

COMP90051

---

# Workshop Week 03

# About the Workshops

---

- 7 sessions in total
  - Tue 12:00-13:00 AH211
  - Tue 12:00-13:00 AH108 \*
  - Tue 13:00-14:00 AH210
  - Tue 16:15-17:15 AH109
  - Tue 17:15-18:15 AH236 \*
  - Tue 18:15-19:15 AH236 \*
  - Fri 14:15-15:15 AH211

# About the Workshops

---

- Homepage

- <https://trevorcohn.github.io/comp90051-2017/workshops>

- Solutions have been released.

# Outline

---

- Review the lecture, background knowledge, etc.
  - Model evaluation, selection, optimization
  - Regularizer as a prior
- Jupyter Usage
- Notebook tasks
  - Task 1: Linear regression
  - Task 2: Polynomial regression

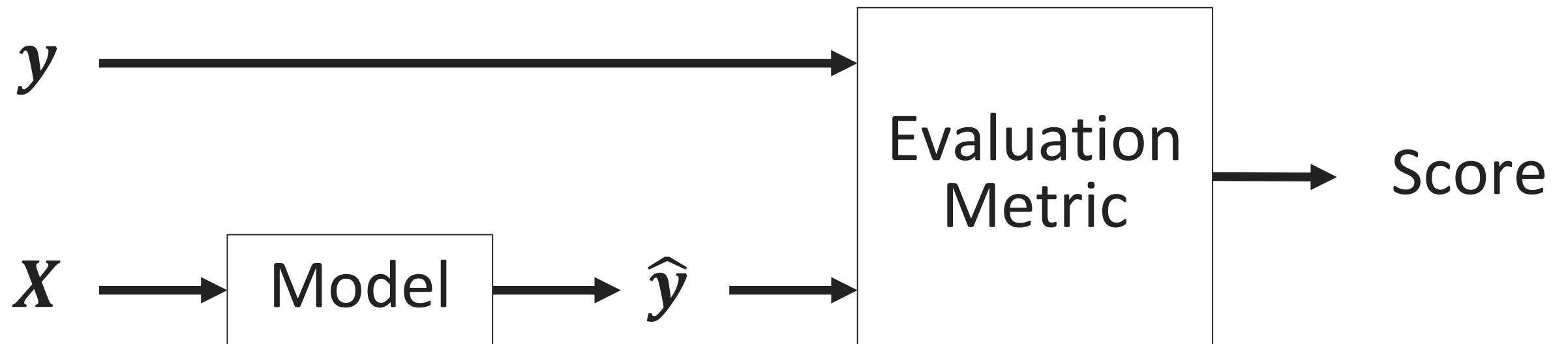
# Outline

---

- Review the lecture, background knowledge, etc.
  - Model evaluation, selection, optimization
  - Regularizer as a prior
- Jupyter Usage
- Notebook tasks
  - Task 1: Linear regression
  - Task 2: Polynomial regression

