting Crash Course by Kris Sankaran

Function Fitting Crash Course

KNN, Linear Regression, and their Relatives

IFT 6758, Fall 2020; Reading: ISLR sections 3.2.1, 3.3.1, 3.3.2, 3.5, 4.3, 7.2, 8.1

```
html`<button onclick="

var changeHandler = function(){

var fs = ((document.fullScreenElement && document.fullScreenElement !== null) ||
(!document.mozFullScreen && !document.webkitIsFullScreen));
      if (fs) {
          [...document.cells].forEach(e=>{e.style.height= ''});
      } else {
          [...document.cells].forEach(e=>{e.style.height=
window.screen.height+'px'});
          var elmnt = document.cells[2];
          elmnt.scrollIntoView();
      }
}
  document.addEventListener('fullscreenchange', changeHandler,false);
  document.addEventListener('webkitfullscreenchange', changeHandler, false);
  document.addEventListener('mozfullscreenchange', changeHandler, false);


    if ((document.fullScreenElement && document.fullScreenElement !== null) ||
        (!document.mozFullScreen && !document.webkitIsFullScreen)) {
        if (document.documentElement.requestFullScreen) {
            document.documentElement.requestFullScreen();
        } else if (document.documentElement.mozRequestFullScreen) {
            document.documentElement.mozRequestFullScreen();
        } else if (document.documentElement.webkitRequestFullScreen) {
```

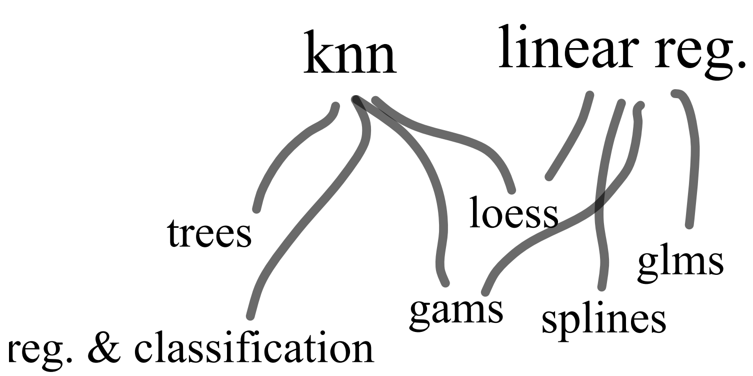
```
 document.documentElement.webkitRequestFullScreen(Element.ALLOW_KEYBOARD_INPUT);
        }
}

    ">Click to go fullscreen</button>`
```

Goals of function fitting

- Deeply understand mechanics of k-nearest neighbors (KNN) and linear models
- See how bias-variance tradeoff and curse of dimensionality manifest themselves in these models
- Get a feeling for how versatile the ideas in KNN and linear models can be

Why KNN and Linear Models?

- These models are the foundation for nonparametric and parametric learning algorithms, respectively
- Understanding strengths and limitations here will help with in other contexts



Locally Weighted Linear Regression (loess or lowess)

Parametric vs. Nonparametric Models

- Parametric: Have a fixed complexity, that remains constant even as the number of samples grows
  - (fixed # of parameters)
- Nonparametric: Has a complexity that grows with the number of samples that arrive
  - (growing # of effective parameters)

KNN

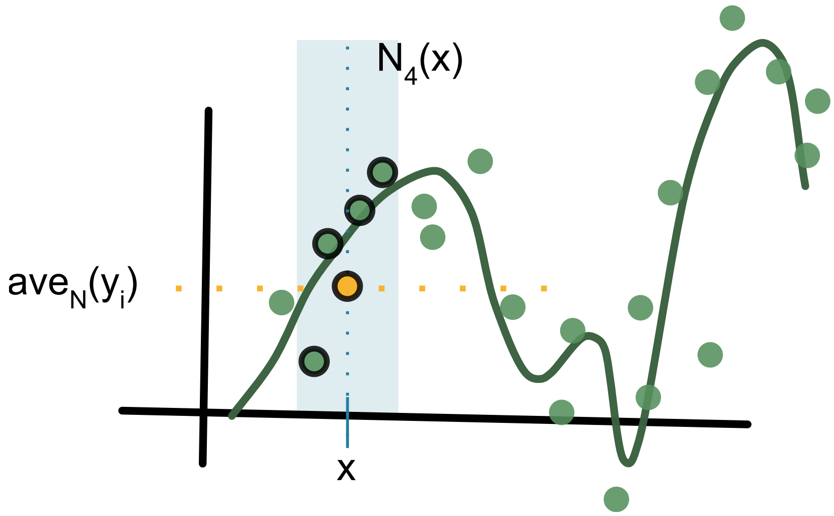
Algorithm (KNN Regression)

Say that we have n samples $(x_i, y_i)_{i=1}^{n}$. Estimate the data generating function f by

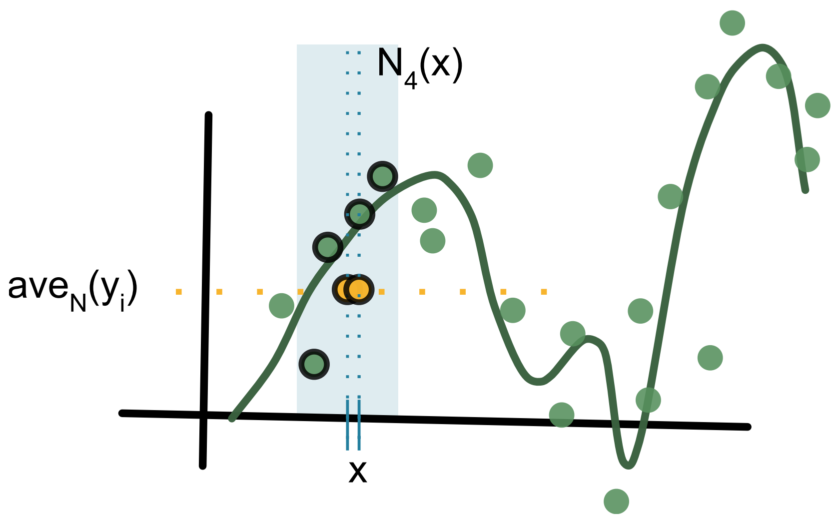
$$\hat{f}(x) = \frac{1}{K} \sum_{i \in N_K(x)} y_i$$

where N_K is the set of K nearest-neighbors of x within the training set.

