COMPSCI 389
Introduction to Machine Learning

Linear Regression and the Optimization Perspective

Prof. Philip S. Thomas (pthomas@cs.umass.edu)

Review: Regression

- X: **Input** (also called **features**, **attributes**, **covariates**, or **predictors**)
  - Typically, X is a vector, array, or list of numbers or strings.
- Y: **Output** (also called **labels** or **targets**)
  - In regression, Y is a real number.
- An input-output pair is (X, Y).
- Let n, called the **data set size**, be the number of input-output pairs in the data set.
- Let (X_i, Y_i) denote the i^{th} input output pair.
- The complete data set is

$$(X_i, Y_i)_{i=1}^n = \big((X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)\big).$$

Review: Nearest Neighbor (Variants)

- Given a query input x_{query}, find the k nearest points in the training data.
- Return a weighted average of their labels.
  - $k = 1$ is nearest neighbor
  - $k > 1$ with all w_i equal is k-nearest neighbor
  - $k > 1$ with not all w_i equal is weighted k-nearest neighbor
- These algorithms don't pre-process the training data much.
  - They can build data structures like KD-Trees for efficiency.

Linear Regression

- Search for the **line** that is a best fit to the data.
- Different performance measures correspond to different ways of measuring the quality of a fit.
- Sample mean squared error, or the **sum of the squared errors (SSE)** is particularly common:

$$\widehat{\mathrm{MSE}}_n\text{: } \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \text{ and SSE: } \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  - Although not identical, the line that minimizes one also minimizes the other.
- Using sample MSE, this method is called "least squares linear regression."

Linear Regression: What is a line?

$$y = mx + b$$

Prediction, $\hat{y}_i$ Slope, m Input, x_i y-intercept, b

“weights,” or “parameters”, $w = (w_1, w_2)$

$$\hat{y} = w_1 x_i + w_2$$

Models (Review)

- A model is a mechanism that maps input data to predictions.
- ML algorithms take data sets as input and produce models as output.

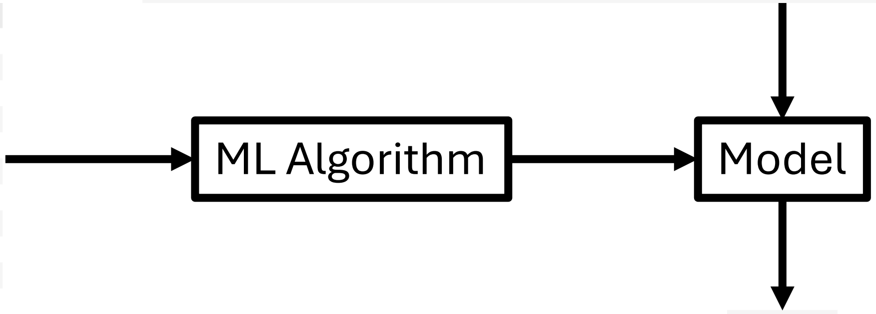
A query can be one **or more** feature vectors.

Query

798.75	817.58	731.98	648.42	751.30	648.67	662.05	773.15	835.25
527.66	443.82	545.88	624.18	420.25	676.80	583.41	395.46	509.80

	physics	biology	history	English	geography	literature	Portuguese	math	chemistry	gpa
0	622.60	491.56	439.93	707.64	663.65	557.09	711.37	731.31	509.80	1.33333
1	538.00	490.58	406.59	529.05	532.28	447.23	527.58	379.14	488.64	2.98333
2	455.18	440.00	570.86	417.54	453.53	425.87	475.63	476.11	407.15	1.97333
3	756.91	679.62	531.28	583.63	534.42	521.40	592.41	783.76	588.26	2.53333
4	584.54	649.84	637.43	609.06	670.46	515.38	572.52	581.25	529.04	1.58667
...	...	...	...	...	...	...	...	...	...	...
43298	519.55	622.20	660.90	543.48	643.05	579.90	584.80	581.25	573.92	2.76333
43299	816.39	851.95	732.39	621.63	810.68	666.79	705.22	781.01	831.76	3.81667
43300	798.75	817.58	731.98	648.42	751.30	648.67	662.05	773.15	835.25	3.75000
43301	527.66	443.82	545.88	624.18	420.25	676.80	583.41	395.46	509.80	2.50000
43302	512.56	415.41	517.36	532.37	592.30	382.20	538.35	448.02	496.39	3.16667

Data Set



3.75000
2.50000

Prediction

Predictions are given for each feature vector in the query.