# Model Evaluation

---



# What could the output be?

---

- ☐ Regression

- ☐ A value

- ☐ A distribution

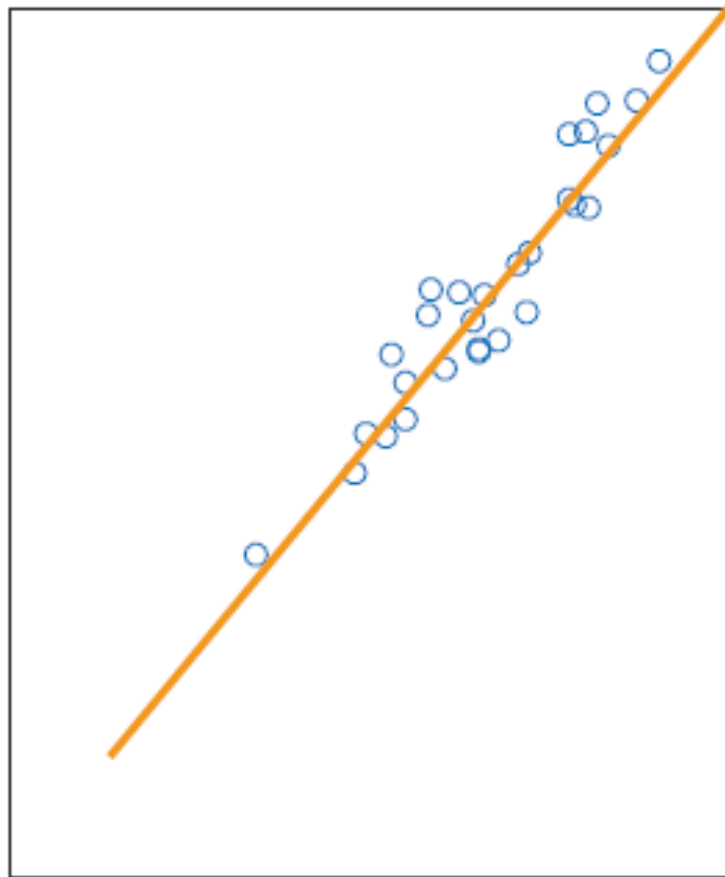
- ☐ Classification

- ☐ A label

- ☐ A value (binary) / values (multi-class)