KNN Regression Picture

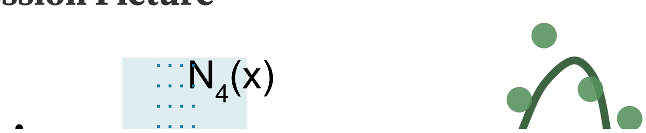


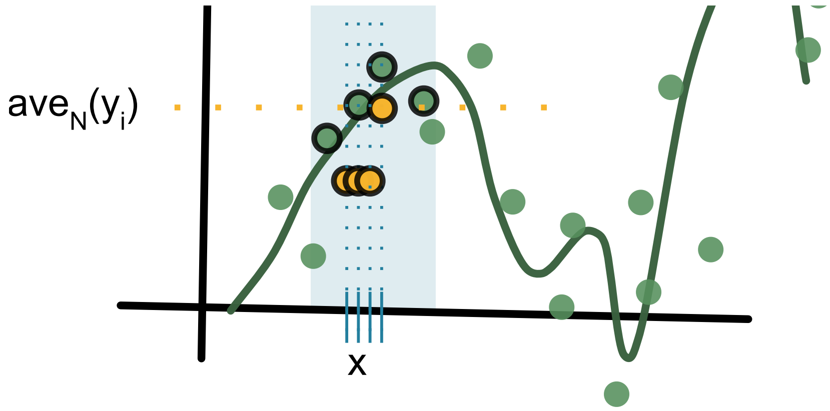
KNN Regression Picture



If you move x a little, the nearest points don't change, so the prediction is the same.

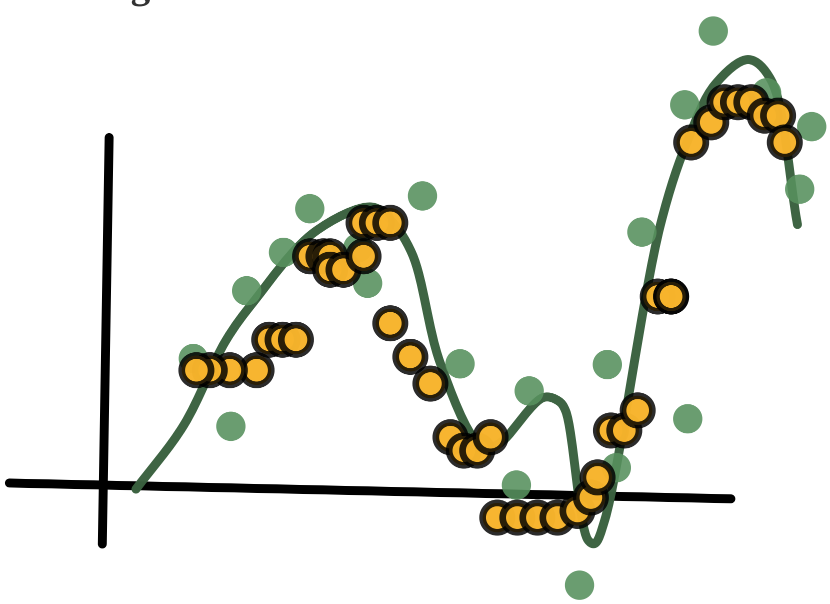
KNN Regression Picture





Once a new point enters the neighborhood, the average over the neighborhood changes.

KNN Regression Picture



You can do this over a fine grid to make an estimate of f.

Bias-variance tradeoff

The model complexity is controlled by the size of the neighborhood.

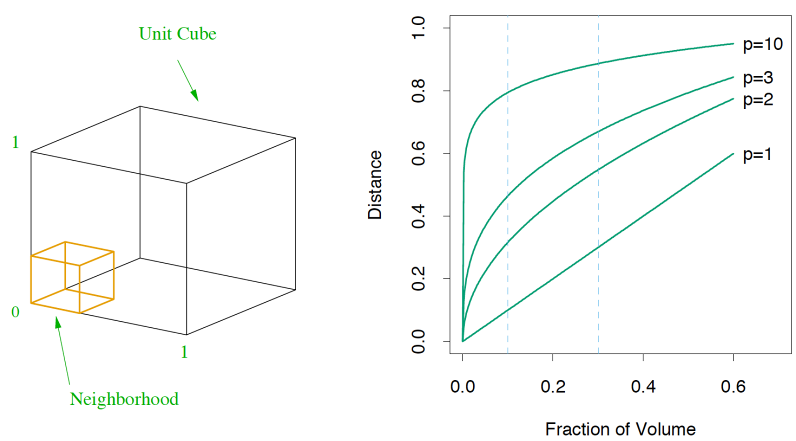
- Large K --> Lower variance, larger bias
- Small K --> Higher variance, smaller bias

Small K — Higher variance, smaller bias

Larger K learns smoother functions that might not match the true function's wiggliness. Smaller K will be larger variance, but can match complex functions when the sampling density is high enough.

Curse of Dimensionality

- In higher dimensions, n the density of samples gets lower and lower
- The lack of close-by neighbors increases both the bias and the variance
- You end up needing to look at almost the whole space to make a prediction. Means you average over points that are quite different in terms of x.



Classification

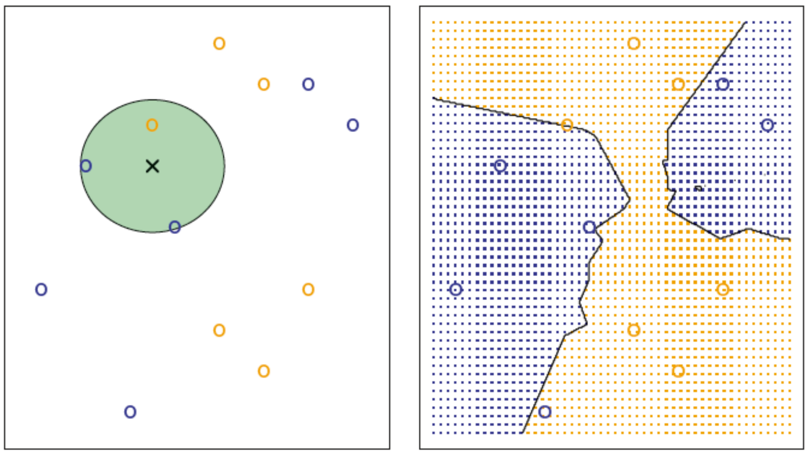
- The method extends to classification. The probability that a location x gets assigned to class j is approximated by

$$\hat{p}_k\left(x\right) = \frac{1}{K} \sum_{i \in N_K(x)} \mathbb{I}\left(y_i = j\right)$$

Looking at the class with top probability, this is like taking local majority votes.

Classification

- The method extends to classification. The probability that a location x gets assigned to class j is approximated by



Looking at the class with top probability, this is like taking local majority votes.

Discussion

- What are the pros and cons of KNN in practice?