Parametric Model

- A model "parameterized" by a weight vector w.
- Different settings of w result in different predictions.
- Let $\hat{y} = f_w(x)$
  - 1-dimensional linear case:

$$f_w(x) = w_1 x + w_2$$

Linear Regression: Hyperplanes

- What if we have more than one input feature?
- Let $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,d})$ be a d-dimensional input.
  - We include the i subscript to make it clear that 1,2,... aren't referencing different input vectors, but different elements of one input vector.
- We use a *hyperplane*:

$$f_w(x_i) = w_1 x_{i,1} + w_2 x_{i,2} + \dots + w_d x_{i,d} + w_{d+1}.$$

Slope along the first dimension

Rate of change of the prediction as the first feature increases

Slope along the second dimension
Rate of change of the prediction as the second feature increases

The **offset**, **bias**, or **intercept** term, which gives the prediction when the input features are all zero.

Linear Regression (cont.)

$$f_w(x_i) = w_1 x_{i,1} + w_2 x_{i,2} + \ \dots + w_d x_{i,d} + w_{d+1}.$$

- **Thought**: We don’t want to have to keep remembering a special “intercept” term.
- **Idea**: Drop the intercept term!
    - If you want to include the intercept term, add one more feature to your data set, $x_{d+1} = 1$.
    - If d is the dimension of the input ***with*** this additional feature, we then have:
    $$f_w(x_i) = w_1 x_{i,1} + w_2 x_{i,2} + \ \dots + w_d x_{i,d}$$
    - We can write this as:
    $$f_w(x_i) = \sum_{j=1}^{d} w_j \, x_{i,j}.$$
    - This is called a **dot product** and can be written as $w \cdot x_i$ or $w^T x_i$.

Linear Regression (cont.)

$$\hat{y_i} = f_w(x_i) = \sum_{j=1}^{d} w_j \, x_{i,j}$$

- How many weights (parameters) does the model have?
  - d, the dimension of any one input vector x_i.
  - **Not** n, the number of training data points.

Linear Regression: Optimization Perspective

- Given a parametric model f_w of any form how can we find the weights w that result in the "best fit"?
- Let L be a function called a **loss function**.
  - It takes as input a model (or model weights w)
  - It also takes as input data D
  - It produces as output a real-number describing how *bad* of a fit the model is to the provided data.
- The evaluation metrics we have discussed can be viewed as loss functions. For example, the sample MSE loss function is:

$$L(w, D) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_w(x_i)\right)^2$$

For the sample MSE loss function, this can be *any* parametric model, not just a linear one!

- We phrase this as an **optimization problem**:

$$\text{argmin}_w\, L(w, D)$$

Linear Regression: Optimization Perspective

$$\text{argmin}_w \, L(w, D)$$

- **Recall**: argmin returns the w that achieves the minimum value of $L(w, D)$, not the minimum value of $L(w, D)$ itself.
- This expression describes a *massive* range of ML methods.
  - Supervised, unsupervised, (batch/offline) RL
  - Deep neural networks
  - Large language models and generative AI
- Different problem settings and algorithms in ML correspond to:
  - Different loss functions
  - Different parametric models.
  - Different algorithms for approximating the best weight vector w.

Least Squares Linear Regression (cont.)

- Find the weights w that minimize

$$L(w, D) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_w(x_i)\right)^2$$

Number of training data points

Dimension of each input vector (number of features per row)

$$L(w, D) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} w_j x_{i,j}\right)^2$$

Linear Regression: Least Squares Solvers

- How should one solve this problem?

$$\operatorname{argmin}_w \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} w_j x_{i,j}\right)^2$$

- Answer: “Least squares solvers”
  - Algorithms based on concepts from linear algebra.
  - Extremely effective for solving problems of precisely this form.
  - Beyond the scope of this class.
  - **Only useful for this exact problem.**
    - Not effective when using other parametric models (e.g., not linear)
    - Not effective when using other loss functions / performance metrics.