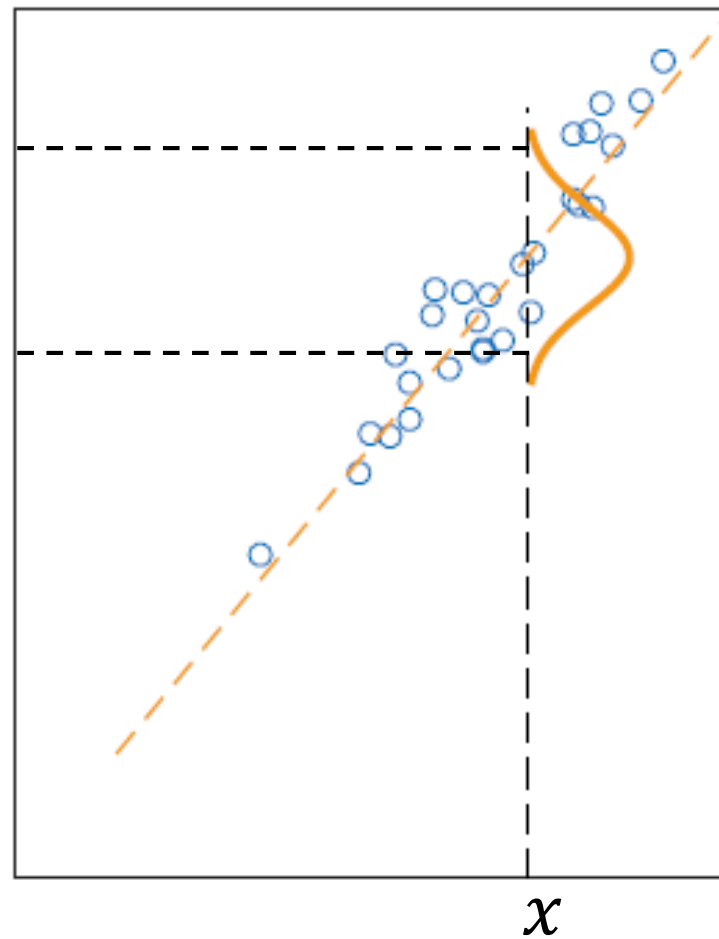
- ☐ A distribution

# Types of models



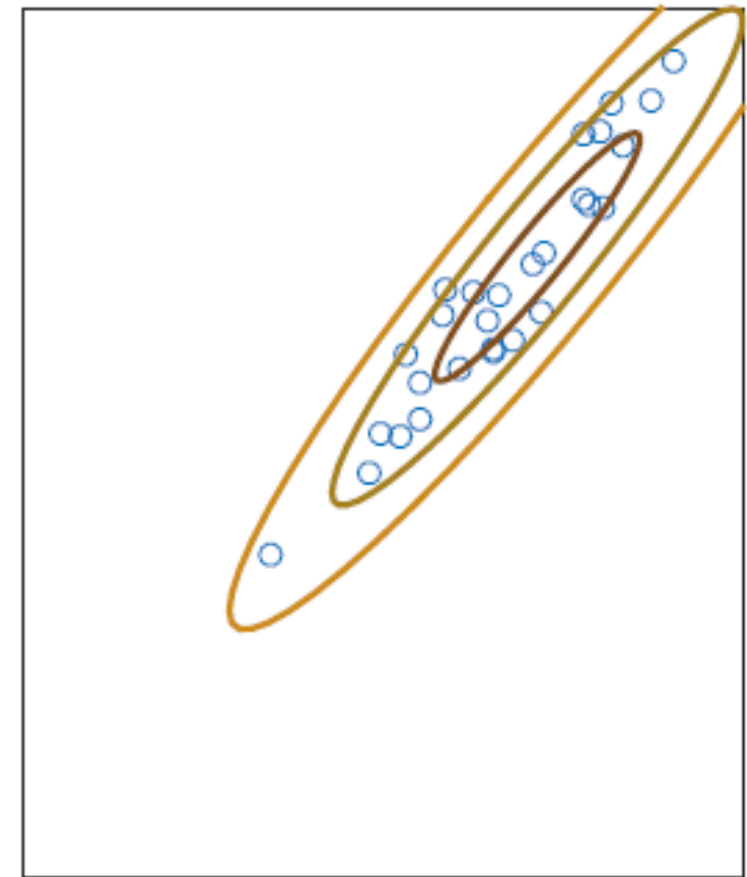
$$\hat{y} = f(x)$$

KT mark was 95, ML  