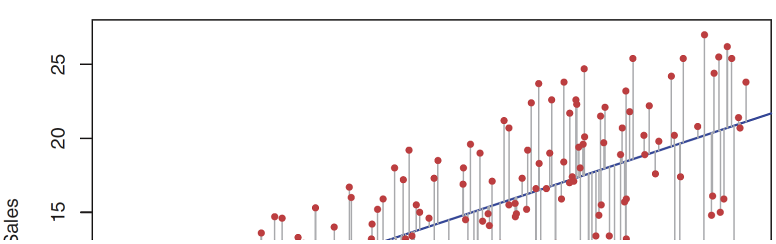
Possible Answers

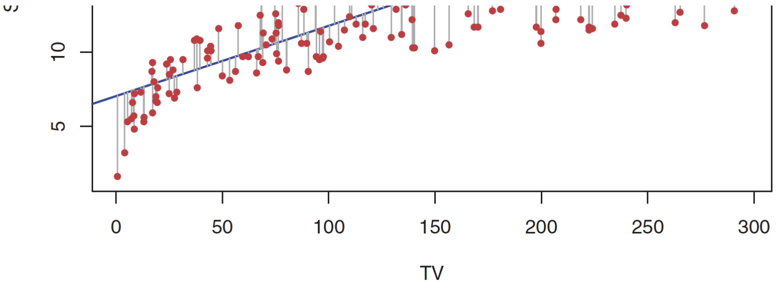
- Pros
  - Possible to fit arbitrarily complex functions
  - Can achieve range of model complexities
  - Easy to inspect predictions
- Cons
  - Looking up nearest neighbor can be computationally expensive
  - Deteriorates when there are many variables

Linear Regression

Definition (one-dimension)

- Function family --> $f_\beta\left(x\right) = \beta x$ for some β
  - One unit increase in x increases mean prediction by β
- Choose β so that $\sum_i \left(y_i - f_\beta\left(x_i\right)\right)^2$ is small
- This minimizes the vertical distances between y and f





Derivation (one-dimension)

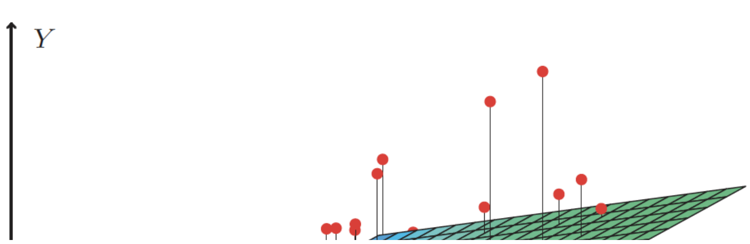
- Can do this in closed form,

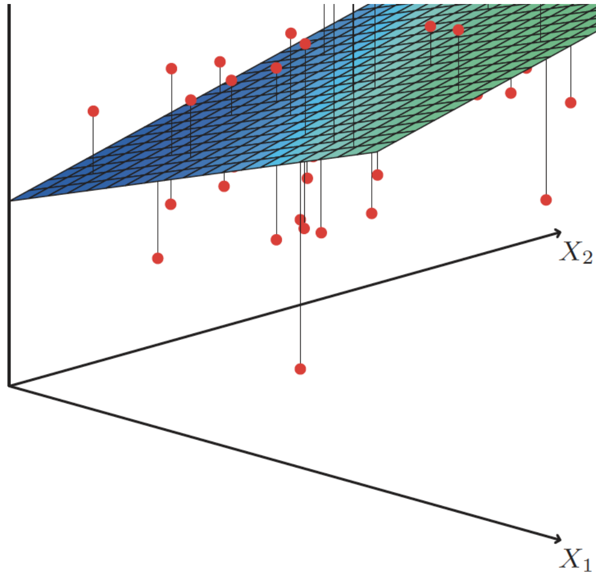
$$\frac{\partial}{\partial \beta} \sum_i (y_i - \beta x_i)^2 = 0$$
$$\iff \sum_i x_i (y_i - \beta x_i) = 0$$
$$\iff \beta = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

Interpretation: Each unit increase in x leads to an increase by one unit variance in y, shrunken by a factor determined by the correlation (compare with ISLR eqn. 3.18).

Definition (Higher Dimensions)

- Function family --> $f_\beta(x) = x^T \beta$ for some $\beta \in \mathbb{R}^p$
  - One unit increase in x_j increases mean prediction by β_j
- Choose β to minimize $\sum_i \left(y_i - x_i^T \beta\right)^2$





Derivation (Higher Dimensions)

This can again be done analytically (but don't worry if you haven't seen the calculus / linear algebra),

$$\begin{aligned} &\frac{\partial}{\partial \beta} \sum_i \left(y_i - x_i^T \beta\right)^2 = 0 \\ \iff &\sum_i x_i \left(y_i - x_i^T \beta\right) = 0 \\ \iff &\beta = \left(\sum_i x_i x_i^T\right)^{-1} \sum_i x_i y_i \\ &= \left(X^T X\right)^{-1} X^T y \end{aligned}$$

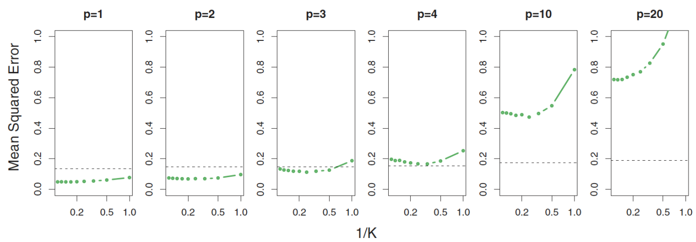
This is a generalization of the previous story.

Curse of dimensionality

- It's possible to show $\mathbb{E}\left[\left(y_i - f_{\hat{\beta}}\left(x_i\right)\right)^2 |x_i\right] \approx \frac{p\sigma^2}{n}$
  - assuming the true function is linear
- In high dimensions, functions **could** get exponentially rough

rough

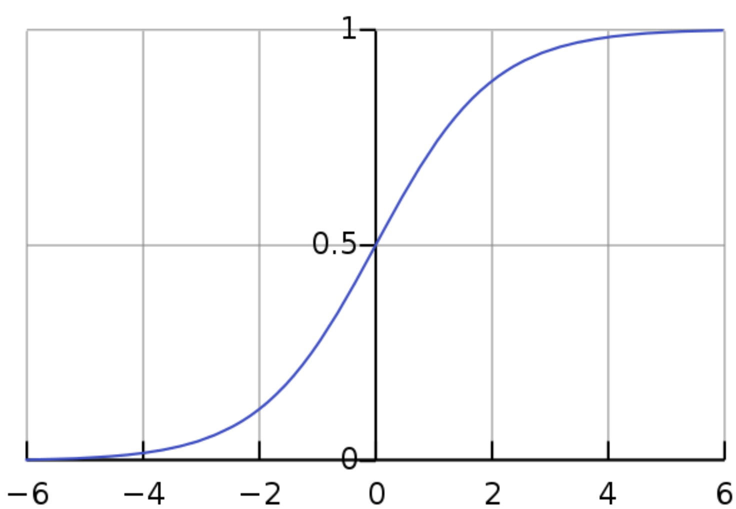
- KNN tries to do well in all cases, and suffers as a result
- By betting against exponential roughness, regression does better



Linear regression deteriorates linearly with dimension.

Logistic Regression

- For binary classification, say the probability of class 1 is $p_\beta(x) = \sigma\left(x^T\beta\right)$, where $\sigma(z) = \frac{\exp(z)}{1+\exp(z)}$
- The transformation keeps the probability between 0 and 1



Logistic Regression

- We can make this resemble linear regression,

$$\begin{aligned}\log\left(\frac{p_\beta(x)}{1-p_\beta(x)}\right) &= \sigma^{-1}(p_\beta(x)) \\ &= x^T\beta\end{aligned}$$

- A 1 unit increase in x_j increases the "logit" by β_j.
  - Equivalently, multiplies $p_\beta(x)$ by a factor of $\exp(\beta_j)$
- This is a special case of using a *link function* in generalized linear models

Logistic Regression

- Can no longer fit this using the least squares criterion
- Instead, use maximum likelihood: Find β so that $p_\beta(x_i)$ is large whenever $y_i = 1$ and small otherwise
- Formally, maximize $\prod_{i:y_i=1} p_\beta(x_i) \prod_{i:y_i=0} (1 - p_\beta(x_i))$ over all possible β
- Can't be done analytically, but derivatives can be found, so we can optimize

Variants

?

