香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

COMP 5212
Machine Learning
Lecture 4

Generalized Linear Models, Kernel Methods

Junxian He

Feb 14, 2024

Recap

- Linear Regression $h_\theta(x) = \theta^T x$
- Logistic Regression $h_\theta(x) = g(\theta^T x)$
- Multi-class Classification Regression $h_\theta(x) = softmax(\theta_1^T x, \cdots, \theta_k^T x)$

Recap

- Linear Regression $h_\theta(x) = \theta^T x$ $\qquad \theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$

- Logistic Regression $h_\theta(x) = g(\theta^T x)$

- Multi-class Classification Regression $h_\theta(x) = softmax(\theta_1^T x, \cdots, \theta_k^T x)$

Recap

- Linear Regression $h_\theta(x) = \theta^T x$ $\qquad \theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
- Logistic Regression $h_\theta(x) = g(\theta^T x)$ $\qquad \theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$
- Multi-class Classification Regression $h_\theta(x) = softmax(\theta_1^T x, \cdots, \theta_k^T x)$

Recap

- Linear Regression $h_\theta(x) = \theta^T x$ $$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$
- Logistic Regression $h_\theta(x) = g(\theta^T x)$ $$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$
- Multi-class Classification Regression $h_\theta(x) = softmax(\theta_1^T x, \cdots, \theta_k^T x)$

$$\theta_k := \theta_k + \alpha \sum_{i=1}^{n} (1\{y^{(i)} = k\} - h_\theta(x)_k) x^{(i)}$$

Recap

- Linear Regression $h_\theta(x) = \theta^T x$ $$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

- Logistic Regression $h_\theta(x) = g(\theta^T x)$ $$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

- Multi-class Classification Regression $h_\theta(x) = softmax(\theta_1^T x, \cdots, \theta_k^T x)$

$$\theta_k := \theta_k + \alpha \sum_{i=1}^{n} (1\{y^{(i)} = k\} - h_\theta(x)_k) x^{(i)}$$

Is this coincidence?

Generalized Linear Models

We're given features $x \in \mathbb{R}^{d+1}$ and a target y. We want a model.
We first we pick a distribution based on y's type.

Generalized Linear Models

We're given features $x \in \mathbb{R}^{d+1}$ and a target y. We want a model.
We first we pick a distribution based on y's type.

- We assume $y \mid x; \theta$ distributed as an exponential family.
  - Binary $\mapsto$ Bernoulli
  - Multiple Classses $\mapsto$ Multinomial
  - Real $\mapsto$ Gaussian
  - Counts $\mapsto$ Poisson
  - $\mathbb{R}_+$ $\mapsto$ Gamma, Exponential
  - Distributions $\mapsto$ Dirichlet

Exponential Family — Recap

Rough Idea *"If P has a a special form, then inference and learning come for free"*

$$P(y;\eta) = b(y)\exp\left\{\eta^T T(y) - a(\eta)\right\}.$$

η: natural parameter or canonical parameter

Here y, $a(\eta)$, and $b(y)$ are scalars. $T(y)$ same dimension as η.

$T(y)$ is called the **sufficient statistic**. holds all information the data provides with regard to the unknown parameter values

$b(y)$ is called the **base measure** – does *not* depend on η.

$a(\eta)$ is called the **log partition function** – does *not* depend on y.

$$1 = \sum_y P(y;\eta) = e^{-a(\eta)} \sum_y b(y)\exp\left\{\eta^T T(y)\right\}$$

$$\implies a(\eta) = \log \sum_y b(y)\exp\left\{\eta^T T(y)\right\}$$

Generalized Linear Models

We're given features $x \in \mathbb{R}^{d+1}$ and a target y. We want a model. We first we pick a distribution based on y's type.

- We assume $y \mid x; \theta$ distributed as an exponential family.
  - Binary $\mapsto$ Bernoulli
  - Multiple Classses $\mapsto$ Multinomial
  - Real $\mapsto$ Gaussian
  - Counts $\mapsto$ Poisson
  - $\mathbb{R}_+ \mapsto$ Gamma, Exponential
  - Distributions $\mapsto$ Dirichlet

- Our model is *linear* beacuse we make the natural parameter $\eta = \theta^T x$ in which $\theta, x \in \mathbb{R}^{d+1}$.

Generalized Linear Models

inference $h_\theta(x) = \mathbb{E}[y \mid x; \theta]$ is the **output**.

learn $\max_\theta \log p(y \mid x; \theta)$ by maximum likelihood.

Generalized Linear Models

inference $h_\theta(x) = \mathbb{E}[y \mid x; \theta]$ is the **output**.

learn $\max_\theta \log p(y \mid x; \theta)$ by maximum likelihood.

$$P(y; \eta) = b(y) \exp\left\{\eta^T T(y) - a(\eta)\right\}.$$

$$a(\eta) = \log \sum_y b(y) \exp\left\{\eta^T T(y)\right\}$$

Then, taking derivatives

$$\nabla_\eta a(\eta) = \frac{\sum_y T(y) b(y) \exp\{\eta^T T(y)\}}{\sum_y b(y) \exp\{\eta^T T(y)\}} = \mathbb{E}[T(y); \eta]$$

Generalized Linear Models

inference $h_\theta(x) = \mathbb{E}[y \mid x; \theta]$ is the **output**.

learn $\max_\theta \log p(y \mid x; \theta)$ by maximum likelihood.

$$P(y; \eta) = b(y) \exp\left\{\eta^T T(y) - a(\eta)\right\}.$$

$$a(\eta) = \log \sum_y b(y) \exp\left\{\eta^T T(y)\right\}$$

Then, taking derivatives

$$\nabla_\eta a(\eta) = \frac{\sum_y T(y) b(y) \exp\{\eta^T T(y)\}}{\sum_y b(y) \exp\{\eta^T T(y)\}} = \mathbb{E}[T(y); \eta]$$

$T(y) = y$ for most of the examples you will see in this course

Generalized Linear Models

inference $h_\theta(x) = \mathbb{E}[y \mid x; \theta]$ is the **output**.

learn $\max_\theta \log p(y \mid x; \theta)$ by maximum likelihood.