mark is predicted to be  
95



$$P(y|x)$$

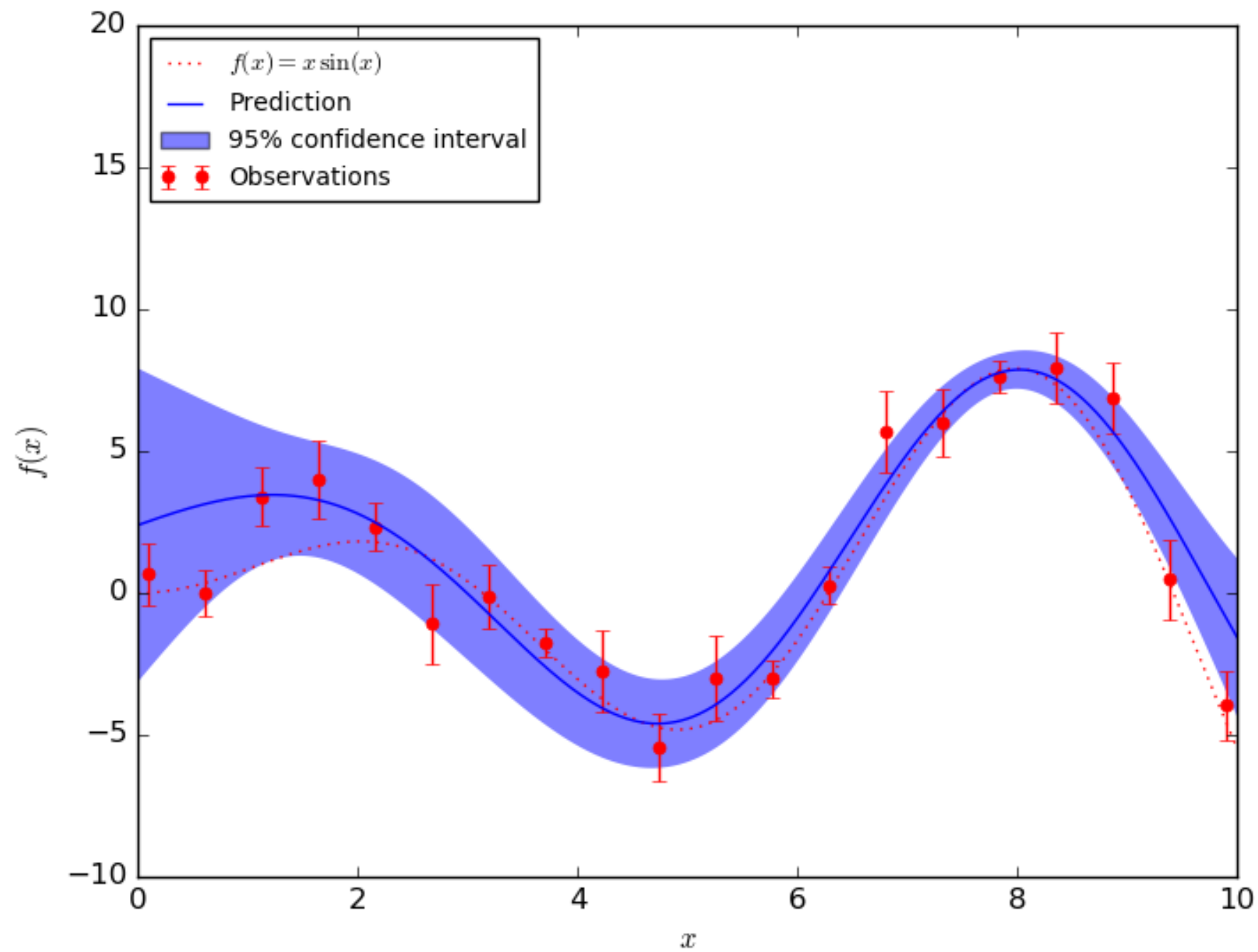
KT mark was 95, ML  
mark is likely to be in  
(92, 97)



