

COMP90051

---

# Workshop Week 07

# 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 will be released on next Friday (a week later).

# Syllabus

1	Introduction; Probability theory	Probabilistic models; Parameter fitting	
2	Linear regression; Intro to regularization	Logistic regression; Basis expansion	
3	Optimization; Regularization	Perceptron	
4	Backpropagation	CNNs; Auto-encoders	
5	Hard-margin SVMs	Soft-margin SVMs	←
6	Kernel methods	Ensemble Learning	←
7	Clustering	EM algorithm	
8	Dimensionality reduction; Principal component analysis	Multidimensional scaling; Spectral clustering	
9	Bayesian fundamentals	Bayesian inference with conjugate priors	
10	PGMs, fundamentals	Conditional independence	
11	PGMs, inference	Belief propagation	
12	Statistical inference; Apps	Subject review	

# Outline

---

- Review the lecture, background knowledge, etc.
  - Bagging
    - OOB score
  - SVM
    - Hard-margin & Soft-margin
    - Comparison with logistic regression, perceptron
  - Kernel method
- Run ipython notebooks

# Outline

---

- Review the lecture, background knowledge, etc.

- Bagging

- OOB score

- SVM

- Hard-margin & Soft-margin

- Comparison with logistic regression, perceptron

- Kernel method

- Run ipython notebooks

# Bagging (bootstrap aggregating)

---

- ❑ A model averaging approach
- ❑ Given a standard training set
- ❑ Bootstrap the standard training set
  - ❑ To generate  $m$  new training sets
  - ❑ Bootstrap = sample uniformly and with replacement
- ❑ Train  $m$  base models on the above  $m$  training sets
- ❑ Aggregate the  $m$  base models to make predictions
  - ❑ By voting (classification) or averaging (regression)

---

## ❑ Sample without replacement

Iteration	Choose from	Generate	Sample
1	[1 2 3 4 5 6 7 8 9]	[4]	[4]
2	[1 2 3 5 6 7 8 9]	[1]	[4,1]
3	[2 3 5 6 7 8 9]	[3]	[4,1,3]
4	[2 5 6 7 8 9]	[2]	[4,1,3,2]

## ❑ Sample with replacement

Iteration	Choose from	Generate	Sample
1	[1 2 3 4 5 6 7 8 9]	[4]	[4]
2	[1 2 3 4 5 6 7 8 9]	[1]	[4,1]
3	[1 2 3 4 5 6 7 8 9]	[4]	[4,1,4]
4	[1 2 3 4 5 6 7 8 9]	[1]	[4,1,4,1]



# Original dataset: $\{(x_i, y_i)\}, i = 1, 2, \dots, 9$

---

Model	Training set
m1	[6 3 6 2 7 4 8 3 4]
m2	[4 7 1 5 4 5 1 1 4]
m3	[7 2 8 4 4 3 7 1 1]
m4	[8 2 9 3 9 3 2 5 9]

□ To make a prediction for  $x$

$$\hat{y} = \frac{1}{4} \{m1.predict(x) + m2.predict(x) \\ + m3.predict(x) + m4.predict(x)\}$$

# OOB: out-of-bag

Model	Training set	OOB
m1	[6 3 6 2 7 4 8 3 4]	[1 5 9]
m2	[4 7 1 5 4 5 1 1 4]	[2 3 6 8 9]
m3	[7 2 8 4 4 3 7 1 1]	[5 6 9]
m4	[8 2 9 3 9 3 2 5 9]	[1 4 6 7]

□ To make a prediction for  $x$

$$\hat{y} = \frac{1}{4} \{m1.predict(x) + m2.predict(x) \\ + m3.predict(x) + m4.predict(x)\}$$

$\tilde{y}_i$ : aggregation of models haven't seen  $x_i$

Model	Training set	OOB
m1	[6 3 6 2 7 4 8 3 4]	[1 5 9]
m2	[4 7 1 5 4 5 1 1 4]	[2 3 6 8 9]
m3	[7 2 8 4 4 3 7 1 1]	[5 6 9]
m4	[8 2 9 3 9 3 2 5 9]	[1 4 6 7]

$$\tilde{y}_1 = \frac{1}{2} \{m2.predict(x_1) + m3.predict(x_1)\}$$

$$\tilde{y}_2 = \frac{1}{3} \{m1.predict(x_2) + m3.predict(x_2) + m4.predict(x_2)\}$$

$$\tilde{y}_3 = \frac{1}{3} \{m1.predict(x_3) + m3.predict(x_3) + m4.predict(x_3)\}$$