Generalized Linear Models

inference $h_\theta(x) = \mathbb{E}[y \mid x; \theta]$ is the **output**.

learn $\max_\theta \log p(y \mid x; \theta)$ by maximum likelihood.

algorithm: SGD $\theta^{(t+1)} = \theta^{(t)} + \alpha \left(y^{(i)} - h_{\theta^{(t)}}(x^{(i)}) \right) x^{(i)}$

Constructing GLMs

Constructing GLMs

- Pick an exponential family distribution given the type of y (Possion, Multinomial, Gaussian…)

Constructing GLMs

- Pick an exponential family distribution given the type of y (Possion, Multinomial, Gaussian…)
- $\eta = \theta^T x$, or $\eta_i = \theta_i^T x$

Constructing GLMs

- Pick an exponential family distribution given the type of y (Possion, Multinomial, Gaussian…)
- $\eta = \theta^T x$, or $\eta_i = \theta_i^T x$
- Training with maximum likelihood estimation

Constructing GLMs

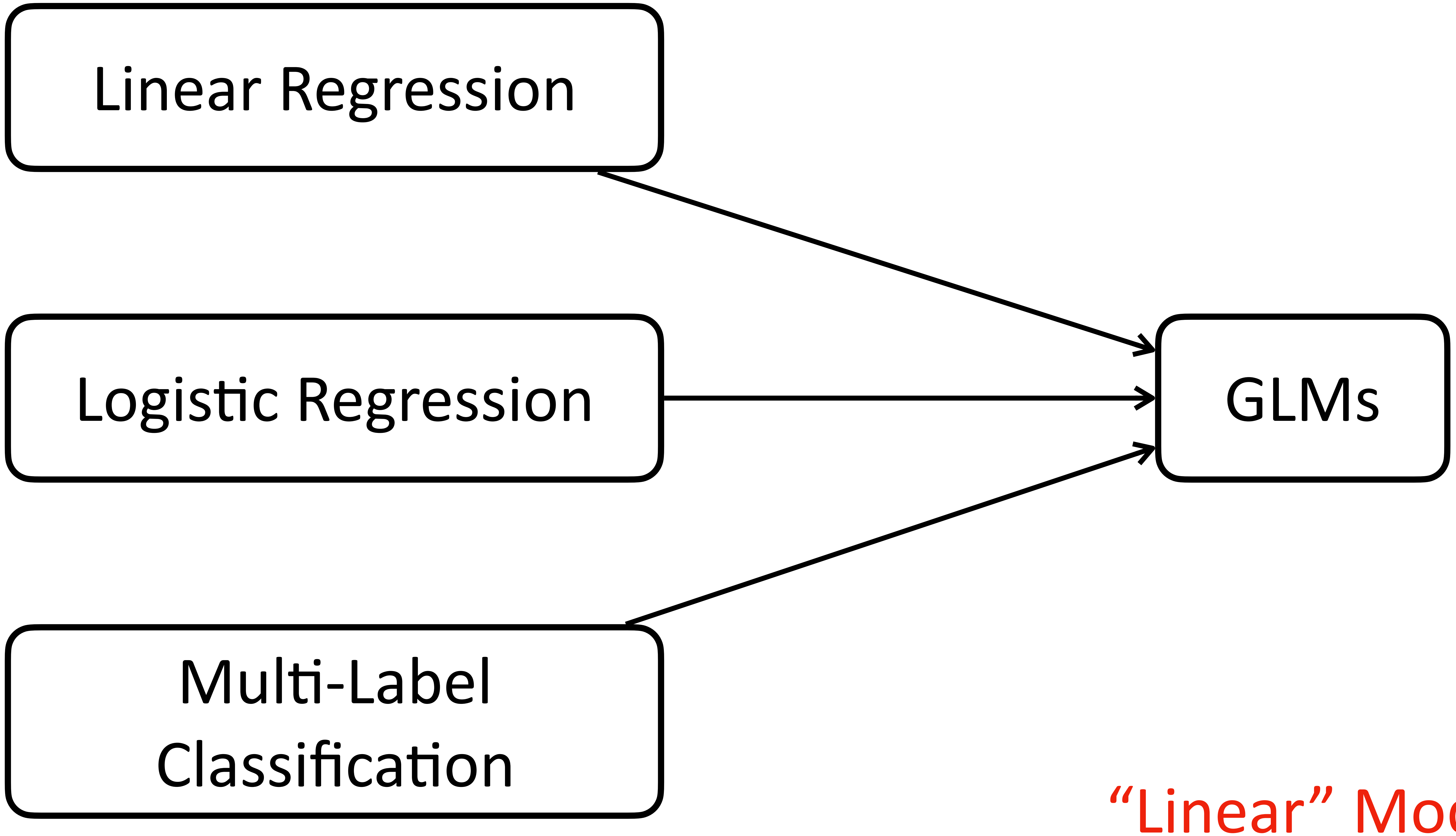
- Pick an exponential family distribution given the type of y (Possion, Multinomial, Gaussian…)
- $\eta = \theta^T x$, or $\eta_i = \theta_i^T x$
- Training with maximum likelihood estimation
- Inference: $h(x) = E[y \mid x]$

