Machine Learning

Linear Regression 2

Kia Nazarpour

Based on Hao Tang's slides

Linear regression

- Given a dataset S, find $\boldsymbol{\theta} = [\mathbf{w}, b]$ by minimising the mean-squared error (MSE)

$$L = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}_i + b - y_i)^2$$

- Centering

$$\dot{\mathbf{y}} = \begin{bmatrix} y_1 - \bar{y} \\ y_2 - \bar{y} \\ \vdots \\ y_N - \bar{y} \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} \dot{\mathbf{x}}_1^\top \\ \dot{\mathbf{x}}_2^\top \\ \vdots \\ \dot{\mathbf{x}}_N^\top \end{bmatrix}$$

- Computing the **Moore-Penrose pseudoinverse**

$$\begin{aligned} \mathbf{w} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \dot{\mathbf{y}} \\ b &= \bar{y} - \mathbf{w}^\top \bar{\mathbf{x}} \end{aligned}$$

Augmenting the feature vector

$$f(\mathbf{x}; \mathbf{w}, b) = \mathbf{w}^\top \mathbf{x} + b = \begin{bmatrix} \mathbf{w}^\top & b \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}^\top \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{w}'^\top \mathbf{x}' = f(\mathbf{x}'; \mathbf{w}')$$

- The 1 can be seen as a feature independent of the input.
- Suppose we have a data point $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^\top$.
- The data point after appending 1 becomes $\mathbf{x}' = \begin{bmatrix} 1 & x_1 & x_2 & x_3 \end{bmatrix}^\top$

Feature Function

- A "linear" regression model is linear in the parameters $\mathbf{w}$, **not** the features.
- A linear regression model can fit an arbitrary nonlinear function of the data, that is $\phi(\mathbf{x})$.
- The data point after appending 1 and quadratic terms becomes

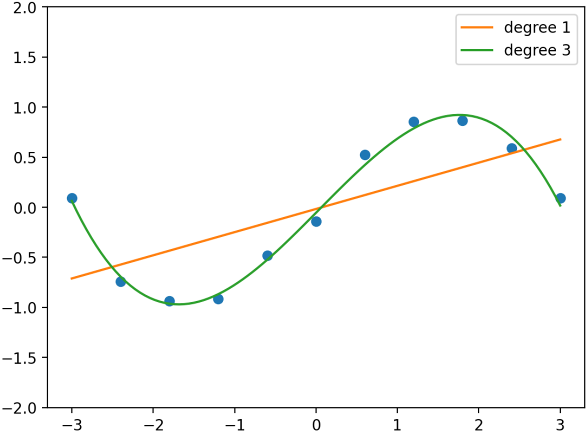
$$\phi(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_1x_2 & x_2x_3 & x_1x_3 & x_1^2 & x_2^2 & x_3^2 \end{bmatrix}^\top$$

- We call ϕ a feature function.

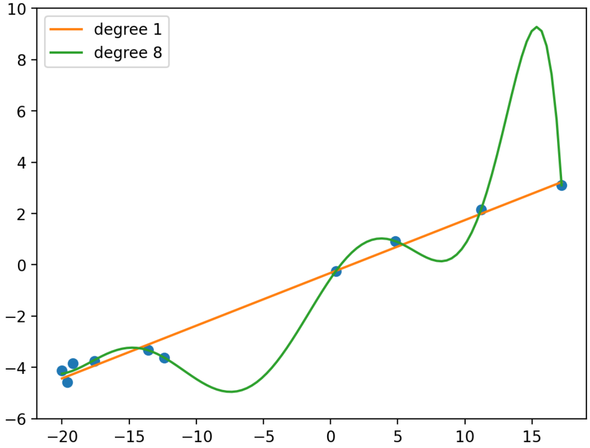
Linear Regression based on Feature Function

- Instead of $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, we now have $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$.
- To solve it, instead of $\mathbf{X}$, we will build $\Phi = \begin{bmatrix} — \phi(\dot{\mathbf{x}}_1) — \\ — \phi(\dot{\mathbf{x}}_2) — \\ \vdots \\ — \phi(\dot{\mathbf{x}}_N) — \end{bmatrix}$
- The optimal solution for linear regression will become $\mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$.