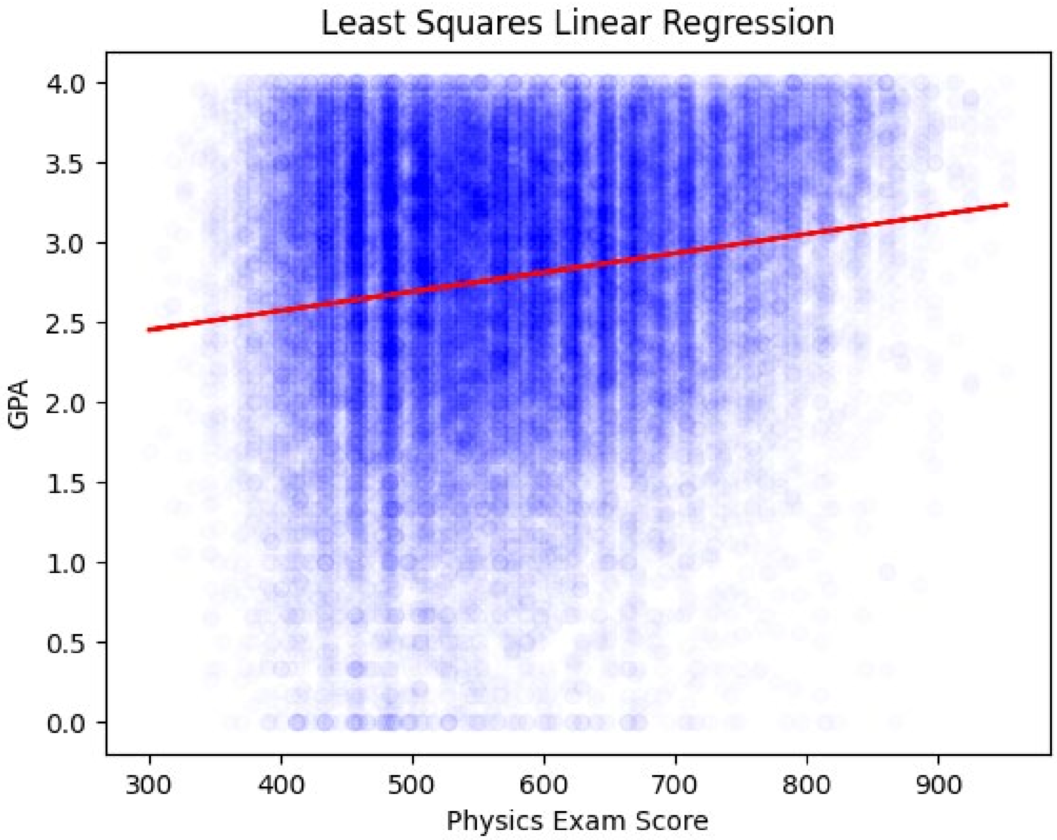
Linear Regression

- How **do we** solve this problem?

$$\text{argmin}_w \ \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{d} w_j x_{i,j}\right)^2$$

- We will study a different approach for solving this problem.
- It *is* less efficient.
- **It applies to almost all loss functions and parametric models of interest.**
- Method: Gradient descent.
  - Soon we will discuss gradient descent.
  - For now, assume we have some way of finding the $\text{argmin}_w L(w, D)$.

Least Squares Linear Regression



Linear Regression vs Weighted k-NN for GPA Prediction

```
Weighted KNN Model:
    Average MSE: 0.571
    MSE Standard Error: 0.004

Linear Regression Model:
    Average MSE: 0.582
    MSE Standard Error: 0.004
```

Very simple method achieves nearly the same performance as a tuned-version of weighted *k*-NN!

Soon, we will consider more complex parametric models that can be even more effective.