Constructing GLMs

- Pick an exponential family distribution given the type of y (Possion, Multinomial, Gaussian…)
- $\eta = \theta^T x$, or $\eta_i = \theta_i^T x$
- Training with maximum likelihood estimation
- Inference: $h(x) = E[y \mid x]$

Enjoy closed-form solution for various statistics

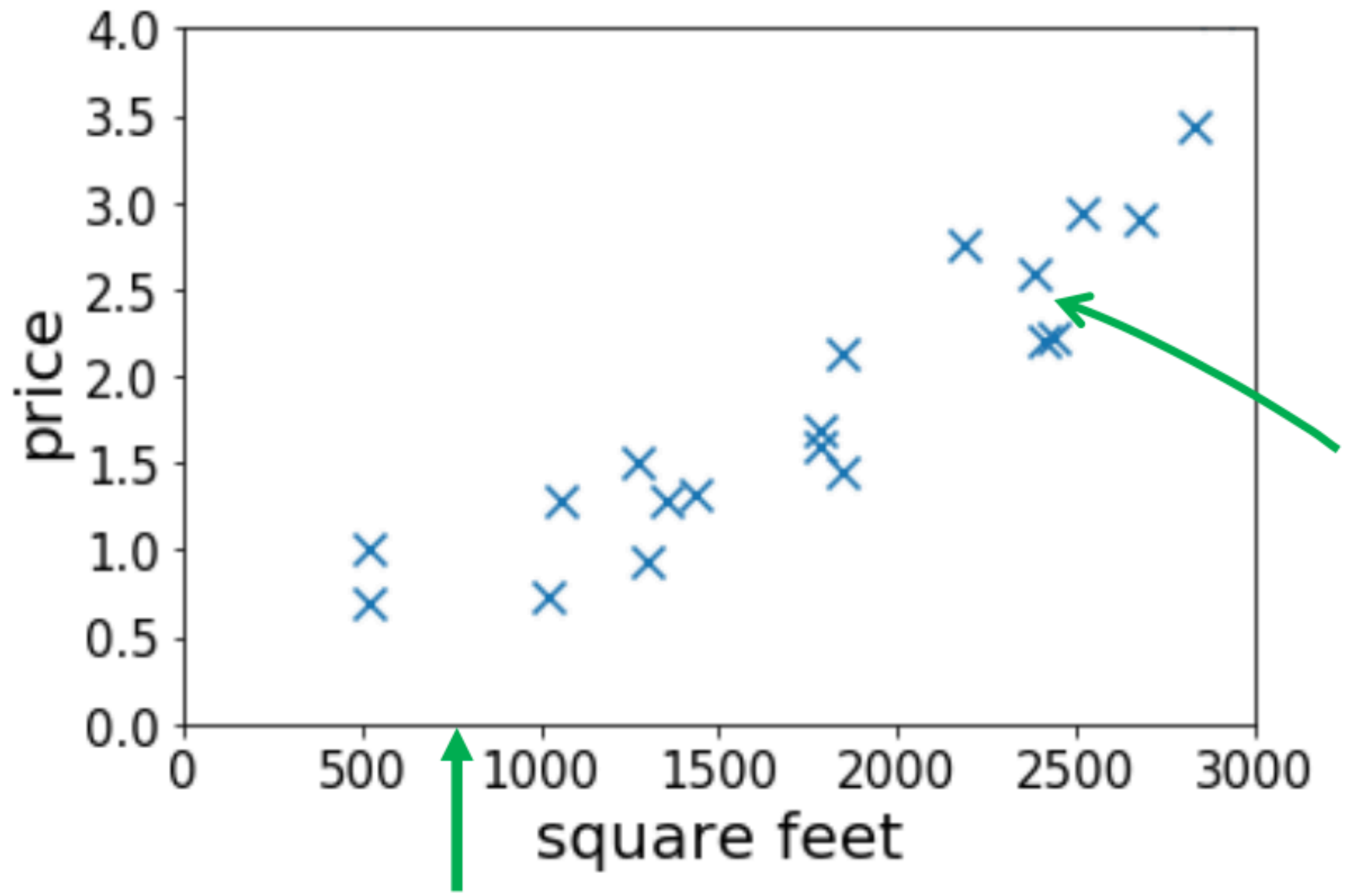
Generalized Linear Models



"Linear" Models

Kernel Methods

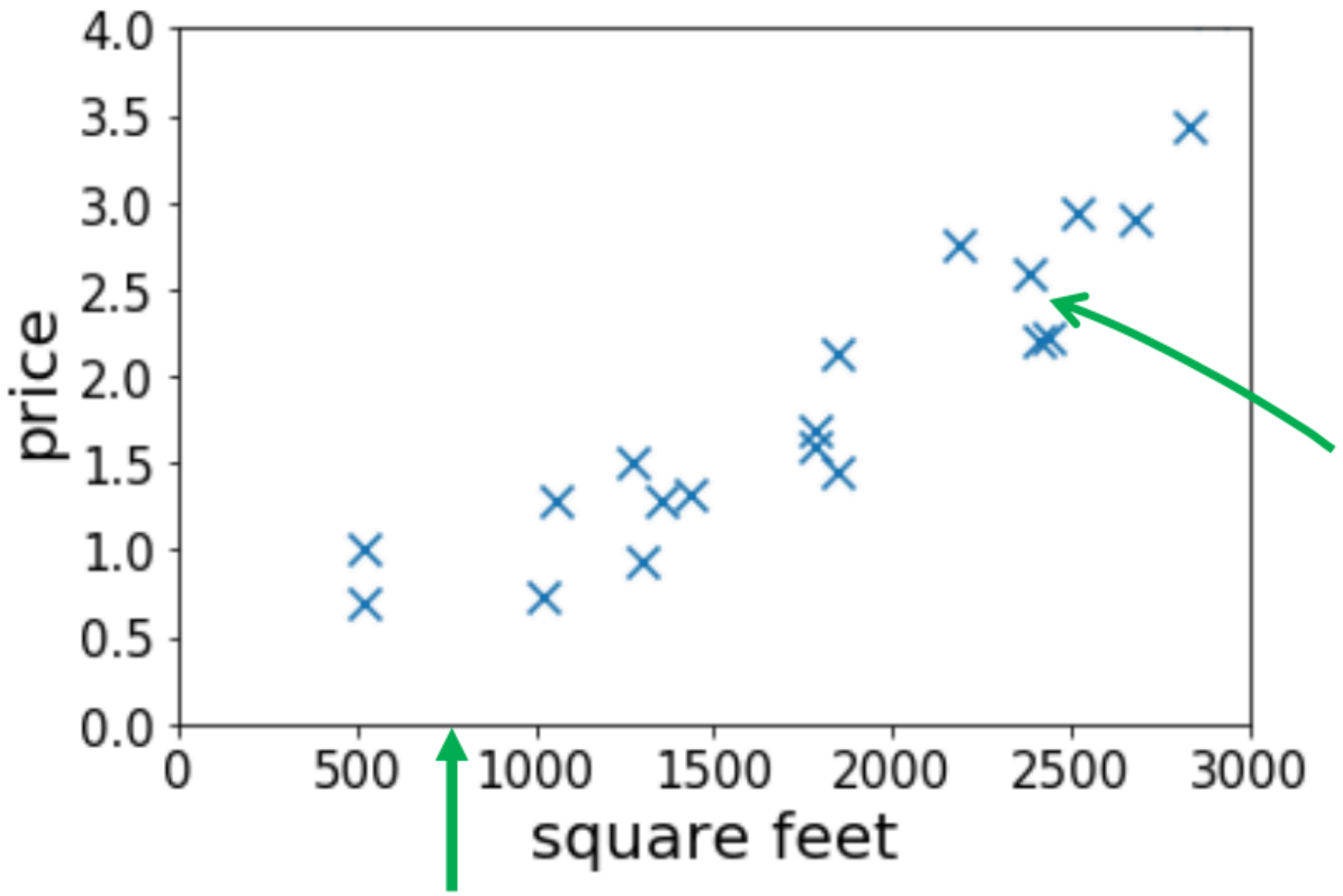
Feature Map



15th sample
$(x^{(15)}, y^{(15)})$

$x = 800$
$y = ?$

Feature Map

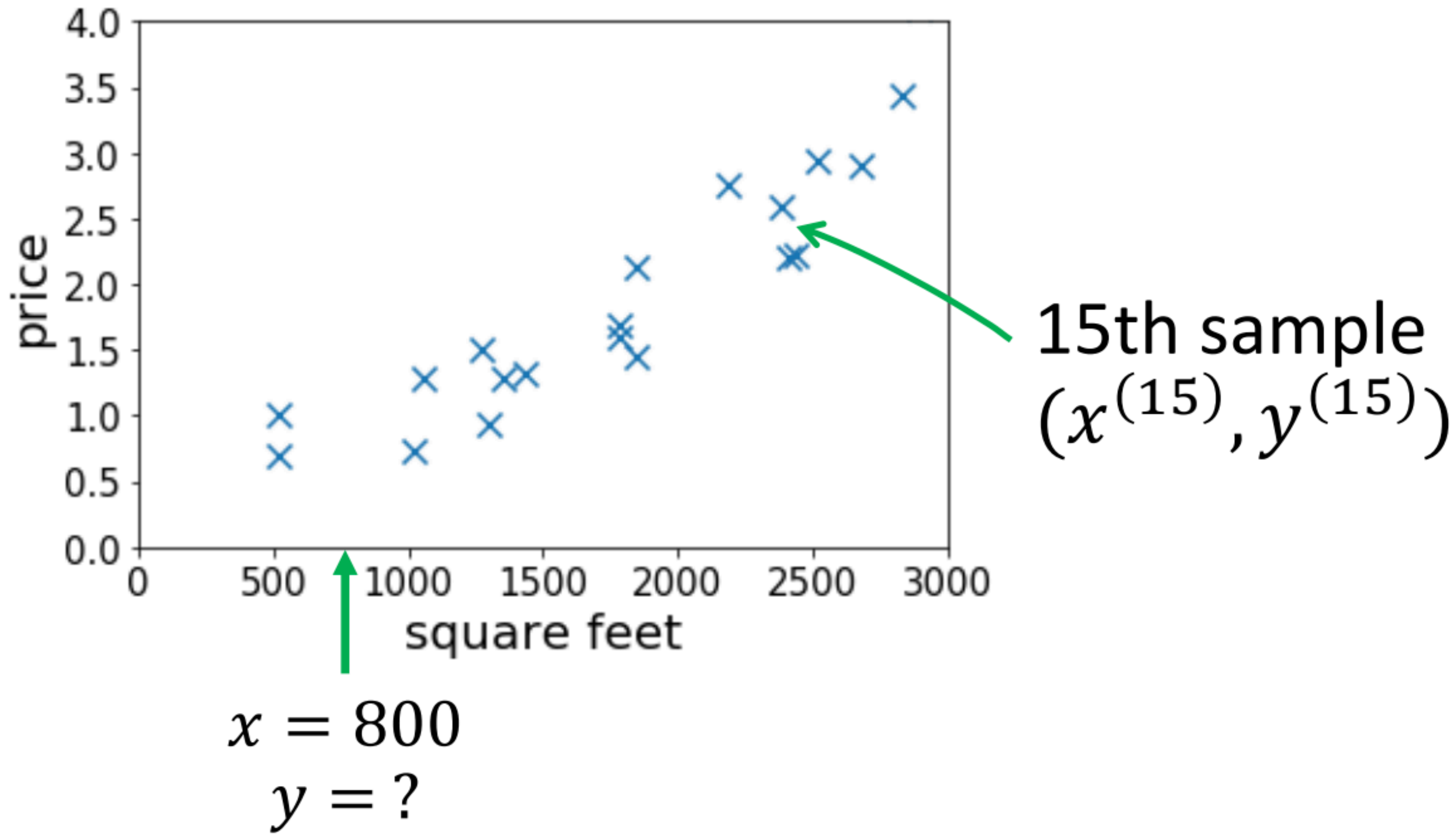


15th sample
$(x^{(15)}, y^{(15)})$

$x = 800$
$y = ?$

$$y = \theta x$$

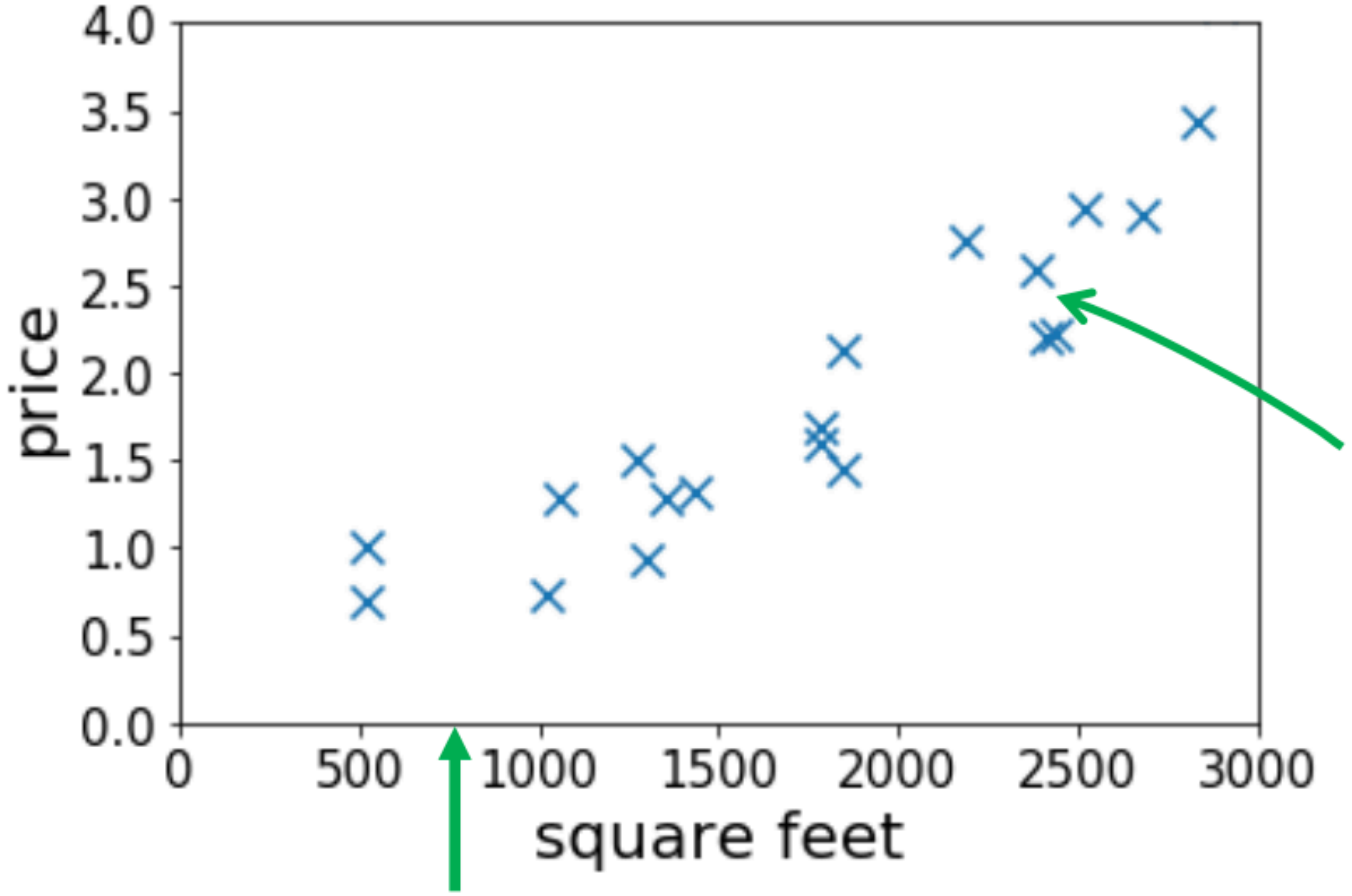
Feature Map



$$y = \theta x$$

$$y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$$

Feature Map



15th sample $(x^{(15)}, y^{(15)})$

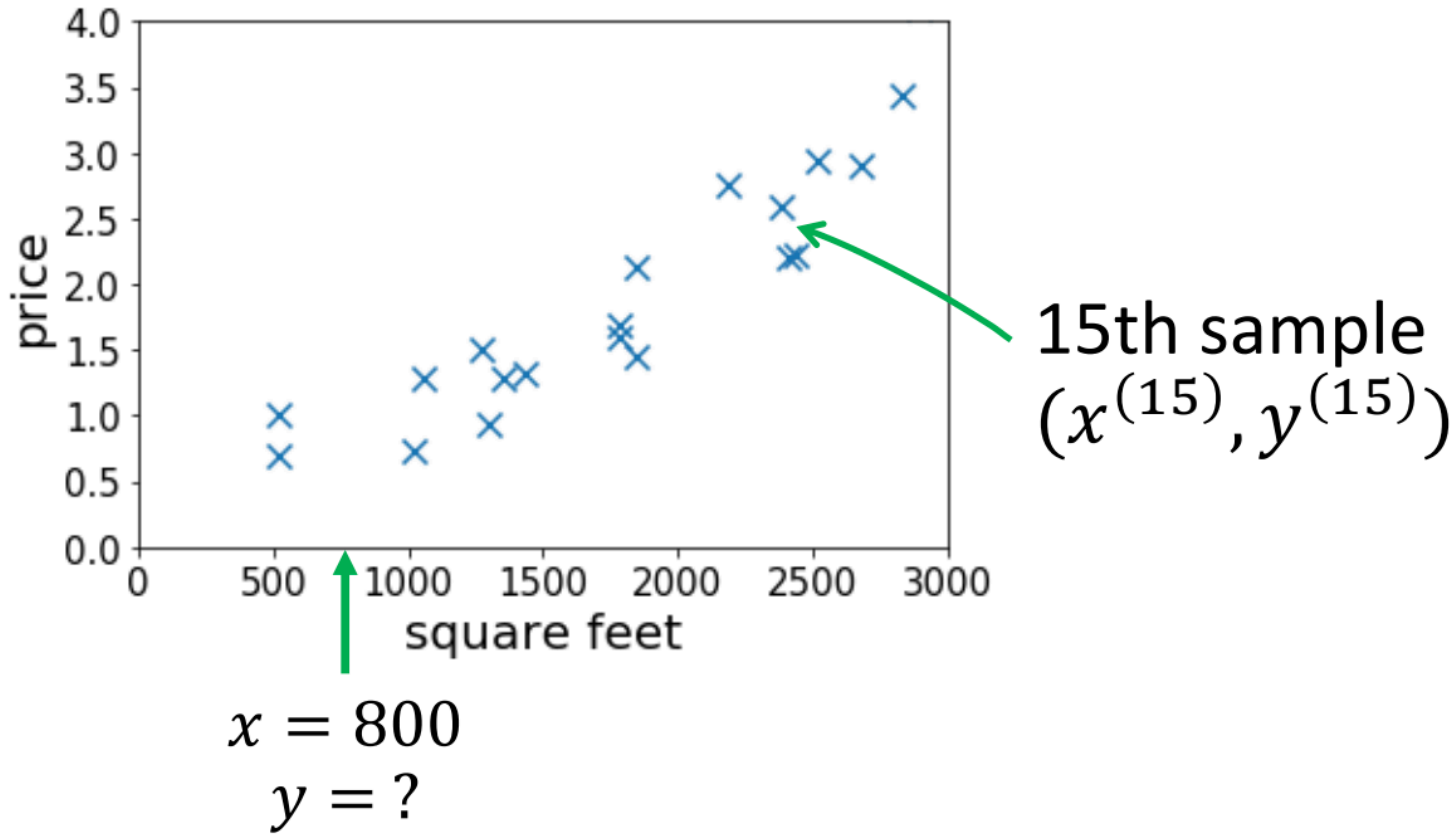
$x = 800$
$y = ?$

$$y = \theta x$$