Ploynomial regression



Ploynomial regression



Example

We have a dataset as below

x	y
-3.0	0.0927
-2.4	-0.7417
-1.8	-0.9344
-1.2	-0.9174
-0.6	-0.4811
0.0	-0.1402
$\vdots$	$\vdots$

Instead of writing $y = wx + b$, let's add another dimension that is always 1 and have

$$y = wx + b = \begin{bmatrix} w \\ b \end{bmatrix}^\top \begin{bmatrix} x \\ 1 \end{bmatrix} = \mathbf{w}^\top \mathbf{x}.$$

Example

We therefore have

	x	y
1	-3.0	0.0927
1	-2.4	-0.7417
1	-1.8	-0.9344
1	-1.2	-0.9174
1	-0.6	-0.4811
1	0.0	-0.1402
⋮	⋮	⋮

What happens if we add a dimension of x^2? [note we replace b with w_0]

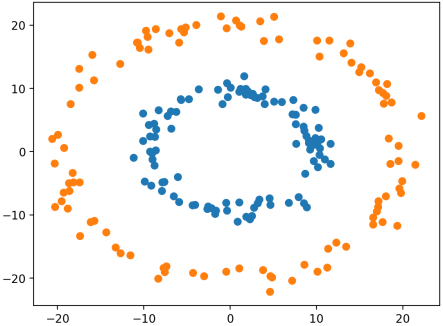
$$y = \begin{bmatrix} w_2 \\ w_1 \\ w_0 \end{bmatrix}^\top \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix} = w_2 x^2 + w_1 x + w_0 = \mathbf{w}^\top \mathbf{x}$$

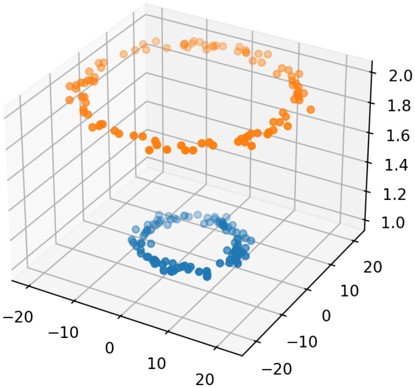
Data

	x	x^2	y
1	-3.0	9.0	0.0927
1	-2.4	5.76	-0.7417
1	-1.8	3.24	-0.9344
1	-1.2	1.44	-0.9174
1	-0.6	0.36	-0.4811
1	0.0	0.0	-0.1402
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- In general, we can add arbitrary high-degree terms.
- If we add degree-2 terms to $\begin{bmatrix}1 & x_1 & x_2\end{bmatrix}$, we get $\begin{bmatrix}1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1x_2\end{bmatrix}$.
- The combination becomes many if we have more dimensions.

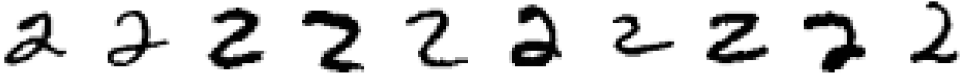
Features





Digit recognition

- Consider digit recognition. How do you describe the digit two?



- A datapoint is a two if it is similar to one of the example twos.
- A feature can be how similar the sample is to one of those examplars.
- If the above examplars are $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{10}$, for a new datapoint $\mathbf{x}$ we can construct a feature vector as follows

$$\begin{bmatrix} 1 & \mathbf{x}^\top \mathbf{x}_1 & \mathbf{x}^\top \mathbf{x}_2 & \ldots & \mathbf{x}^\top \mathbf{x}_{10} \end{bmatrix}$$

From Polynomial regression to Kernel regression

- Feature
  - A feature describes something about the input.
  - The feature vector of $\mathbf{x}$ is written as $\phi(\mathbf{x}$.
  - We do $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$ to make a prediction.
- Kernel
  - A kernel describes similarities of the input to other samples.
  - The similarity of two samples $\mathbf{x}$ and $\mathbf{x}'$ is written as $k(\mathbf{x}, \mathbf{x}')$.
  - We do $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$ to make a prediction.

