NLP Programming Tutorial 9 - Advanced Discriminative Learning

Graham Neubig
Nara Institute of Science and Technology (NAIST)

Review: Classifiers and the Perceptron

Prediction Problems

Given x, predict y

Example we will use:

- Given an introductory sentence from Wikipedia
- Predict whether the article is about a person

Given		Predict
Gonso was a Sanron sect priest (754-827) in the late Nara and early Heian periods.	→	Yes!
Shichikuzan Chigogataki Fudomyoo is a historical site located at Magura, Maizuru City, Kyoto Prefecture.	→	No!

- This is binary classification

Mathematical Formulation

$$\begin{aligned} y &= \mathrm{sign}\left(\mathbf{w}\cdot\boldsymbol{\phi}(x)\right) \\ &= \mathrm{sign}\left(\sum_{i=1}^{I} w_i \cdot \phi_i(x)\right) \end{aligned}$$

- x: the input
- **φ(x)**: vector of feature functions $\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\}$
- **w**: the weight vector $\{w_1, w_2, \ldots, w_I\}$
- y: the prediction, +1 if “yes”, -1 if “no”
  - (sign(v) is +1 if v >= 0, -1 otherwise)

Online Learning

```
create map w
for I iterations
    for each labeled pair x, y in the data
        phi = CREATE_FEATURES(x)
        y' = PREDICT_ONE(w, phi)
        if y' != y
            UPDATE_WEIGHTS(w, phi, y)
```

- In other words
  - Try to classify each training example
  - Every time we make a mistake, update the weights
- Many different online learning algorithms
  - The most simple is the perceptron

Perceptron Weight Update

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y\,\boldsymbol{\phi}(x)$$

- In other words:
  - If y=1, increase the weights for features in **φ**(x)
    - Features for positive examples get a higher weight
  - If y=-1, decrease the weights for features in **φ**(x)
    - Features for negative examples get a lower weight

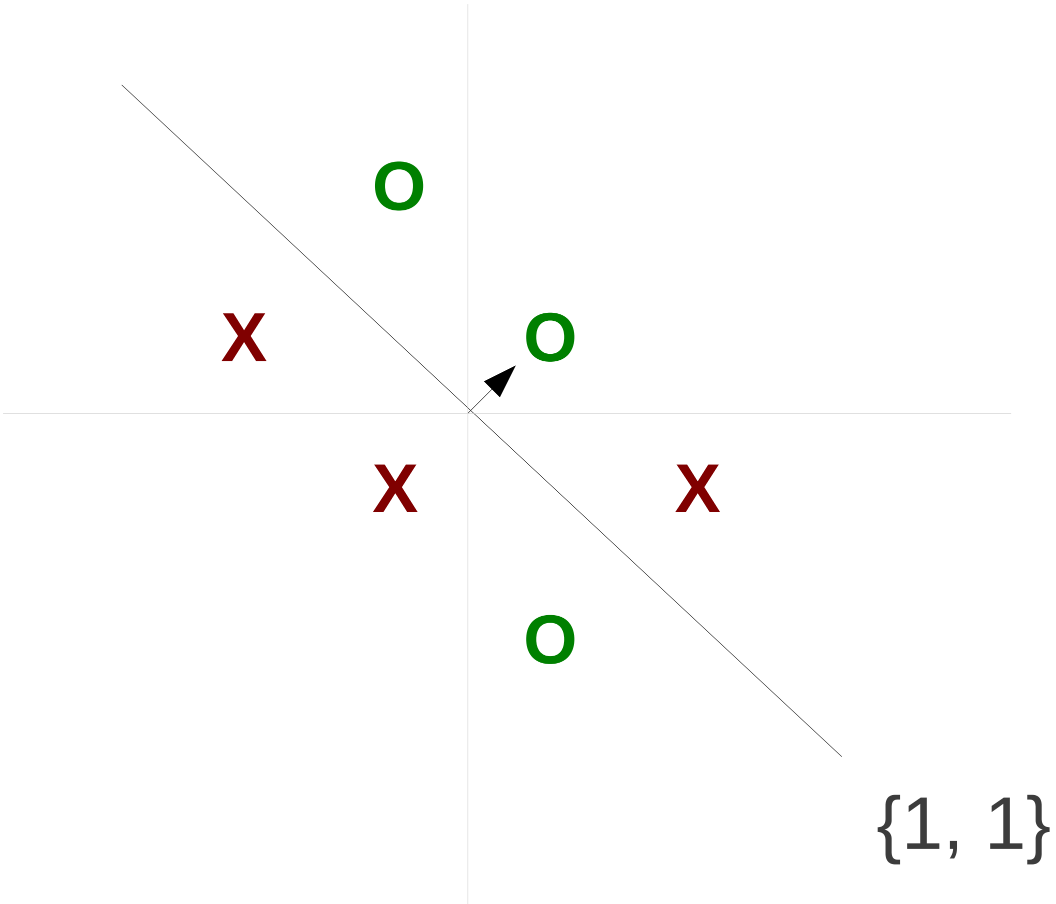
→ Every time we update, our predictions get better!

```
UPDATE_WEIGHTS(w, phi, y)
  for name, value in phi:
    w[name] += value * y
```