$$y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$$

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4.$$

Feature Map



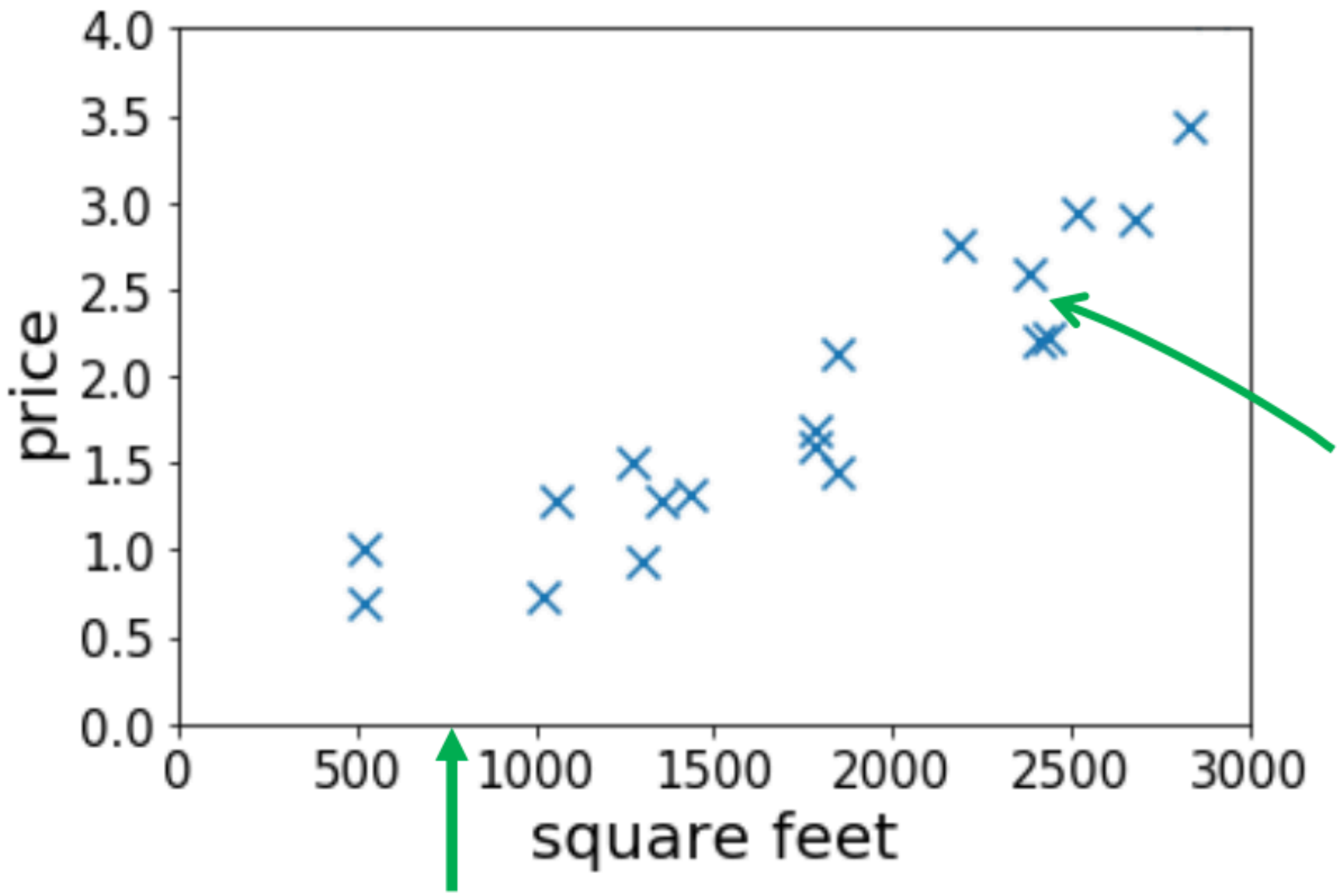
$$y = \theta x$$

$$y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$$

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4.$$

$$y = \theta^T \phi(x)$$

Feature Map



15th sample
$(x^{(15)}, y^{(15)})$

$x = 800$
$y = ?$

$$y = \theta x$$

$$y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + \theta_0$$

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix} \in \mathbb{R}^4.$$

Feature map
$\phi : R^d \rightarrow R^p$

$$y = \theta^T \phi(x)$$

LMS Update Rule with Features

Linear Regression:

$$\begin{aligned}\theta &:= \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x^{(i)} \\ &:= \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)}\right) x^{(i)}.\end{aligned}$$

With Features:

LMS Update Rule with Features