Linear Regression Limitation

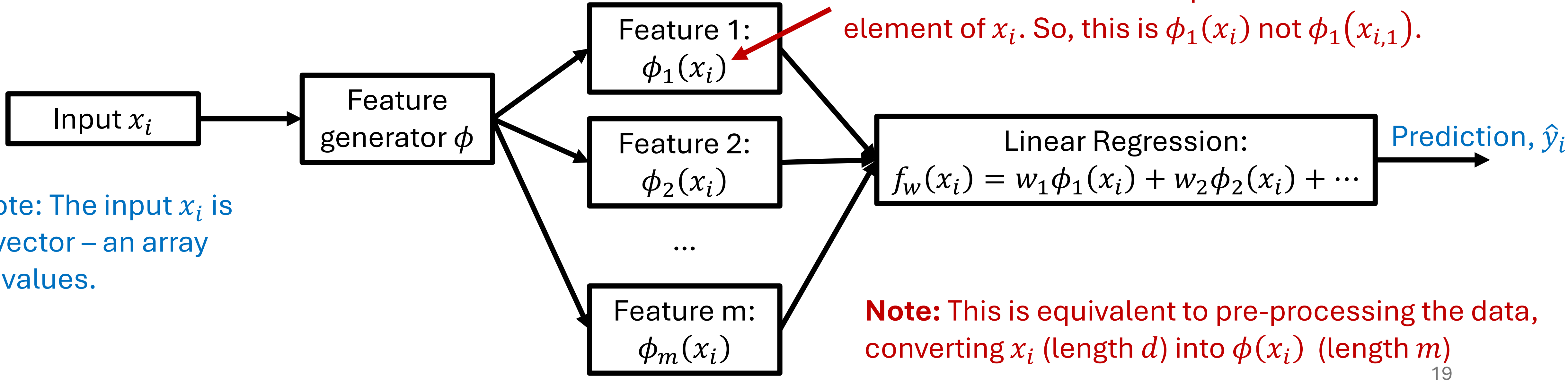
- What if the relationship between the inputs and outputs is not linear (or affine)?
  - Linear: $A_1 x_{i,1} + A_2 x_{i,2} + \cdots + A_n x_{i,n}$
  - Affine: $A_1 x_{i,1} + A_2 x_{i,2} + \cdots + A_n x_{i,n} + b$
    - Equivalent to linear with an additional feature $x_{i,n+1} = 1$.
- **Idea**: Have parametric functions that can represent more than linear functions!

Linear Parametric Model ≠Linear Functions

- **Linear parametric functions** are functions $f_w(x_i)$ that are **linear functions of the weights** $\boldsymbol{w}$.
- They need not be linear functions of the input x_i.

Each feature is a real number (not a vector or array)

Note: Each feature can depend on more than one element of x_i. So, this is $\phi_1(x_i)$ not $\phi_1(x_{i,1})$.



Note: The input x_i is a vector – an array of values.

Note: This is equivalent to pre-processing the data, converting x_i (length d) into $\phi(x_i)$ (length m)

Linear Parametric Model $\neq$ Linear Functions

- **Linear parametric functions** are functions $f_w(x_i)$ that are **linear functions of the weights w**.
- They need not be linear functions of the input x_i.
- That is, a linear parametric model has the form:

$$f_w(x_i) = \sum_{j=1}^{m} w_j \phi_j(x_i)\,,$$

where ϕ takes the input vector x_i as input and produces a vector of m features as output. That is, $\phi_j(x_i)$ is the j^{th} feature output by ϕ.

- ϕ is called the **basis function, feature generator,** or **feature mapping function**.

Note: Explain "basis" on the board. You are not responsible for the material on the board.

Linear Parametric Model

$$f_w(x_i) = \sum_{j=1}^{m} w_j \phi_j(x_i)$$

- Polynomial basis
  - If $x_i \in \mathbb{R}$ then $\phi_j(x_i) = x_i^{j-1}$ so that:
    $$\phi(x_i) = \left[1, x_i, x_i^2, x_i^3, \dots, x_i^{m-1}\right]$$
  - Here $m - 1$ is the **degree** or **order** of the polynomial basis.
  - $f_w(x_i) = w_1 + w_2 x_i + w_3 x_i^2 + w_4 x_i^3 + \cdots + w_m x_i^{m-1}$
  - We are fitting a polynomial to the data!
  - This is a non-linear function of the input x_i
  - This can represent *any* smooth function (if m is big enough).
  - This is a linear function of w.