Averaged Perceptron

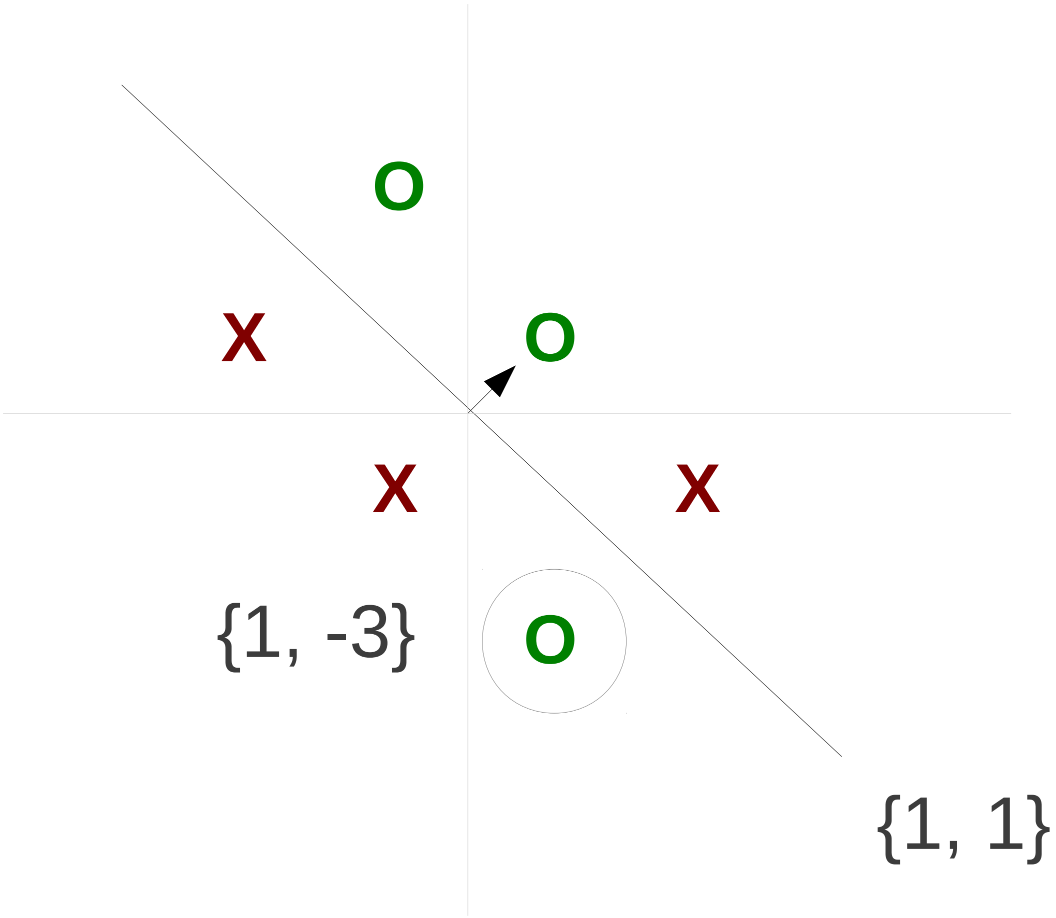
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