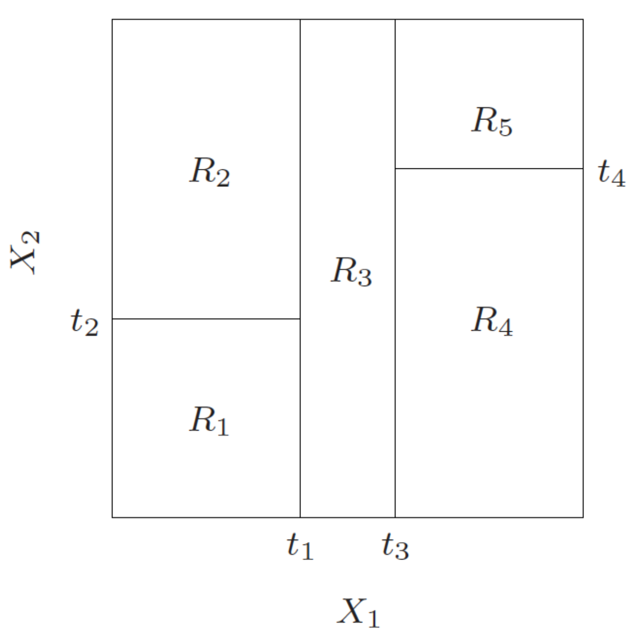
Decision Trees

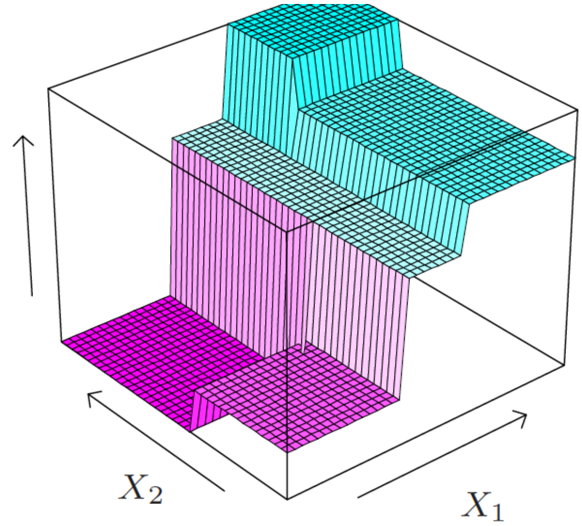
- The local averaging idea in KNN is powerful, but it got overwhelmed in high dimensions
- Can we adapt it so that it only considers a few dimensions that really matter?

Decision Trees

- The basic idea is,
  - Split the input space into nonoverlapping rectangles, $R_1, R_2, \ldots, R_K$
  - For a new x lying in region R_k, estimate the output by

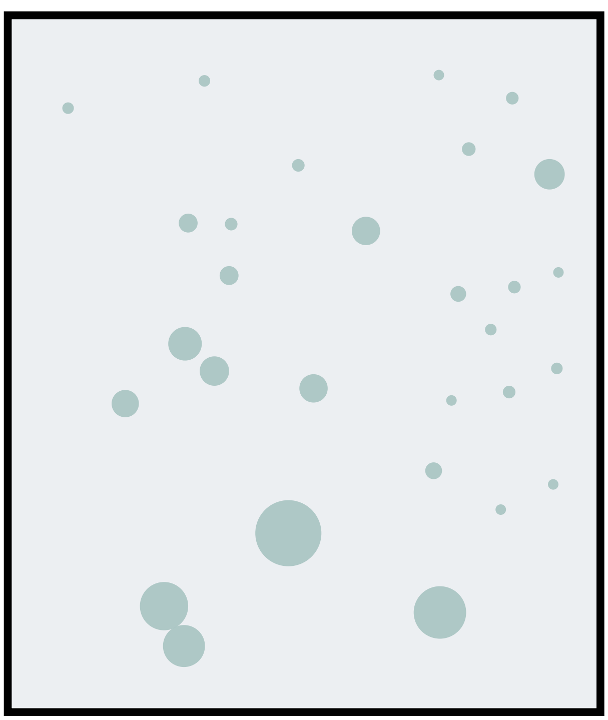
$$\hat{f}\left(x\right) = \frac{1}{|R_k|} \sum_{i : x_i \in R_k} y_i$$





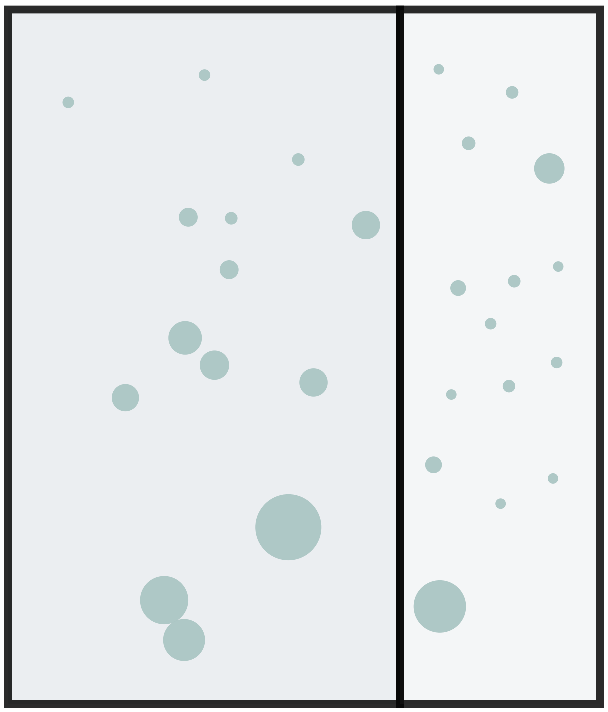
Finding R_k

- The R_k's replace KNN's local neighborhoods
  - Not all variables will be split, helping avoid curse of dimensionality
- Do the splitting recursively, from the top down
- Choose splitting variables and positions so that y_i's are similar within regions and different across them