Kernels and features

- Imagine for some feature function ϕ, we can define a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}').$$

- We can immediately see that k is symmetric, i.e., $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$.
- Ideally, we want ϕ to return an infinite-dimensional vector but we want to avoid computing $\phi(\mathbf{x})^\top \phi(\mathbf{x}')$.
- It would be ideal if this transformation is useful!

Going from features to kernels

- The mean-squared error can be written as

$$L = \|\Phi \mathbf{w} - \mathbf{y}\|_2^2$$

  where

$$\Phi = \begin{bmatrix} — \ \phi(\mathbf{x}_1) \ — \\ — \ \phi(\mathbf{x}_2) \ — \\ \vdots \\ — \ \phi(\mathbf{x}_n) \ — \end{bmatrix}$$

- The optimal solution is $\mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}$.

Going from features to kernels

- To make a prediction,

$$
\begin{aligned}
f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) &= \left((\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y} \right)^\top \phi(\mathbf{x}) \\
&= \mathbf{y}^\top \Phi (\Phi^\top \Phi)^{-1} \, \phi(\mathbf{x}) \\
&= \mathbf{y}^\top (\Phi \Phi^\top)^{-1} \Phi \, \phi(\mathbf{x})
\end{aligned}
$$

Note 1: $\left((A^\top A)^{-1} A^\top \right)^\top = A (A^\top A)^{-1}$
Note 2: $A (A^\top A)^{-1} = (A A^\top)^{-1} A$

Going from features to kernels

$$
\begin{aligned}
f(\mathbf{x}) &= \mathbf{y}^\top (\Phi\Phi^\top)^{-1}\Phi\ \phi(\mathbf{x}) \\
&= \mathbf{y}^\top \begin{bmatrix} \phi(\mathbf{x}_1)^\top\phi(\mathbf{x}_1) & \phi(\mathbf{x}_1)^\top\phi(x_2) & \dots & \phi(\mathbf{x}_1)^\top\phi(\mathbf{x}_n) \\ \phi(\mathbf{x}_2)^\top\phi(\mathbf{x}_1) & \phi(\mathbf{x}_2)^\top\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_2)^\top\phi(\mathbf{x}_n) \\ \vdots & \vdots & & \vdots \\ \phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_1) & \phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_2) & \dots & \phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_n) \end{bmatrix}^{-1} \begin{bmatrix} \phi(\mathbf{x}_1)^\top\phi(\mathbf{x}) \\ \phi(\mathbf{x}_2)^\top\phi(\mathbf{x}) \\ \vdots \\ \phi(\mathbf{x}_n)^\top\phi(\mathbf{x}) \end{bmatrix}
\end{aligned}
$$

Going from features to kernels

$$
\begin{aligned}
f(\mathbf{x}) &= \mathbf{y}^\top (\Phi\Phi^\top)^{-1}\Phi\ \phi(\mathbf{x}) \\
&= \mathbf{y}^\top \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \dots & k(\mathbf{x}_1,\mathbf{x}_n) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \dots & k(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & k(\mathbf{x}_n,\mathbf{x}_2) & \dots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}^{-1} \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}) \\ k(\mathbf{x}_2,x) \\ \vdots \\ k(\mathbf{x}_n,\mathbf{x}) \end{bmatrix} = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i,\mathbf{x})
\end{aligned}
$$

- Linear kernel: $k(\mathbf{x},\mathbf{x}') = \mathbf{x}^\top\mathbf{x}'$
- Polynomial kernel: $k(\mathbf{x},\mathbf{x}') = (r + \mathbf{x}^\top\mathbf{x}')^d$
- Gaussian (RBF) kernel: $k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|_2^2}{2\boldsymbol{\sigma}^2}\right)$

Some implications

- We suddenly can compute infinite-dimensional features. Does that mean we don't need to craft features anymore?
- How do we use kernels for classification?
- Are neural networks kernels?
- The runtime of computing the closed-form solution with kernels is $O(n^3)$.
- The inference time for computing $f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, n)$ is $O(n)$.