$\vdots$

$$\tilde{y}_9 = m4.predict(x_9)$$

# OOB score (regression)

---

- OOB mean square error

$$MSE_{OOB} = \frac{1}{|D_{train}|} \sum_{(x_i, y_i) \in D_{train}} (y_i - \tilde{y}_i)^2$$

- Mean square error on a validation set

$$MSE_{val} = \frac{1}{|D_{val}|} \sum_{(x_i, y_i) \in D_{val}} (y_i - \hat{y}_i)^2$$

- OOB score is an alternative to the score on validation set

# Bagging

---

- ❑ Can be used with any type of method
- ❑ Usually applied to decision tree method
- ❑ In sklearn:

```
>>> from sklearn.ensemble import BaggingClassifier
>>> from sklearn.neighbors import KNeighborsClassifier
>>> bagging = BaggingClassifier(KNeighborsClassifier(),
...                             max_samples=0.5, max_features=0.5)
```

# Outline

---

- Review the lecture, background knowledge, etc.
  - Bagging
    - OOB score
  - SVM
    - Hard-margin & Soft-margin
    - Comparison with logistic regression, perceptron
  - Kernel method
- Run ipython notebooks

# SVM

---

## □ Hard-margin

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

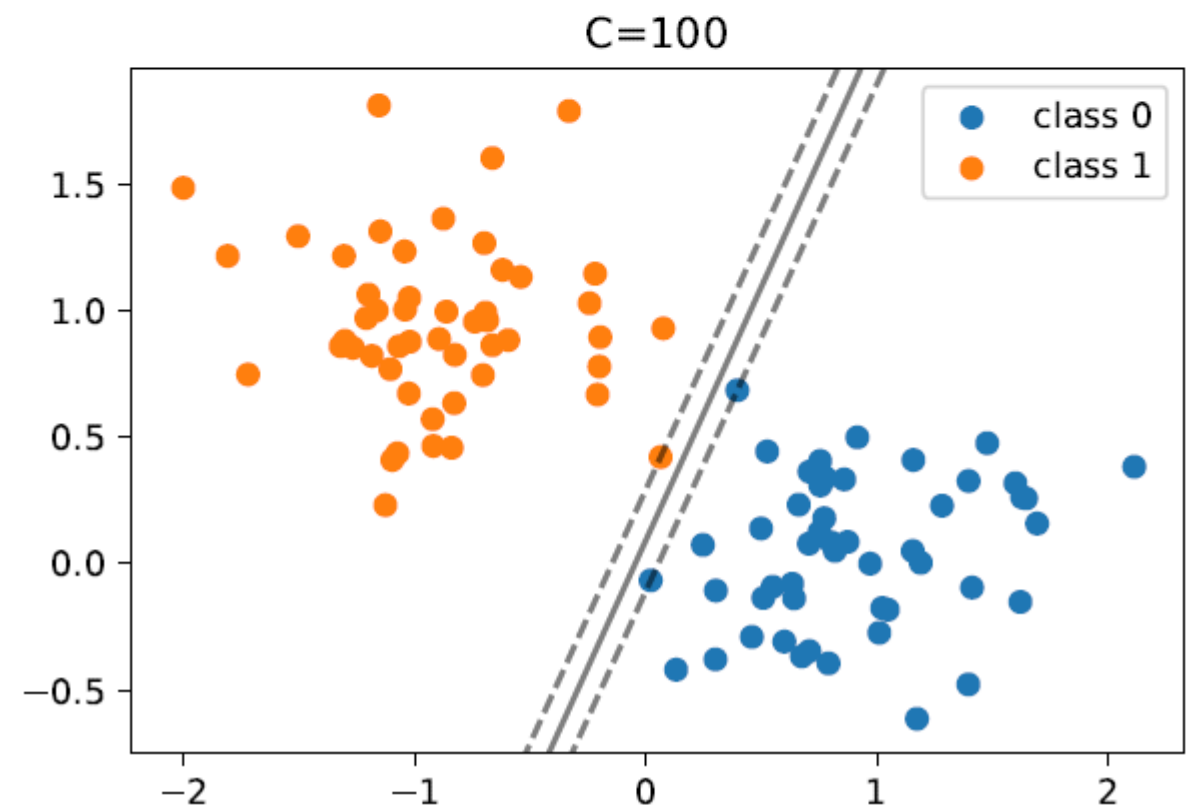
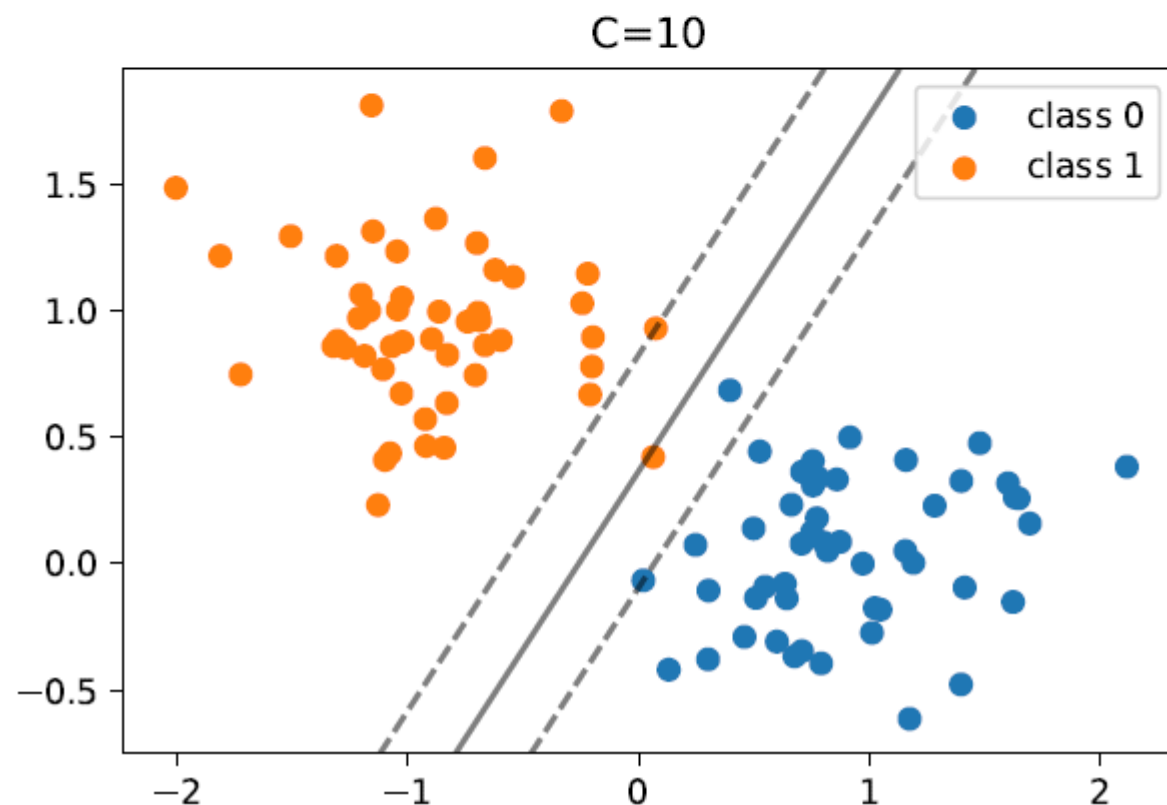
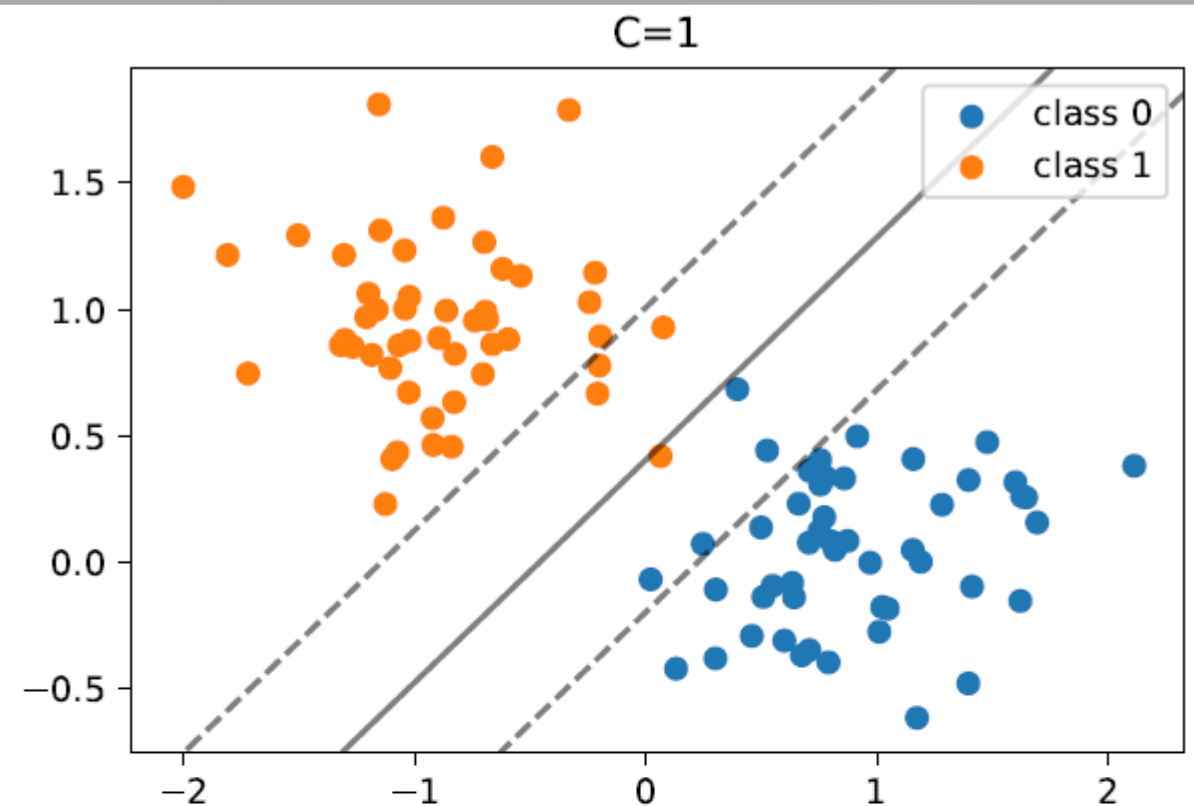
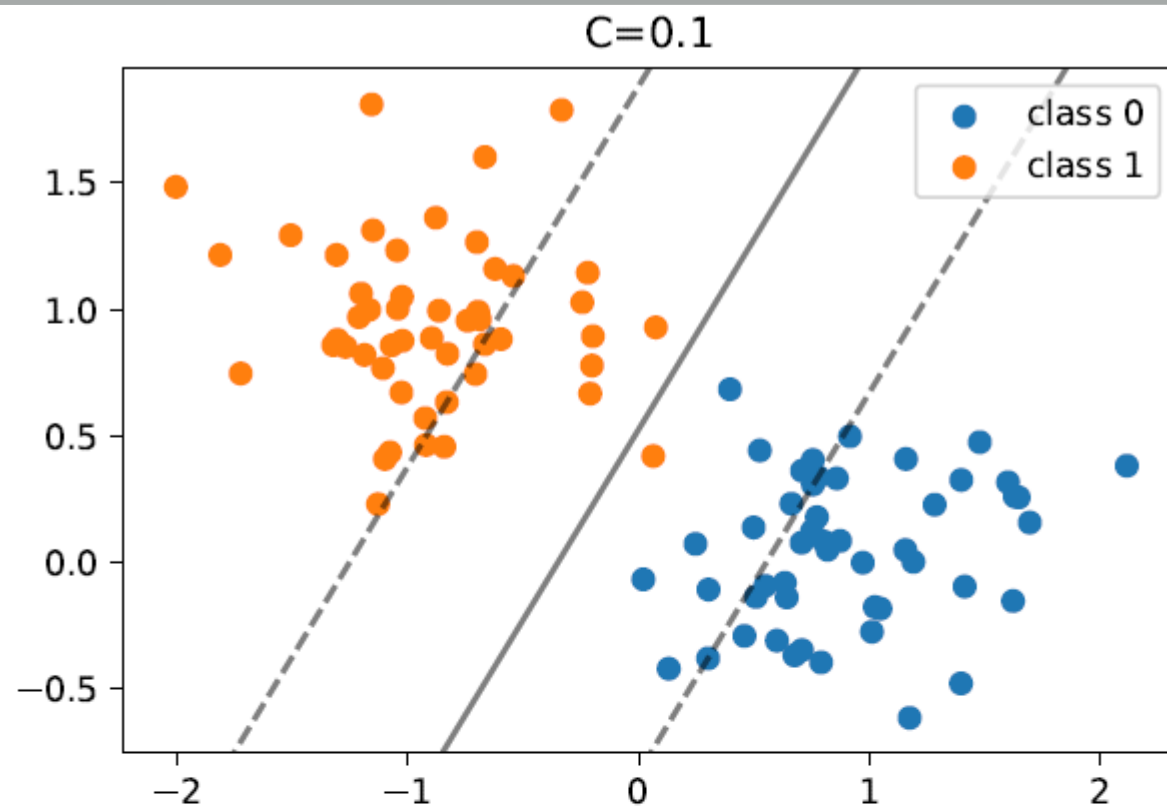
$$s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