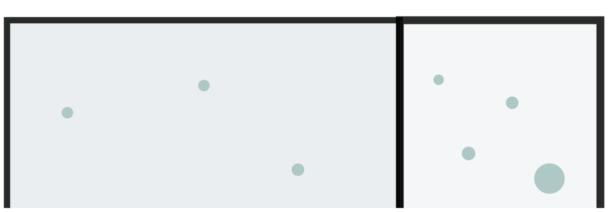
Finding R_k

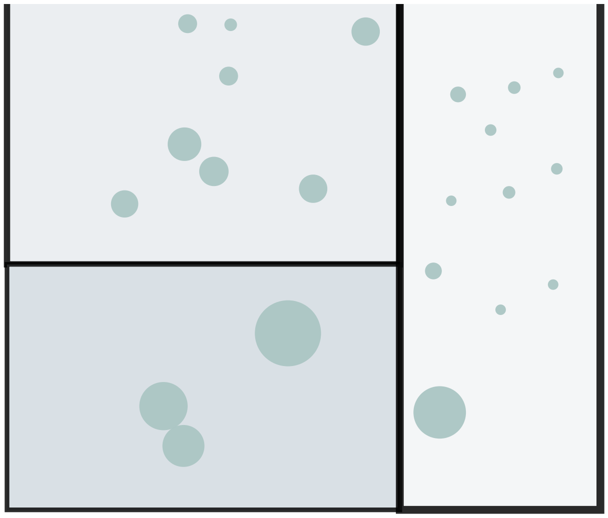
- The R_k's replace KNN's local neighborhoods
  - Not all variables will be split, helping avoid curse of dimensionality
- Do the splitting recursively, from the top down
- Choose splitting variables and positions so that y_i's are similar within regions and different across them



Finding R_k

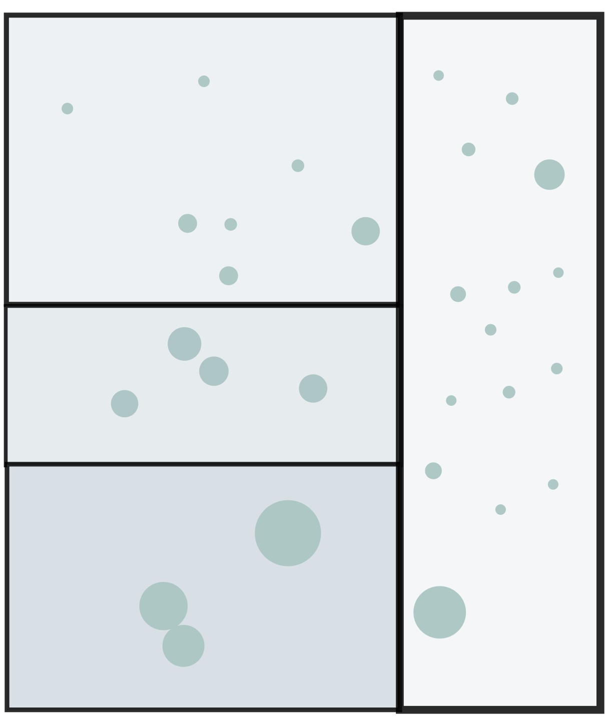
- The R_k's replace KNN's local neighborhoods
  - Not all variables will be split, helping avoid curse of dimensionality
- Do the splitting recursively, from the top down
- Choose splitting variables and positions so that y_i's are similar within regions and different across them





Finding R_k

- The R_k's replace KNN's local neighborhoods
  - Not all variables will be split, helping avoid curse of dimensionality
- Do the splitting recursively, from the top down
- Choose splitting variables and positions so that y_i's are similar within regions and different across them



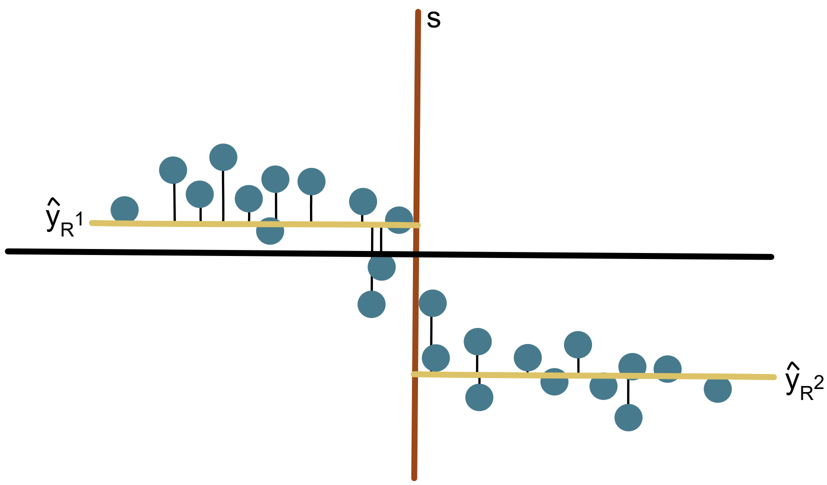
Finding R_k

Define a candidate splits of R_k by the split point s and split dimension j,

- $R_1\left(s, j\right) = \{x \in R_k | x_j < s\}$
- $R_2\left(s, j\right) = \{x \in R_k | x_j \geq s\}$.

The lower this quantity, the better the split,

$$\sum_{\{i|x_i \in R_1(s,j)\}} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{\{i|x_i \in R_2(s,j)\}} \left(y_i - \hat{y}_{R_2}\right)^2$$



Example of a split that would have a good value.

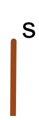
Finding R_k

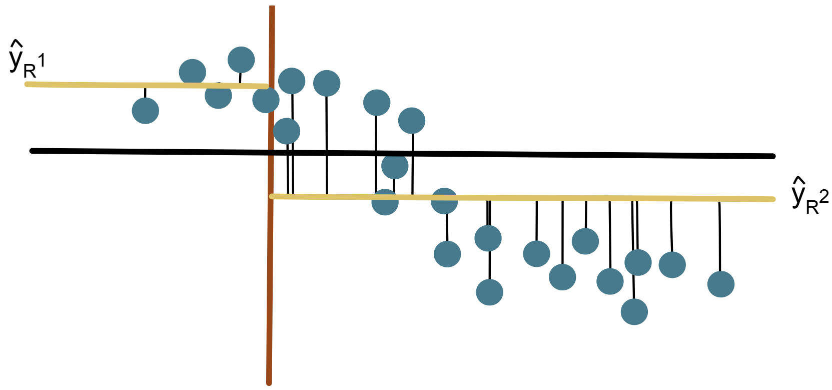
Define a candidate splits of R_k by the split point s and split dimension j,

- $R_1\left(s, j\right) = \{x \in R_k | x_j < s\}$
- $R_2\left(s, j\right) = \{x \in R_k | x_j \geq s\}$.

