Regression - le features

Second lecture on regression

Srinandan Dasmahapatra

Linear regression with multiple weights

Arbitrary (linear/non-linear) but FIXED functions

- Recap simple fit of straight line through points, introduce intercept
- **Flexibility** of functions chosen to represent data

- Linear vs non-linear
- Fits with **linear combinations** of functio puts
- Use of **matrix** to represent hypothesised (input-output) relation
- Gradient descent to reduce **loss**: average of **square(prediction - training output)**
- Calculus to compute gradient vector
- Express in numpy

Fitting a straight line through points

Subtracting the average = data entering

- Subtract from each (x,y) the average :

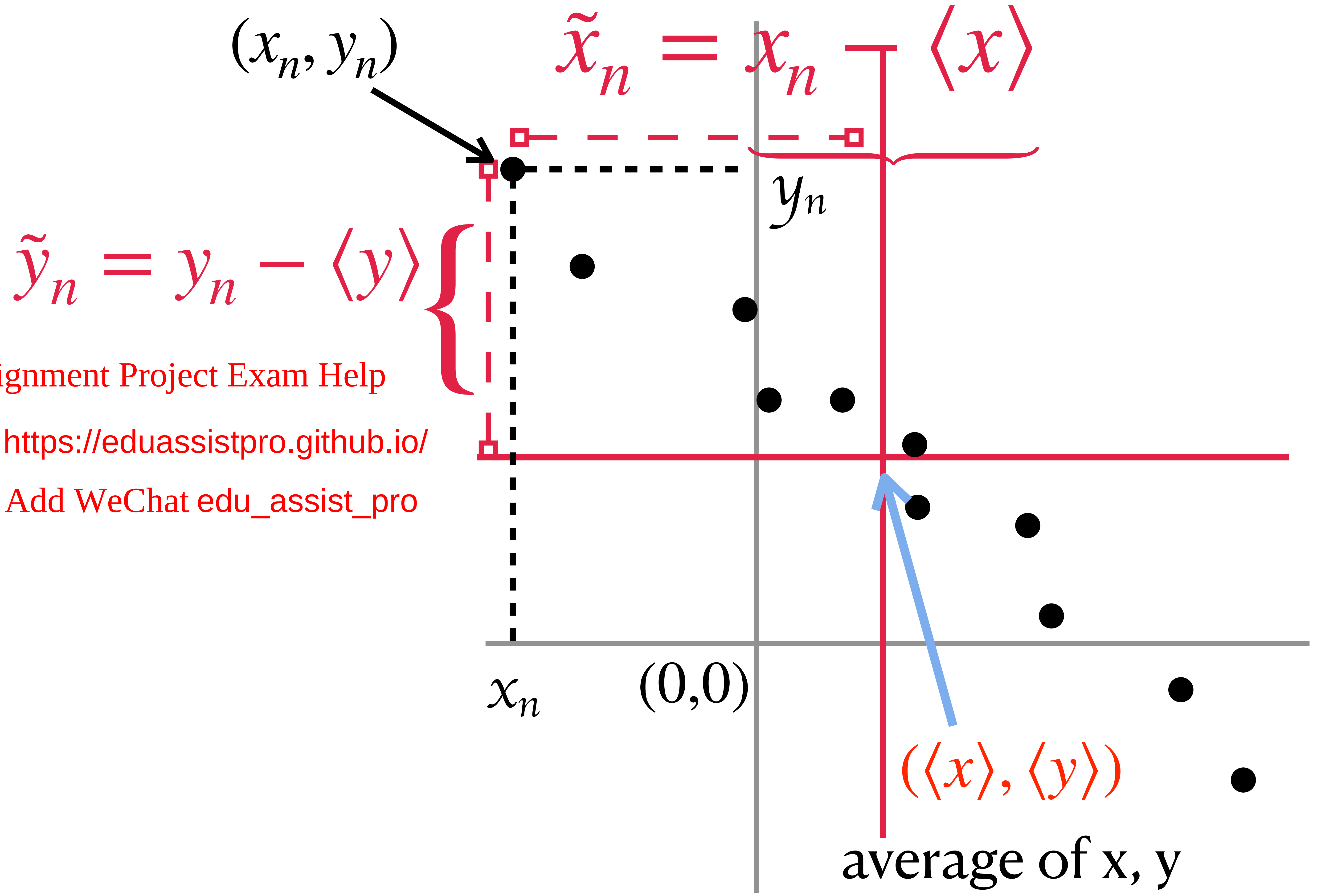
$$(\langle x\rangle, \langle y\rangle) = \frac{1}{N}\sum_n (x_n, y_n)$$