Linear Parametric Models (cont.)

- What does it mean for a function $g(x, y)$ to be **linear** with respect to an input, x?
  - The slope is constant as x changes.
  - The derivative with respect to x is a constant (does not vary with x)
- Is $g(x, y) = x^2 y^2$ linear *with respect to* (w.r.t.) x?
  - $\frac{\partial g(x,y)}{\partial x} = 2xy^2$, which changes with x, so no.
- Is $g(x, y) = x \sin(y)$ linear w.r.t. x?
  - $\frac{\partial g(x,y)}{\partial x} = \sin(y)$, which does not change with x, so yes!
- Is $f_w(x_i) = \sum_{j=1}^{m} w_j \phi_j(x_i)$ linear w.r.t. w?
  - $\frac{\partial f_w(x_i)}{\partial w_j} = \phi_j(x_i)$, for all j, which does not change with w, so yes!

Linear Parametric Models (cont.)

- Is $f_w(x_i) = \sum_{j=1}^{m} w_j \phi_j(x_i)$ linear w.r.t. x?
  - $\frac{\partial f_w(x_i)}{\partial x_{i,k}} = \sum_{j=1}^{m} w_j \frac{\partial \phi_j(x_i)}{\partial x_{i,k}}$, for all k.
  - If ϕ is linear w.r.t. x then yes, otherwise no.
- Is $f_w(x_i) = w_1 w_2 x_{i,1}^2$ linear w.r.t. w?
  - $\frac{\partial f_w(x_i)}{\partial w_1} = w_2 x_{i,1}^2$
  - **No**. It is linear w.r.t. w_1 but not linear w.r.t. w.
  - Linear w.r.t. w means that the derivative w.r.t. w (a vector) does not depend on w (a vector).
    - Note: The derivative w.r.t. w is

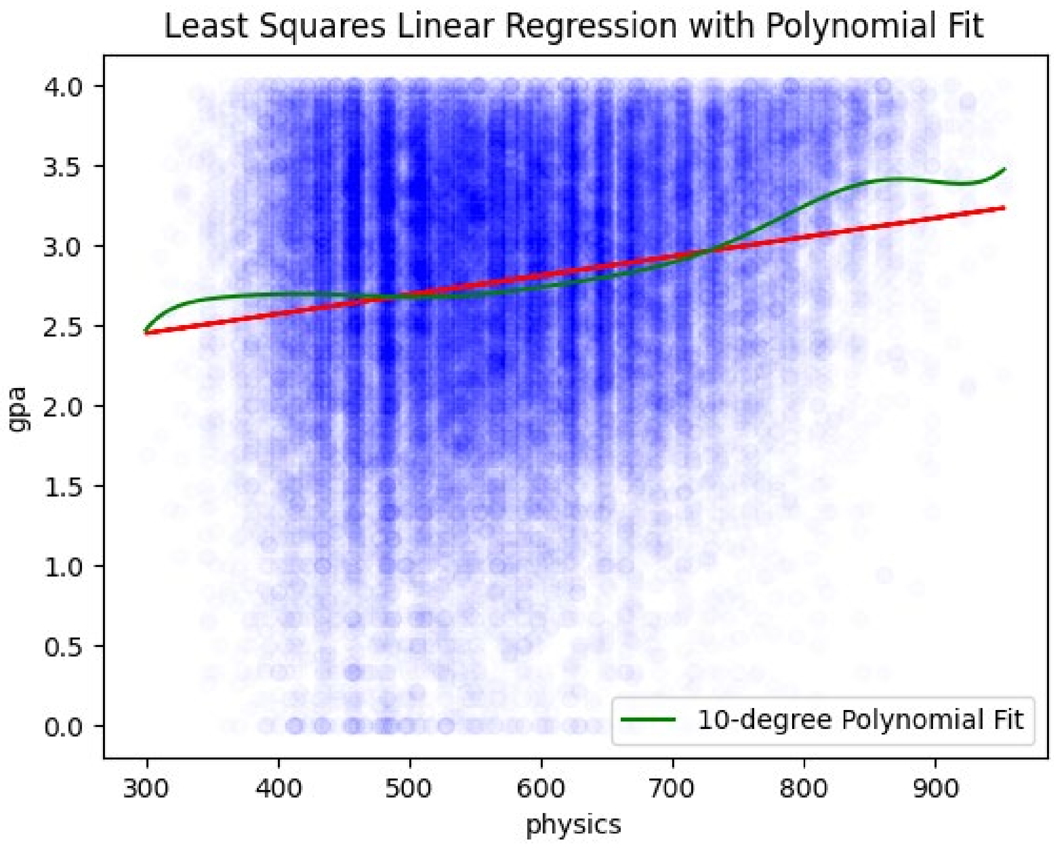
$$\left[\frac{\partial f_w(x_i)}{\partial w_1}, \frac{\partial f_w(x_i)}{\partial w_2}\right]^T$$