The lower this quantity, the better the split,

$$\sum_{\{i|x_i \in R_1(s,j)\}} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{\{i|x_i \in R_2(s,j)\}} \left(y_i - \hat{y}_{R_2}\right)^2$$



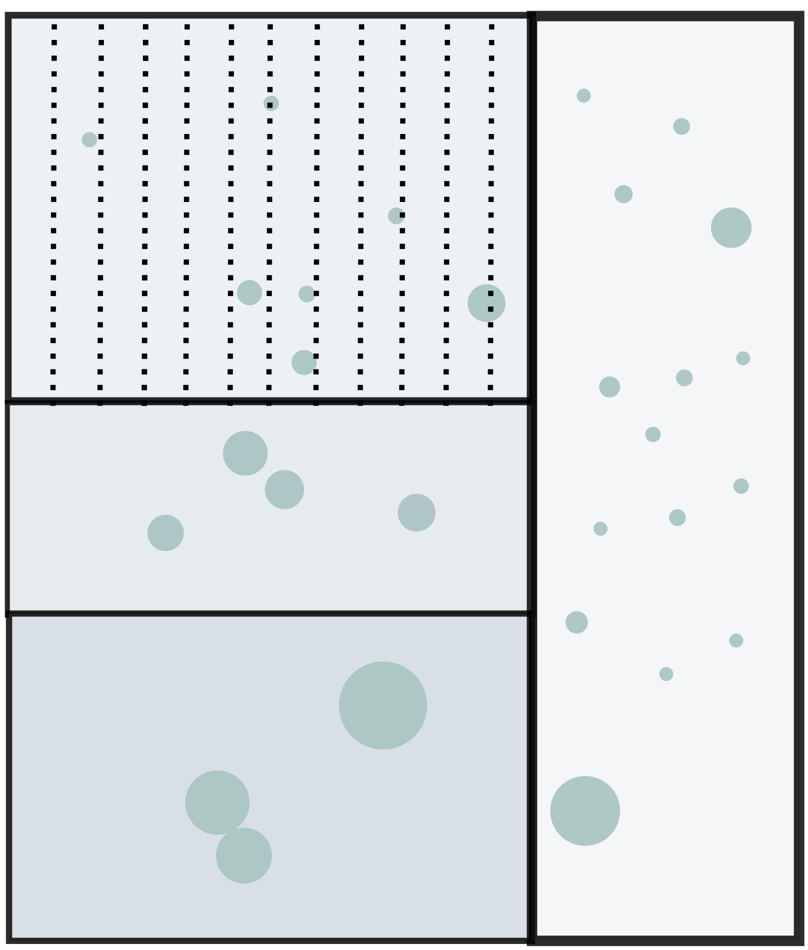


Example of a split that would have a bad value.

Finding R_k

Across R_k, scan over j and s,

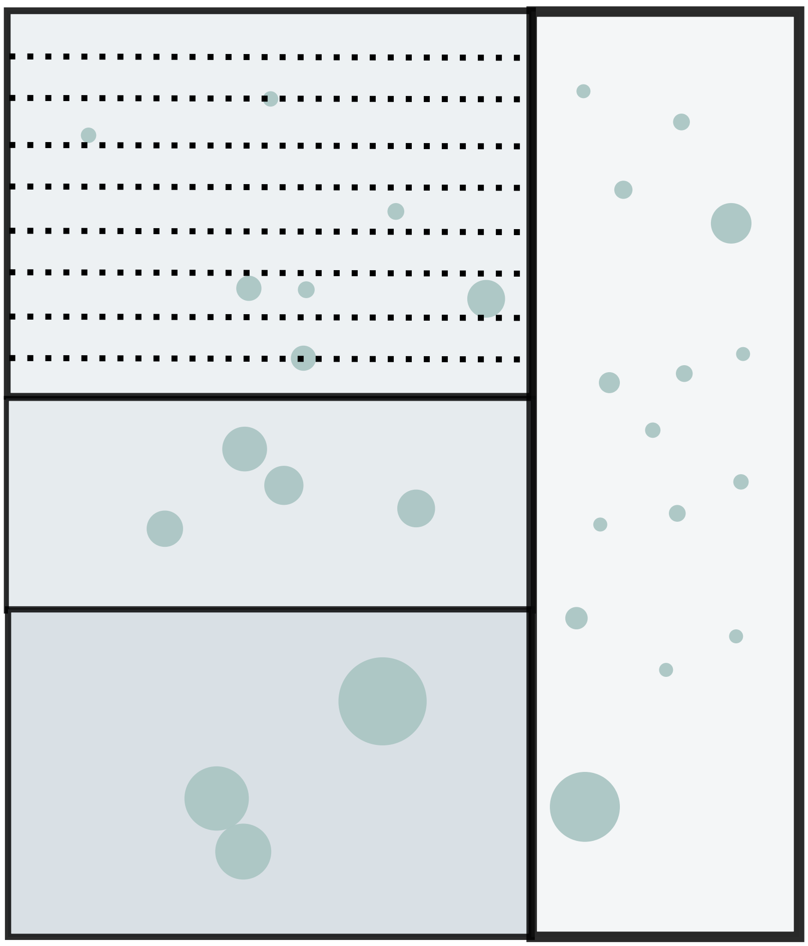
- Choose best of any split, according to the criterion above



Finding R_k

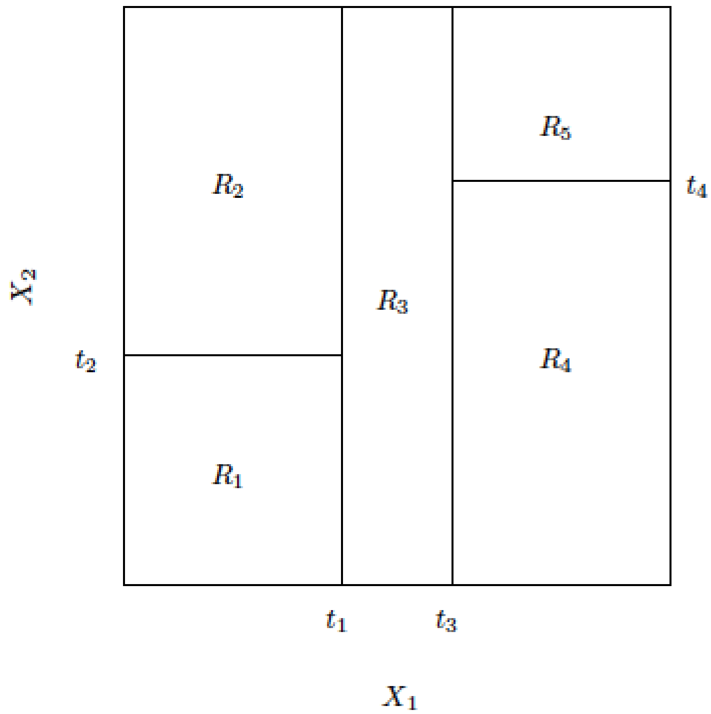
Across R_k, scan over j and s,

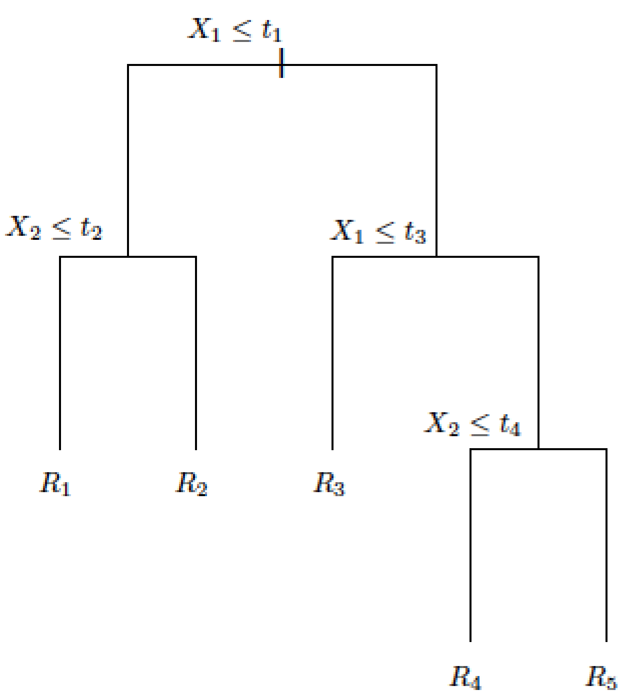
- Choose best of any split, according to the criterion above