- The origin is shifted to the location of the
- Centred data: line goes through new origin

$$y_n = w_0 + w_1 x_n \Leftrightarrow \langle y\rangle = w_0 + w_1\langle x\rangle$$

- Subtract means:

$$y_n - \langle y\rangle = w_1(x_n - \langle x\rangle) \Leftrightarrow \tilde{y}_n = w_1 \tilde{x}_n$$



Flexibility of polynomials

$$y = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

- Changing each weight w_i alters the shape of the function
- Each power $f_j(x) := x^j$
- $y = w_0 + w_1 f_1(x) + w_2 f_2(x) + \cdots + w_M f_M$
- w_i is called "feature-touching" $i \geq 1$

Linear regression with non-linear functions - 1

What is linearity?

- $f(x) = wx \Rightarrow \begin{cases} (1.)\ f(x_1 + x_2) = w(x_1 + x_2) = wx_1 + wx_2 = f(x_1) + f(x_2) \\ (2.)\ f(ax) = af(x) \end{cases}$ (linear)

- Complex relationships between inputs and outputs not captured by linear functions
- $g(x) = wx^2 \Rightarrow \begin{cases} g(x_1 + x_2) = w(x_1^2 + 2x_1x_2 + x_2^2) \neq wx_1^2 + wx_2^2 = g(x_1) + g(x_2); \\ g(ax) = a^2g(x) \neq ag(x) \end{cases}$

  (non-linear in x)
- But both *f*, *g* are linear in *w*

Linear regression with non-linear functions - 2

$\hat{y}_n = w_0 + w_1\phi_1(x_n) + w_2\phi_2(x_n) + \cdots + w_p\phi_p(x_n),$ where $x_n \in \mathbb{R}^d, \hat{y}_n \in \mathbb{R},$ and $\phi_i : \mathbb{R}^d \to \mathbb{R}$

- Instead of $f_j(x) := x^j$, choose arbitrary functions $\phi_j(x)$

- $\hat{y}_1 = w_0 \cdot 1 + w_1\phi_1(x_1) + w_2\phi_2(x_1)$ … $x_1)$, first data point $x_1 \in \mathbb{R}^d$
- $\hat{y}_2 = w_0 \cdot 1 + w_1\phi_1(x_2) + w_2\phi_2(x_2) + \cdots + w_p\phi_p(x_2)$, second data point $x_2 \in \mathbb{R}^d$
- $\hat{y}_N = w_0 \cdot 1 + w_1\phi_1(x_N) + w_2\phi_2(x_N) + \cdots + w_p\phi_p(x_N)$, last ($N$-th) data point
- Create column vectors $\hat{\mathbf{y}} := (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)^\top \in \mathbb{R}^N$, $\mathbf{w} := (w_0, w_1, w_2, \cdots, w_p)^\top \in \mathbb{R}^{p+1}$
- Write out $(N \times (p+1))$ matrix $\mathbf{A}$ such that $\mathbf{A}\mathbf{w} = \hat{\mathbf{y}}$.

Matrix A is called a design matrix

$$w_0 + \sum_{j=1}^{p} w_j \phi_j(x_n) = \hat{y}_n, \quad \mathcal{D} := \{x_n, y_n\}_{n=1,\ldots,N}$$

Express the collection of proposed functions for each input-output pair as matrix form:

$$\underbrace{\begin{pmatrix} 1 & \phi_1(x_1) & \cdots & \phi_p(x_1) \\ 1 & \phi_1(x_2) & \cdots & \phi_p(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_N) & \cdots & \phi_p(x_N) \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{pmatrix}}_{\mathbf{w}} = \underbrace{\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{pmatrix}}_{\hat{\mathbf{y}}}$$

Matrix A is called a design matrix

Minimise mean squared residuals $\frac{1}{N}\|\mathbf{y} - \hat{\mathbf{y}}\|^2$ to find weights w

$$L(w_0, w_1, \ldots, w_p) := \frac{1}{N}\sum_{n=1}^{N} r_n^2(w) = \frac{1}{N}\sum_{n=1}^{N}\left(\overbrace{w_0 + \sum_{j=1}^{p} w_j \phi_j(x_n)}^{\hat{y}_n} - y_n\right)^2$$

- Exercise: show that the loss functi ic in each of the weig $w_0, w_1, \ldots, w_p$

- Exercise: deduce that the gradient vector $\nabla_{\mathbf{w}}$ of partial derivatives: $\frac{\partial}{\partial w_k} L(w_0, w_1, \ldots, w_p)$ is linear in the weights $w_k, k = 0,1,\ldots,p$.