$$P(x, y)$$

probability of having  
( $KT = x, ML = y$ )





[http://scikit-learn.org/0.17/auto\\_examples/gaussian\\_process/plot\\_gp\\_regression.html](http://scikit-learn.org/0.17/auto_examples/gaussian_process/plot_gp_regression.html)



**mite**

**container ship**

**motor scooter**

**leopard**

	mite		container ship		motor scooter		leopard
	black widow		lifeboat		go-kart		jaguar
	cockroach		amphibian		moped		cheetah
	tick		fireboat		bumper car		snow leopard
	starfish		drilling platform		golfcart		Egyptian cat

[https://www.tensorflow.org/tutorials/image\\_recognition](https://www.tensorflow.org/tutorials/image_recognition)

# sklearn.linear\_model.LogisticRegression

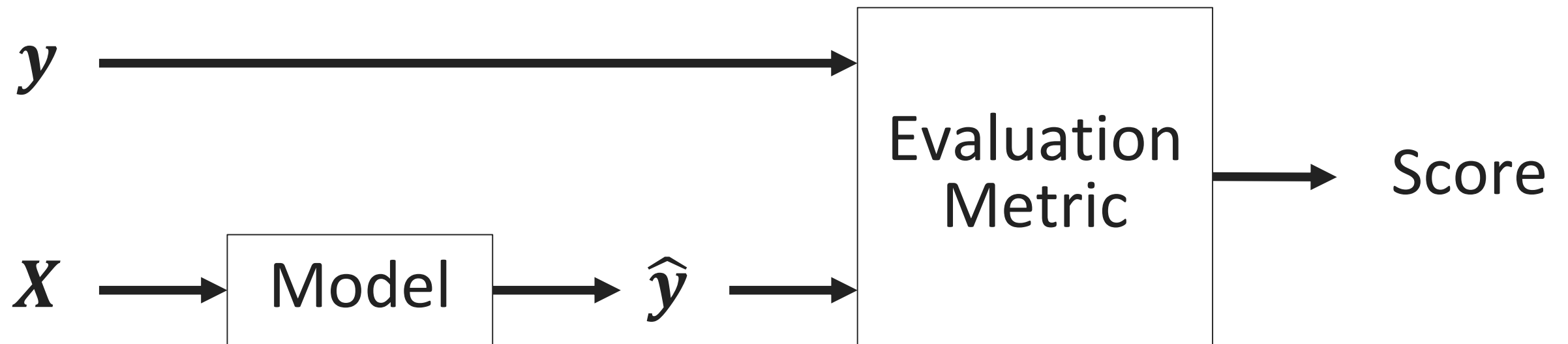
## Methods

<code>decision_function</code> (X)	Predict confidence scores for samples. ←
<code>densify</code> ()	Convert coefficient matrix to dense array format.
<code>fit</code> (X, y[, sample_weight])	Fit the model according to the given training data.
<code>fit_transform</code> (X[, y])	Fit to data, then transform it.
<code>get_params</code> ([deep])	Get parameters for this estimator.
<code>predict</code> (X)	Predict class labels for samples in X. ←
<code>predict_log_proba</code> (X)	Log of probability estimates. ←
<code>predict_proba</code> (X)	Probability estimates. ←
<code>score</code> (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.

[http://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

# Model Evaluation

---



## ☐ Regression

- ☐ RMSE, MAE, etc.

## ☐ Classification

- ☐ Accuracy, precision, recall, f-score, etc.

- ☐ Log-loss (a.k.a. cross entropy), likelihood, etc.

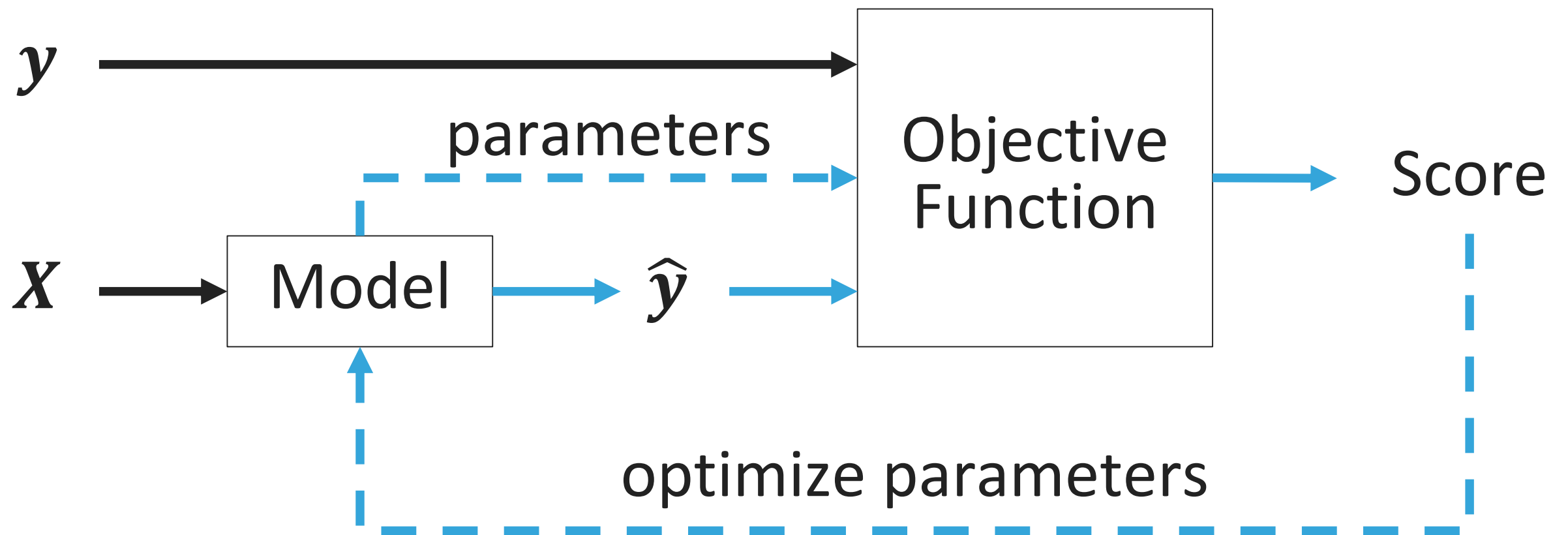
# Model Selection

---



# Model Optimization

---



- ❑ The evaluation metric & the objective function may differ
  - ❑ Could be entirely different
  - ❑ Or additional terms in the objective function, e.g. L1/L2