R_k and Trees

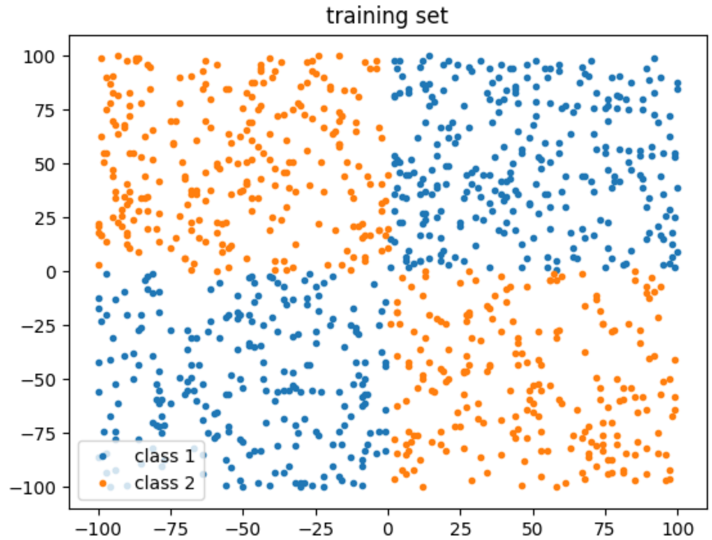
- The region splitting procedure can be interpreted as "growing a tree"
- Each node in the tree corresponds to one rectangular subregion
- Cutting an R_k along dimension j is like splitting a branch





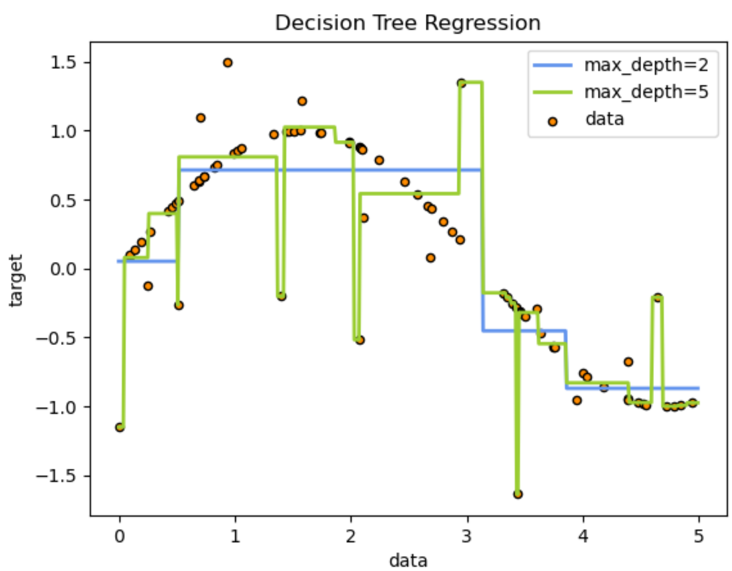
"Cautious" optimism is necessary

- Some splits don't appear useful until we proceed a few splits further down
- This motivates "overgrowing" the tree, and then pruning away unecessary splits



No one split point seems useful if you look at only one dimension. But if you see two, the choice becomes clear.

- "Overgrowing" the tree: too many split can lead to overfitting.



Trees wrap-up

- Both regression and classification
- Performance can be substantially improved by

- Performance can be substantially improved by ensembling; Boosting and Random Forests (not in this lecture)
- Let's go through this demo

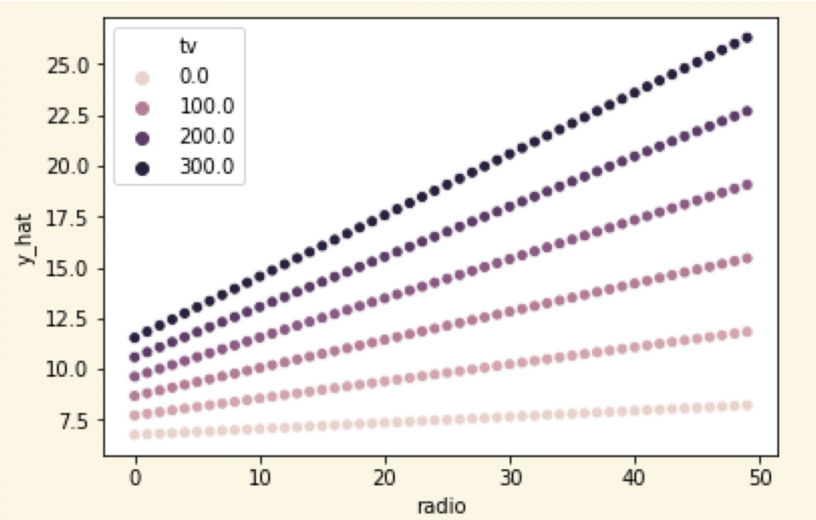
Linear Regression Tricks

Overview

- A few tricks can make linear regression more powerful
  - Interaction terms
  - Categorical predictors
  - Nonlinear basis
  - ℓ^2 and ℓ^1 regularization
- We'll cover the first three

Interactions

Recall the example of model families.



Interactions

Consider the regression,

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 \\ &= \beta_0 + \beta_1 x_1 + \left(\beta_2 + \beta_3 x_1\right) x_2 . \end{aligned}$$

Notice that the slope for x_2 varies smoothly as a function of x_1. The same holds when you reverse the variables.

Categorical predictors

- Suppose a variable x takes on one of L levels
  - e.g. $x \in \{\text{Montreal, Toronto, Vancouver}\}$
- You would like a different mean for each city,

  $f(x) = \beta_0 + \beta_1 \mathbb{I}(x \in \text{Montreal}) +$
  $\beta_2 \mathbb{I}(x \in \text{Toronto}) + \beta_3 \mathbb{I}(x \in \text{Vancouver})$

Categorical predictors

- Suppose a variable x takes on one of L levels
  - e.g. $x \in \{\text{Montreal, Toronto, Vancouver}\}$

- Equivalently, you can set the intercept β_0 to be the Montreal mean, so that

  $$f(x) = \beta_0 + \beta_1 \mathbb{I}(x \in \text{Toronto}) + \beta_2 \mathbb{I}(x \in \text{Vancouver})$$

Categorical predictors

- Equivalently, you can set the intercept β_0 to be the Montreal mean, so that

  $$f(x) = \beta_0 + \beta_1 \mathbb{I}(x = \text{Toronto}) + \beta_2 \mathbb{I}(x = \text{Vancouver})$$
- This can be accomplished by,
  - encode $x = (0, 0)$ for Montreal, $(1, 0)$ for Toronto, and $(0, 1)$ for Vancouver
  - write $f(x) = \beta_0 + x^T \beta$

?