## □ Soft-margin

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=0}^n \max(0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b))$$

## □ Hard-margin $\Leftrightarrow C \rightarrow +\infty$

# SVM - different C values





# SVM and Perceptron

---

$$s_i = \mathbf{w}^T \mathbf{x}_i + b$$

□ Hinge loss

$$L(\mathbf{x}_i, y_i) = \max(0, 1 - y_i s_i)$$

□ Perceptron loss

$$L(\mathbf{x}_i, y_i) = \max(0, -y_i s_i)$$

# Logistic regression (binary classification)

---

$$s_i = \mathbf{w}^T \mathbf{x}_i + b$$

□ Log-loss when  $y \in \{0, 1\}$

$$\hat{y}_i = p(y = 1 | \mathbf{x} = \mathbf{x}_i) = \frac{1}{1 + e^{-s_i}}$$

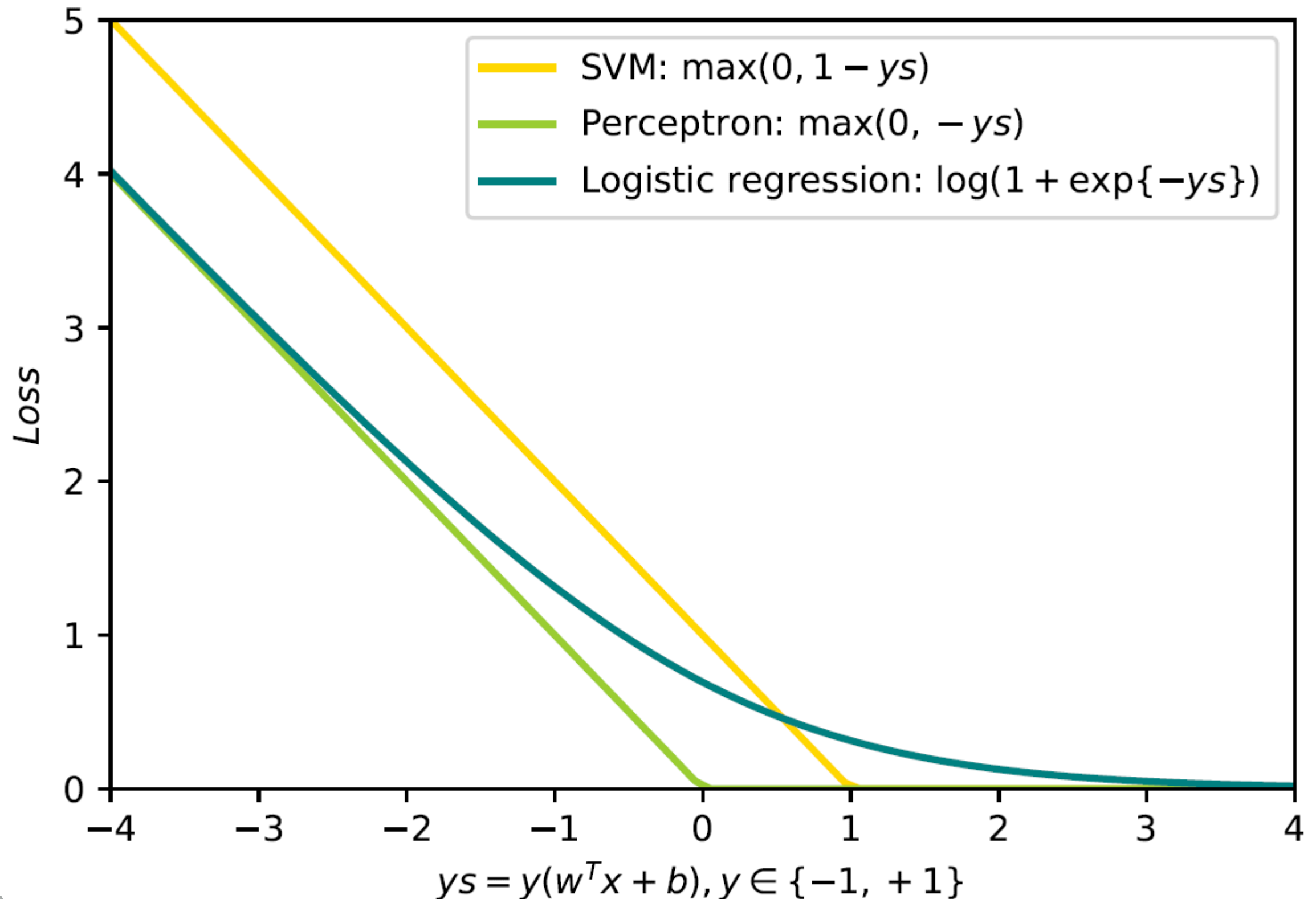
$$L(\mathbf{x}_i, y_i) = -(1 - y_i) \log(1 - \hat{y}_i) - y_i \log \hat{y}_i$$

□ Log-loss when  $y \in \{-1, +1\}$

$$p(y | \mathbf{x} = \mathbf{x}_i) = \frac{1}{1 + e^{-y s_i}}$$

$$L(\mathbf{x}_i, y_i) = -\log p(y = y_i | \mathbf{x} = \mathbf{x}_i) = \log(1 + e^{-y_i s_i})$$

# Loss function for an example $(x, y)$



# Outline

---

- Review the lecture, background knowledge, etc.
  - Bagging
    - OOB score
  - SVM
    - Hard-margin & Soft-margin
    - Comparison with logistic regression, perceptron
  - Kernel method
- Run ipython notebooks

# Kernel method

---

- A kernel function is
  - a similarity function over pairs of raw data points
  - the dot product of a pair of transformed data points

$$K(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u}) \cdot \phi(\mathbf{v})$$

- Could be used for many models:
  - SVM, perceptron, logistic regression, linear regression, etc.
- Kernel SVM is the best known one

# Kernel method

---

- Prove a kernel  $K(\mathbf{u}, \mathbf{v})$  is valid by finding its  $\phi$  function
- $K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^2$  is a valid kernel for 2-d points
- Because
- $K(\mathbf{u}, \mathbf{v}) = (u_1 v_1 + u_2 v_2 + 1)^2 = u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 u_2 v_2 + 2u_1 v_1 + 2u_2 v_2 + 1$
- Let  $\phi(\mathbf{x}) = [x_1^2 \quad x_2^2 \quad \sqrt{2}x_1 x_2 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad 1]$
- Then  $K(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u}) \cdot \phi(\mathbf{v})$

# Outline

---

- Review the lecture, background knowledge, etc.
  - Bagging
    - OOB score
  - SVM
    - Hard-margin & Soft-margin
    - Comparison with logistic regression, perceptron
  - Kernel method
- Run ipython notebooks