Linear Regression:

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)}) \right) x^{(i)}$$

$$:= \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)} \right) x^{(i)}.$$

With Features:

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)}) \right) \phi(x^{(i)})$$

LMS Update Rule with Features

Linear Regression:

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)})\right) x^{(i)}$$

$$:= \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T x^{(i)}\right) x^{(i)}.$$

With Features:

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

How about Generalized Linear Models with Features?

New Feature Vector Can Be Very High-Dimensional

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_2 x_1 \\ \vdots \\ x_1^3 \\ x_1^2 x_2 \\ \vdots \end{bmatrix}$$

Computationally expensive

New Feature Vector Can Be Very High-Dimensional

$$\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_1^2 \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_2 x_1 \\ \vdots \\ x_1^3 \\ x_1^2 x_2 \\ \vdots \end{bmatrix}$$

Computationally expensive

Is the computation evitable given $\theta \in R^p$?

Kernel Trick

- If θ is initialized as 0, then at any step of the gradient descent:

$$\theta = \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) \qquad \beta_i \in R$$

Kernel Trick

- If θ is initialized as 0, then at any step of the gradient descent:

$$\theta = \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) \qquad \beta_i \in R$$

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$= \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$= \sum_{i=1}^{n} \underbrace{\left(\beta_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)\right)}_{\text{new } \beta_i} \phi(x^{(i)})$$

Kernel Trick

- If θ is initialized as 0, then at any step of the gradient descent:

$$\theta = \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) \qquad \beta_i \in R$$

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$= \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$= \sum_{i=1}^{n} \underbrace{\left(\beta_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)\right)}_{\text{new } \beta_i} \phi(x^{(i)})$$

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)$$

Kernel Trick

- If θ is initialized as 0, then at any step of the gradient descent:

$$\theta = \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) \qquad \beta_i \in R$$

$$\theta := \theta + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$= \sum_{i=1}^{n} \beta_i \phi(x^{(i)}) + \alpha \sum_{i=1}^{n} \left(y^{(i)} - \theta^T \phi(x^{(i)})\right) \phi(x^{(i)})$$

$$= \sum_{i=1}^{n} \underbrace{\left(\beta_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)\right)}_{\text{new } \beta_i} \phi(x^{(i)})$$