- Exercise: go through the derivation (next slide): $[\nabla_{\mathbf{w}} L(\mathbf{w})]_k = (2/N)\sum_{n=1}^{N} r_n(\mathbf{w})\phi_k(x_n)$

Taking partial derivatives:

$$[\nabla_{\mathbf{w}} L(\mathbf{w})]_k := \frac{\partial L(\mathbf{w})}{\partial w_k} = (2/N) \sum_{n=1}^{N} r_n(\mathbf{w}) \phi_k(x_n)$$

Recall: design matrix A maps weights to predictions

$$w_0 + \sum_{j=1}^{p} w_j \phi_j(x_n) = \hat{y}_n, \quad \mathcal{D} := \{x_n, y_n\}_{n=1,\ldots,N}$$

Express the collection of proposed functions for each input-output pair as matrix form.
Each **column** of design matrix: **feature** transform on inputs; each **row** is a data-point

$$\underbrace{\begin{pmatrix} 1 & \phi_1(x_1) & \cdots & \phi_p(x_1) \\ 1 & \phi_1(x_2) & \cdots & \phi_p(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_N) & \cdots & \phi_p(x_N) \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{pmatrix}}_{\mathbf{w}} = \underbrace{\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{pmatrix}}_{\hat{\mathbf{y}}}$$

Gradient in terms of design matrix: $[\nabla_{\mathbf{w}} L(\mathbf{w})]_k := \dfrac{\partial L(\mathbf{w})}{\partial w_k} = (2/N)\sum_{n=1}^{N} r_n(\mathbf{w})\phi_k(x_n)$

$$\underbrace{\begin{pmatrix} \frac{\partial}{\partial w_0} L \\ \frac{\partial}{\partial w_1} L \\ \vdots \\ \frac{\partial}{\partial w_p} L \end{pmatrix}^{\top}}_{(\nabla_{\mathbf{w}} L)^{\top}} = \frac{2}{N} \underbrace{\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{pmatrix}^{\top}}_{\mathbf{r}^{\top}} \underbrace{\begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_p(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_p(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ & \phi_1(x_N) & \cdots & \phi_p(x_N) \end{pmatrix}}_{\mathbf{A}}$$

single weights: y = w*x

```
[27]: def loss_slope_w1(w1, Xtrain, ytrain):
          return (2/len(Xtrain))*(np.dot(w1*Xtrain - ytrain, Xtrain))
```

residuals

Gradient in terms of design matrix: $[\nabla_{\mathbf{w}} L(\mathbf{w})]_k := \dfrac{\partial L(\mathbf{w})}{\partial w_k} = (2/N) \sum_{n=1}^{N} r_n(\mathbf{w}) \phi_k(x_n)$

$$\underbrace{\begin{pmatrix} \partial_{w_1} L \\ \partial_{w_2} L \\ \vdots \\ \partial_{w_p} L \end{pmatrix}^{\top}}_{(\nabla_{\mathbf{w}} L)^{\top}} = \frac{2}{N} \underbrace{\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{pmatrix}^{\top} \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_p(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & \cdots & \phi_p(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_p(x_N) \end{pmatrix}}_{\mathbf{A}}$$