# More on the objective function

---

- Maximize the likelihood (or log likelihood)

$$\max_{\mathbf{w}} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \Leftrightarrow \max_{\mathbf{w}} \prod p(y_i|\mathbf{x}_i, \mathbf{w}) \Leftrightarrow \max_{\mathbf{w}} \sum \log p(y_i|\mathbf{x}_i, \mathbf{w})$$

- Maximize the posterior (a.k.a. max a posteriori, MAP)

$$\max_{\mathbf{w}} p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \rightarrow \max_{\mathbf{w}} p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) \text{ (assume } \mathbf{w} \perp \mathbf{X})$$

- Minimize the loss function (+regularization)

$$\min_{\mathbf{w}} \sum L(f(\mathbf{x}_i; \mathbf{w}), y_i) \quad \text{or} \quad \min_{\mathbf{w}} \sum L(f(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda R(\mathbf{w})$$

- Minimize the log-loss (a.k.a. cross entropy) (+L1/L2)

- Minimize the hinge-loss (+L2)

- Minimize the mean squared error (+L1/L2)




# More on the objective function

---

- Maximize the likelihood (or log likelihood)

$$\max_{\mathbf{w}} p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \Leftrightarrow \max_{\mathbf{w}} \prod p(y_i|\mathbf{x}_i, \mathbf{w}) \Leftrightarrow \max_{\mathbf{w}} \sum \log p(y_i|\mathbf{x}_i, \mathbf{w})$$


- Maximize the posterior (a.k.a. max a posteriori, MAP)

$$\max_{\mathbf{w}} p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \rightarrow \max_{\mathbf{w}} p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) \text{ (assume } \mathbf{w} \perp \mathbf{X})$$


- Minimize the loss function (+regularization)

$$\min_{\mathbf{w}} \sum L(f(\mathbf{x}_i; \mathbf{w}), y_i) \quad \text{or} \quad \min_{\mathbf{w}} \sum L(f(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda R(\mathbf{w})$$

- Minimize the log-loss (a.k.a. cross entropy) (+L1/L2)

- Minimize the hinge-loss (+L2)

- Minimize the mean squared error (+L1/L2)



# Outline

---

- Review the lecture, background knowledge, etc.
  - Model evaluation, selection, optimization
  - Regularizer as a prior
- Jupyter Usage
- Notebook tasks
  - Task 1: Linear regression
  - Task 2: Polynomial regression

# Regulariser as a prior

- Without regularisation model parameters are found based entirely on the information contained in the training set  $\mathbf{X}$
- Regularisation essentially means introducing additional information
- Recall our probabilistic model  $\mathcal{Y} = \mathbf{x}'\mathbf{w} + \varepsilon$ 
  - \* Here  $\mathcal{Y}$  and  $\varepsilon$  are random variables, where  $\varepsilon$  denotes noise
- Now suppose that  $\mathbf{w}$  is also a random variable (denoted as  $\mathcal{W}$ ) with a normal prior distribution
$$\mathcal{W} \sim \mathcal{N}(0, \lambda^2)$$

# Computing posterior using Bayes rule

- The prior is then used to compute the posterior

A diagram illustrating the Bayes rule equation. The equation is 
$$p(\mathbf{w}|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)}$$
. Four callout bubbles are present: a red bubble labeled 'posterior' pointing to the left side of the equation; a green bubble labeled 'likelihood' pointing to the numerator's first term  $p(\mathbf{y}|X, \mathbf{w})$ ; a purple bubble labeled 'prior' pointing to the numerator's second term  $p(\mathbf{w})$ ; and a blue bubble labeled 'marginal likelihood' pointing to the denominator  $p(\mathbf{y}|X)$ .