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \theta^T \phi(x^{(i)})\right)$$

$$\boxed{\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right)}$$

Kernel Trick

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right)$$

Kernel Trick

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right)$$

Rewrite $\phi(x^{(j)})^T \phi(x^{(i)}) = < \phi(x^{(j)}), \phi(x^{(i)}) >$

Kernel Trick

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right)$$

Rewrite $\phi(x^{(j)})^T \phi(x^{(i)}) = < \phi(x^{(j)}), \phi(x^{(i)}) >$

We can precompute all pairwise $< \phi(x^{(j)}), \phi(x^{(i)}) >$ beforehand, and reuse it for every gradient descent update

Kernel Trick

$$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j \phi(x^{(j)})^T \phi(x^{(i)}) \right)$$

Kernel $K(x, z)$ $\quad \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ $\quad \mathcal{X}$ is the space of the input

$$K(x, z) \triangleq \langle \phi(x), \phi(z) \rangle$$

The Algorithm

The Algorithm

- Compute $K(\phi(x^{(i)}), \phi(x^{(j)})) = <\phi(x^{(i)}), \phi(x^{(j)})>$ for all i, j

The Algorithm

- Compute $K(\phi(x^{(i)}), \phi(x^{(j)})) = < \phi(x^{(i)}), \phi(x^{(j)}) >$ for all i, j
- Loop $$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j K(x^{(i)}, x^{(j)}) \right) \quad \forall i \in \{1, \ldots, n\}$$

The Algorithm

- Compute $K(\phi(x^{(i)}), \phi(x^{(j)})) = < \phi(x^{(i)}), \phi(x^{(j)}) >$ for all i, j
- Loop $$\beta_i := \beta_i + \alpha\left(y^{(i)} - \sum_{j=1}^{n} \beta_j K(x^{(i)}, x^{(j)})\right) \quad \forall i \in \{1, \ldots, n\}$$

Recall that n is the number of data samples

The Algorithm

- Compute $K(\phi(x^{(i)}), \phi(x^{(j)})) = < \phi(x^{(i)}), \phi(x^{(j)}) >$ for all i, j
- Loop $$\beta_i := \beta_i + \alpha \left(y^{(i)} - \sum_{j=1}^{n} \beta_j K(x^{(i)}, x^{(j)}) \right) \quad \forall i \in \{1, \ldots, n\}$$

Recall that n is the number of data samples

Or in vector notation, letting K be the $n \times n$ matrix with $K_{ij} = K(x^{(i)}, x^{(j)})$, we have

$$\beta := \beta + \alpha(\vec{y} - K\beta)$$

Inference

Inference

We do not need to explicitly compute θ !

Inference

We do not need to explicitly compute θ !

$$\theta^T \phi(x) = \sum_{i=1}^{n} \beta_i \phi(x^{(i)})^T \phi(x) = \sum_{i=1}^{n} \beta_i K(x^{(i)}, x)$$

Inference

We do not need to explicitly compute θ !

$$\theta^T \phi(x) = \sum_{i=1}^{n} \beta_i \phi(x^{(i)})^T \phi(x) = \sum_{i=1}^{n} \beta_i K(x^{(i)}, x)$$

The Kernel function is all we need for training and inference!

Implicit Feature Map

Do we still need to define feature maps?

$$K(x, z) \triangleq \langle \phi(x), \phi(z) \rangle$$

Implicit Feature Map

Do we still need to define feature maps?

$$K(x, z) \triangleq \langle \phi(x), \phi(z) \rangle$$

What kinds of kernel functions K() can correspond to some feature map ϕ

Example

$$K(x, z) = (x^T z)^2 \qquad x, z \in \mathbb{R}^d$$

Example

$$K(x, z) = (x^T z)^2 \qquad x, z \in \mathbb{R}^d$$

What is the feature map to make K a valid kernel function?

Example

$$K(x,z) = (x^T z)^2 \qquad x, z \in \mathbb{R}^d$$

What is the feature map to make K a valid kernel function?

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{d} x_i z_i\right)\left(\sum_{j=1}^{d} x_j z_j\right) \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{d} (x_i x_j)(z_i z_j) \end{aligned}$$

Example

$$K(x, z) = (x^T z)^2 \qquad x, z \in \mathbb{R}^d$$

What is the feature map to make K a valid kernel function?

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{d} x_i z_i\right)\left(\sum_{j=1}^{d} x_j z_j\right) \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{d} (x_i x_j)(z_i z_j) \end{aligned}$$

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

Example

$$K(x,z) = (x^T z)^2 \qquad x, z \in \mathbb{R}^d$$

What is the feature map to make K a valid kernel function?

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{d} x_i z_i\right)\left(\sum_{j=1}^{d} x_j z_j\right) \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{d} (x_i x_j)(z_i z_j) \end{aligned}$$

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

Requires O(d^2) compute for feature mapping

Example

$$K(x, z) = (x^T z)^2 \qquad x, z \in \mathbb{R}^d$$

What is the feature map to make K a valid kernel function?

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{d} x_i z_i\right)\left(\sum_{j=1}^{d} x_j z_j\right) \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{d} (x_i x_j)(z_i z_j) \end{aligned}$$

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

Requires O(d^2) compute for feature mapping

Requires O(d) compute for Kernel function

Next Lecture

- What kinds of functions would make a kernel function?
- Infinite dimensions of feature mapping?
- Support Vector Machines

Thank You!
Q & A