Question: Why $x_{i,k}$ instead of $x_{i,j}$?

Answer: It's just a symbol – it could be a smiley face! It represents an integer. We already used the symbol j in $\Sigma_{j=1}^{m}$, and that j is not the same as this value, so we call this k.

This T means "transpose," which just means that this should be viewed as a column not a row (the elements stacked vertically rather than horizontally). This isn't important for this course.

Linear Parametric Models



Linear Parametric Model vs Linear Regression vs Weighted k-NN for GPA Prediction (20-fold cross-validation)

- `Weighted KNN Model:`
  - `Average MSE: 0.571`
  - `MSE Standard Error: 0.004`
- `Linear Regression Model:`
  - `Average MSE: 0.582`
  - `MSE Standard Error: 0.004`
- `Polynomial Regression Model (Degree 4):`
  - `Average MSE: 0.576`
  - `MSE Standard Error: 0.004`

Recall k-NN results:

	k	MSE
0	1	1.152084
1	2	0.853430
2	3	0.764468
3	5	0.688330
4	10	0.631001
5	100	**0.579404**
6	1000	0.581676
7	5000	0.600544

A simple linear model outperforms *k*-NN (not quite a well-tuned weighted *k*-NN)!

Linear Parametric Models

- Pros:
  - Relatively simple.
  - Can represent any smooth function (given the right / enough features).
  - Can use hand-crafted features.
  - Quite efficient to solve for optimal w.
    - Can still use least squares solvers – need not use gradient descent.
  - Extremely fast to generate predictions for new inputs
    - Compute features, take the dot-product with the weights (take the weighted sum)
- Cons:
  - Can be hard to find good features.
  - People often think linear parametric models can only represent lines, and so they think negatively of them.

Parametric vs Nonparametric

- ML algorithms are often categorized into **parametric** and **nonparametric**.
  - In general:
    - Parametric methods use parameterized functions with weights w.
    - Nonparametric methods store the training data or statistics of the training data.
  - More precisely
    - Parametric:
      - Have a fixed number of weights w.
      - Tend to make specific assumptions about the form of the function.
    - Nonparametric:
      - Do not make explicit assumptions about the form of the function.
      - Number of values stored tends to vary with the amount of training data (e.g., storing data).
  - There is some debate about whether some methods are parametric or nonparametric.
    - Linear regression and regression with linear parametric are canonical examples of parametric.
    - Nearest neighbor algorithms are canonical examples of nonparametric.

Multivariate Polynomial Basis

- How does the polynomial basis, ϕ, work if x is multidimensional (an array rather than a number?)
- Multivariate polynomial on inputs x, y:

$$a + bx + cy + dxy + ex^2 + fy^2 + gxy^2 + hx^2y + ix^3 + \cdots$$

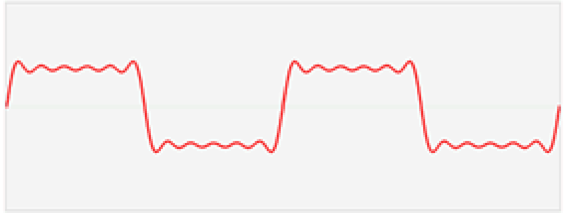
- Multivariate polynomial on input $x_{i,1}, x_{i,2}$:

$$w_1 + w_2 x_{i,1} + w_3 x_{i,2} + w_4 x_{i,1} x_{i,2} + w_5 x_{i,1}^2 + w_6 x_{i,2}^2 + w_7 x_{i,1} x_{i,2}^2 + w_8 x_{i,1}^2 x_{i,2}^2 + w_9 x_{i,1}^3 + \cdots$$