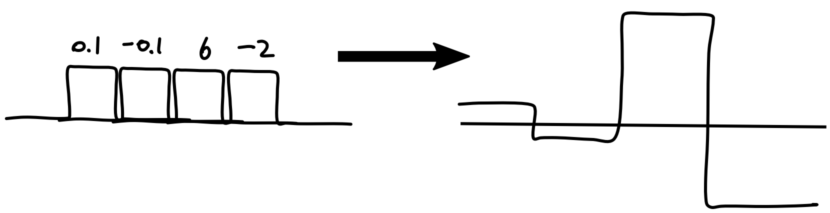
Nonlinearities: Basis Functions

- Usual (linear) model family: $f(x) = x^T \beta$
- Instead, could use $f(x) = h(x)^T \beta$,
  - e.g., $h(x) = \left(1, x_1, x_2, , x_1 x_2, x_1^2, x_2^2\right)$
  - Or, $h(x) = (\sin(x), \cos(x))$
- It's still linear in β, so formulas from before still work (just replace x_i with $h(x_i)$)

Nonlinearities: Local Basis

- Polynomials are global, unbounded functions. It's safer to consider $h\left(x\right)$ that are more local
- E.g., piecewise constant

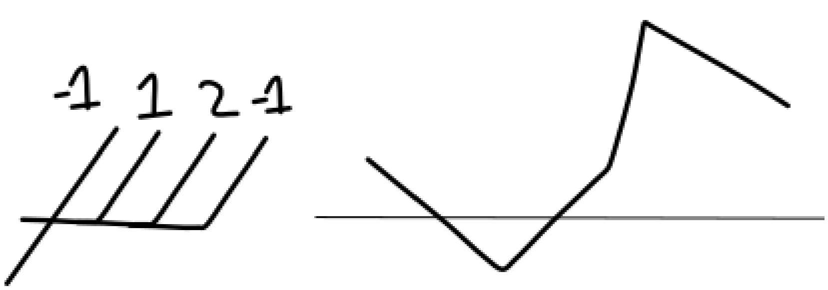
$h\left(x\right)=\left[\mathbb{I}\left(x<c_1\right),\mathbb{I}\left(x\in\left[c_1,c_2\right]\right),\ldots,\mathbb{I}\left(x>c_k\right)\right]$



Nonlinearities: Local Basis

- Polynomials are global, unbounded functions. It's safer to consider $h\left(x\right)$ that are more local
- E.g., piecewise linear

$h\left(x\right)=\left[x,\left(x-c_1\right)_+,\ldots,\left(x-c_k\right)_+\right]$



Nonlinearities: Generalized Additive Models

- In higher-dimensions, it's not obvious what the nonlinear basis should look like
- 💡 Simple decomposition across dimensions, and use nonlinearities only within dimensions:

$$\hat{f}(x) = \sum_{j=1}^{p} \hat{f}_j(x_j),$$

where each $\hat{f}_j$ is fit using nonlinear basis functions.

- Can no longer directly use regression formulas (link, nonlinear features), but a slight (iterative) variant works.

```
keypresscontrol = undefined
```

```
slide = ƒ()
```

```
hl = ▶ Object {highlight: ƒ(e, t, a, c), highlightAuto: ƒ(e, t)
```

```
<style>
```

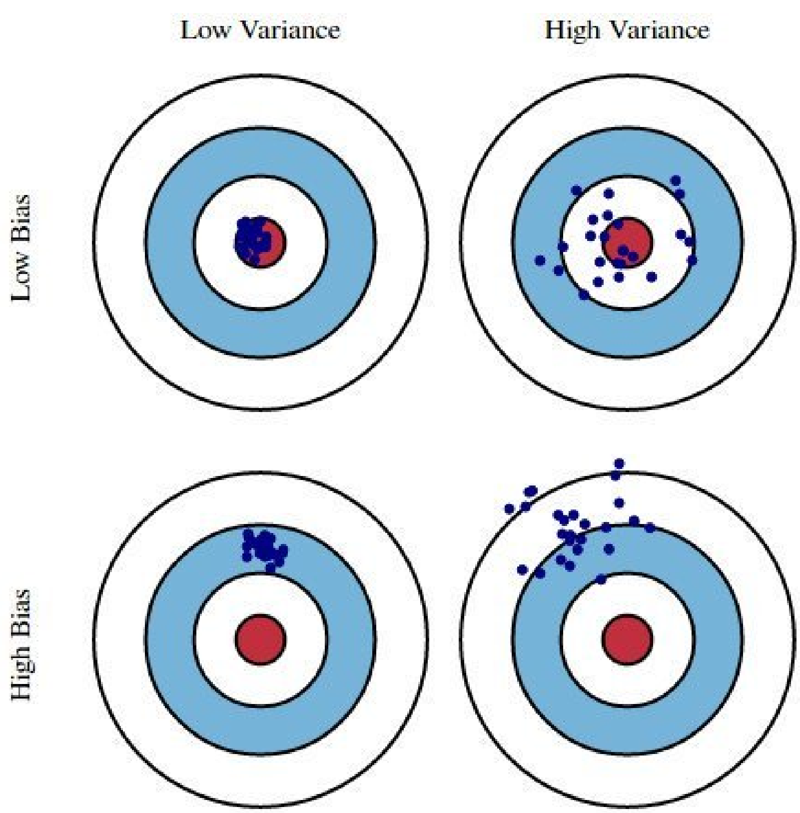
```
mtex = ƒ()
```

```
mtex_block = ƒ()
```

Asides

Bias-variance tradeoff

- The number of variables is one measure of model complexity
  - More variables -> lower bias but higher variance



- *Ridge Regression*: You can also "regularize" the fit, using

$$\hat{\beta} = \left(X^T X + \lambda I\right)^{-1} X^T y$$

Larger λ reduce model complexity:

$$\lambda \to 0,\, \hat{\beta}_{\text{ridge}} \to \hat{\beta}_{\text{OLS}}; \qquad \lambda \to \infty,\, \hat{\beta}_{\text{ridge}} \to 0$$

- *Bias-variance trade-off in ridge*:

$$\text{Bias}(\hat{\beta}_{\text{ridge}}) = -\lambda(X^TX + \lambda I)^{-1}\beta$$

$$\text{Variance}(\hat{\beta}_{\text{ridge}}) = \sigma^2(X^TX + \lambda I)^{-1}X^TX(X^TX + \lambda I)^{-1}$$

- Choice of λ.
  - Minimizing information criteria: estimate the model with many different values for λ and choose the one that minimizes the Akaike or Bayesian Information Criterion (AIC, BIC)

Interactive demo

Locally Linear Regression

- Instead of taking averages within neighborhoods, can fit small linear regressions
- A type of blend between KNN and linear regression