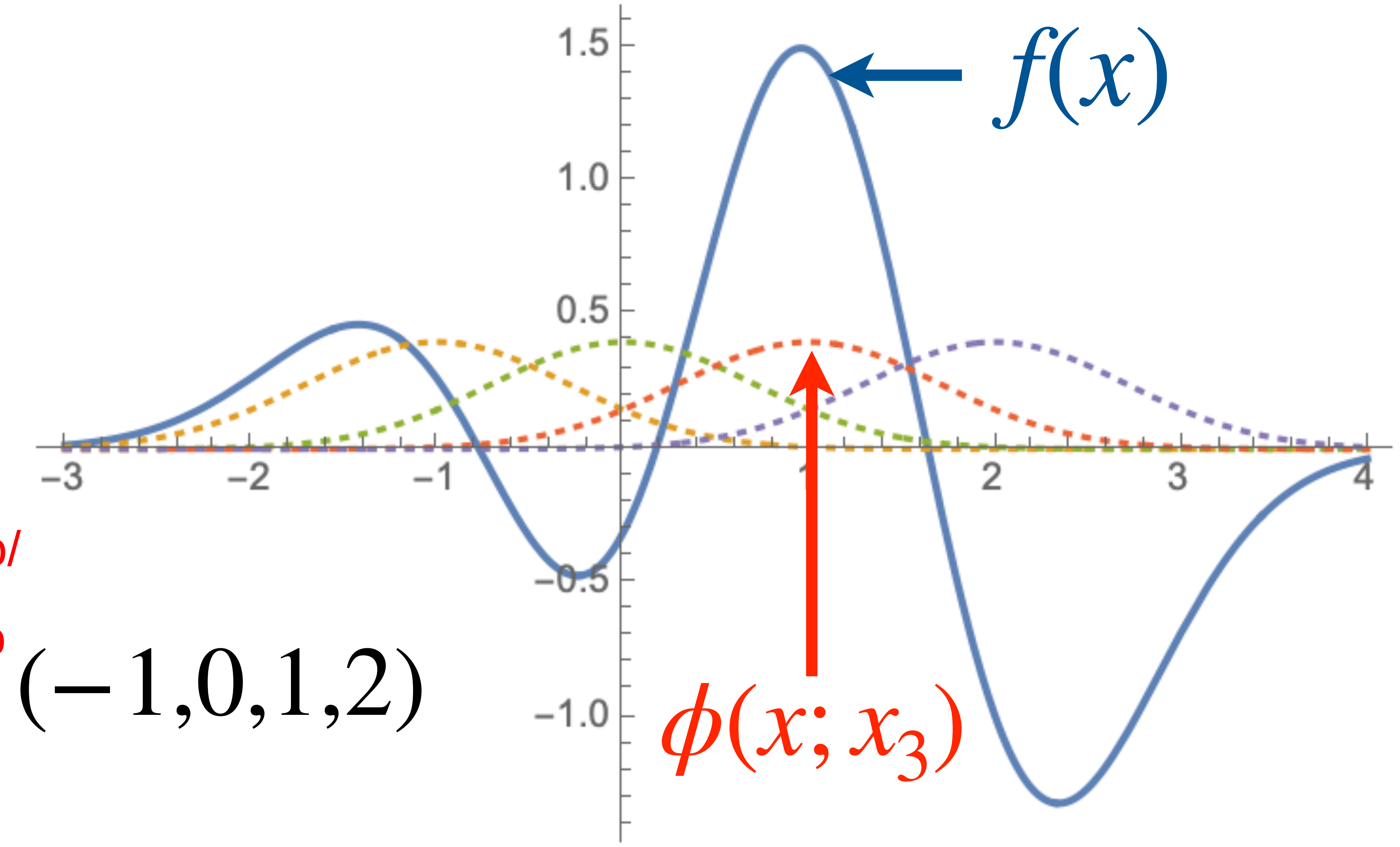
$(\mathbf{a}^{\top}\mathbf{b})^{\top} =$

multiple weights: y = A*w

Choosing features $\phi_j(x_n)$

A few choices

- Monomials $f_j(x_n) := x_n^j$ (seen before)
- Radial basis functions $\phi(x; x_n) = g\left(\left\|\frac{x - x_n}{a}\right\|\right)$
  - choose $g(x) = \exp(-x^2)$, $a = 1$, $(x_1, x_2, x_3, x_4) = (-1,0,1,2)$
  - $f(x) = \sum_{n=1}^{4} w_n \phi(x, x_n)$, $(w_1, w_2, w_3, w_4) = (2, -4, 7, -5)$
  - Local — influence of x_n restricted, unlike monomials; kernel for similarity/“blur”
- Orthogonal polynomials such as Chebyshev, Bessel, etc.



Readings for regression

- First Course in Machine Learning (FCML) — Rogers, Girolami. Chapter 1.
- Page 299-300 of Bishop, Pattern Recognition and Machine Learning (PRML)
- Geron, Hands-on Machine Learni t-Learn, Keras and Tensorflow, chapter 4 (with code on GitHub) — 20 pa

Revisiting gradient descent for linear regression

When the gradient vanishes

- Later: Revisit problem from perspective of linear algebra
- But first, a first look at classification next, with logistic/softmax regression