- The expression above is $f_w(x_i)$ for a linear parametric model using the multivariate polynomial basis.
- Notice that some $\phi_j(x_i)$ terms depend on more than one element of x_i!
    - This term is $w_8 \phi_8(x_i)$

Fourier Basis

- Each ϕ_j is a cosine function with a different period.
  - Can optionally include both sine and cosine functions.
- Univariate:
  - $\phi_j(x_i) = \cos(j\pi x)$ if $j \in \{0,1,\dots\}$
  - $\phi_j(x_i) = \cos\big((j-1)\pi x\big)$ if $j \in \{1,2,\dots\}$
- Approximation of a step function (from Wikipedia “Fourier series” page)



Fourier Basis (Multivariate)

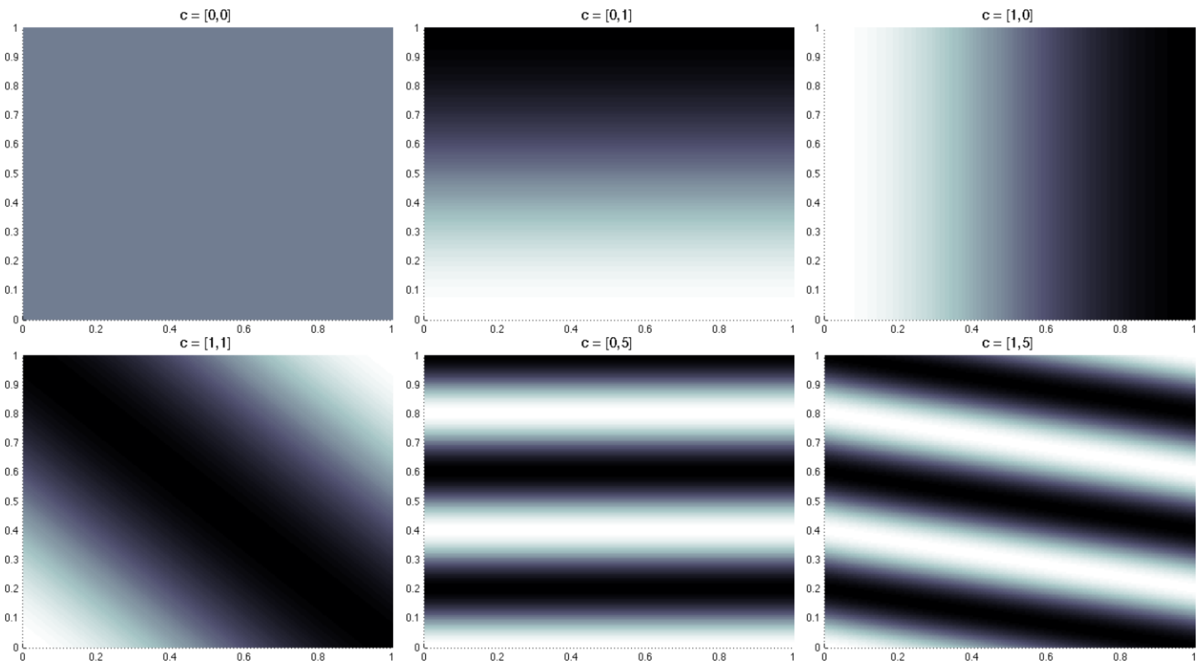


Figure 3: A few example Fourier basis functions defined over two state variables. Lighter colors indicate a value closer to 1, darker colors indicate a value closer to -1.

Feature Engineering

- In some cases, you can hand-craft features
- Examples:
  - Average STEM score
  - Average non-STEM score
- Question: Why might these not be good features?
- Answer: They do not change the functions that can be represented!
  - A weight of w_j on STEM score equates to $\frac{w_j}{4}$ being added to the weights on each of the four STEM exams.
- Effective features are **not** linear combinations of existing features.

End