- Instead of maximum likelihood (MLE), take **maximum a posteriori** estimate (MAP)
- Apply log trick, so that
 
$$\log(\text{posterior}) = \log(\text{likelihood}) + \log(\text{prior}) - \log(\text{marg})$$
- Arrive at the problem of minimising
 
$$\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$


this term doesn't  
affect optimisation

# Outline

---

- Review the lecture, background knowledge, etc.
  - Model evaluation, selection, optimization
  - Regularizer as a prior
- Jupyter Usage
- Notebook tasks
  - Task 1: Linear regression
  - Task 2: Polynomial regression

# Keyboard Shortcuts

 jupyter 3a\_linear\_regression-answers (autosaved)

File Edit View Insert Cell Kernel Help

          Mark

User Interface Tour

Keyboard Shortcuts

Edit Keyboard Shortcuts

Opens a dialog which shows all keyboard shortcuts

Notebook Help



Markdown



Python



IPython



NumPy



SciPy



## Worksheet 3a: Linear Regression

The aim of this workshop is to get you familiar with using iterative updates (coordinate descent) to fit models in python. For regression we will use linear algebra. Finally we will ensure our plots appear in separate windows.

Firstly we will import the relevant libraries and ensure our plots appear in separate windows.

els in python. For regression we will use linear algebra. Finally we will ensure our plots appear in separate windows.

nsuring our plots appear in separate windows.

# Outline

---

- Review the lecture, background knowledge, etc.
  - Model evaluation, selection, optimization
  - Regularizer as a prior
- Jupyter Usage
- Notebook tasks
  - Task 1: Linear regression
  - Task 2: Polynomial regression

# Linear regression

---

□  $x_1 \rightarrow y_1, x_2 \rightarrow y_2, x_3 \rightarrow y_3, x_4 \rightarrow y_4$

□ 
$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix}$$

□ Minimize the objective function

$$\frac{1}{4} \sum_{i=1}^4 (\hat{y}_i - y_i)^2 \quad \text{or} \quad \frac{1}{4} \sum_{i=1}^4 (\hat{y}_i - y_i)^2 + \lambda \sum_{j=0}^1 w_j^2$$

□ Analytic solution & iterative solution

# Multivariate linear regression (2-D points)

---

$$\square (x_{1,1}, x_{1,2}) \rightarrow y_1, (x_{2,1}, x_{2,2}) \rightarrow y_2$$

$$\square (x_{3,1}, x_{3,2}) \rightarrow y_3, (x_{4,1}, x_{4,2}) \rightarrow y_4$$

$$\square \begin{bmatrix} 1 & x_{1,1} & x_{1,2} \\ 1 & x_{2,1} & x_{2,2} \\ 1 & x_{3,1} & x_{3,2} \\ 1 & x_{4,1} & x_{4,2} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix}$$

$\square$  Minimize the objective function

$$\frac{1}{4} \sum_{i=1}^4 (\hat{y}_i - y_i)^2 \quad \text{or} \quad \frac{1}{4} \sum_{i=1}^4 (\hat{y}_i - y_i)^2 + \lambda \sum_{j=0}^2 w_j^2$$

$\square$  Analytic solution & iterative solution



# Polynomial regression (Quadratic)

---

□  $x_1 \rightarrow y_1, x_2 \rightarrow y_2, x_3 \rightarrow y_3, x_4 \rightarrow y_4$

□ 
$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix}$$

□ Minimize the objective function

$$\frac{1}{4} \sum_{i=1}^4 (\hat{y}_i - y_i)^2 \quad \text{or} \quad \frac{1}{4} \sum_{i=1}^4 (\hat{y}_i - y_i)^2 + \lambda \sum_{j=0}^2 w_j^2$$

□ Analytic solution & iterative solution