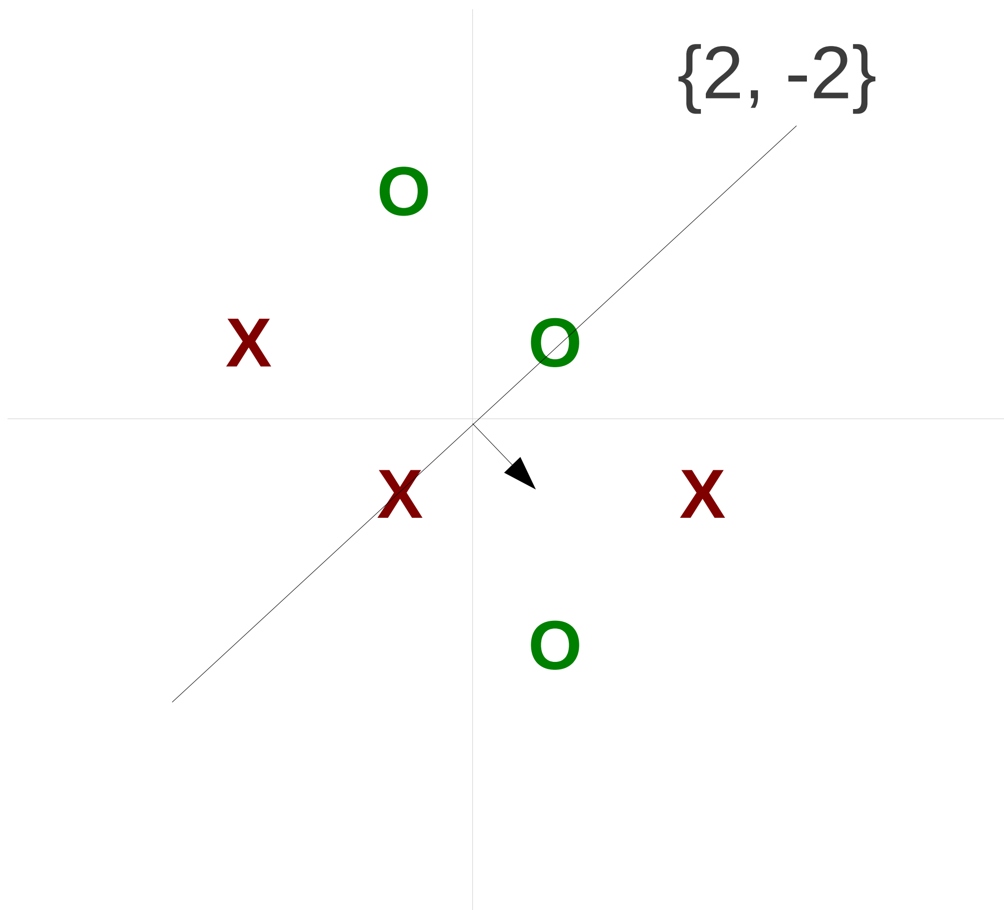
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



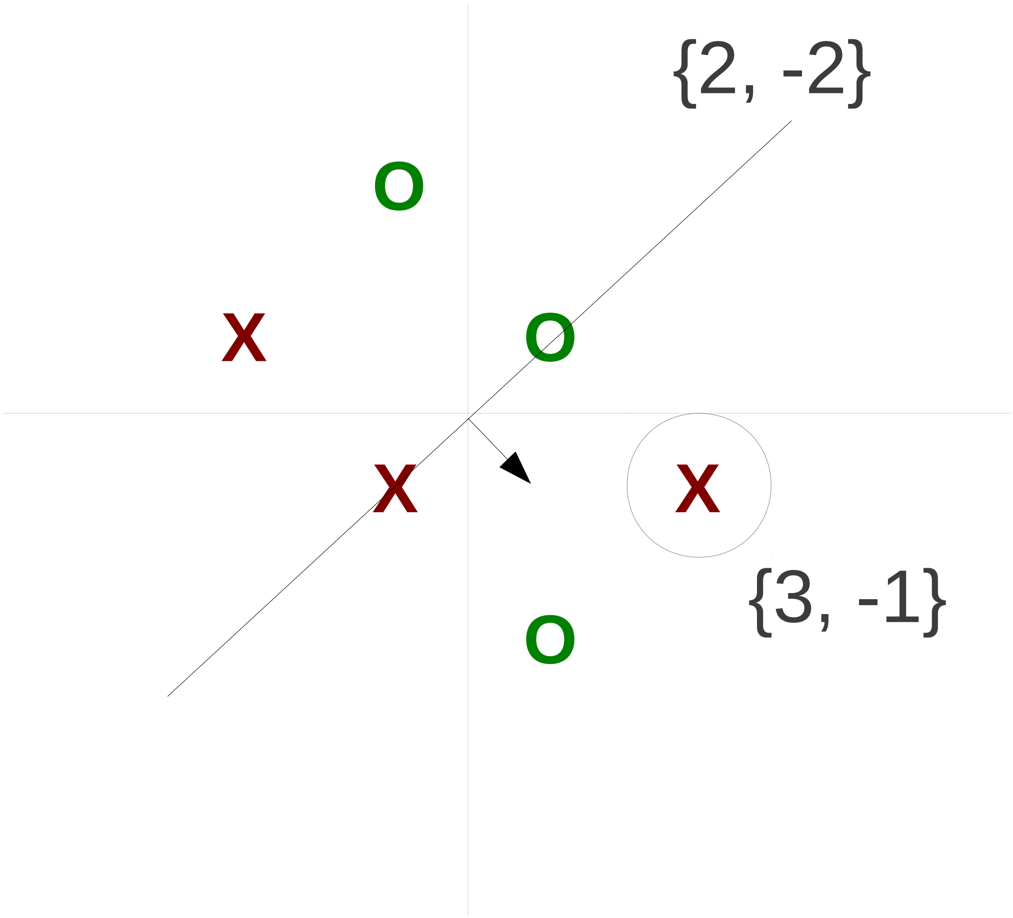
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



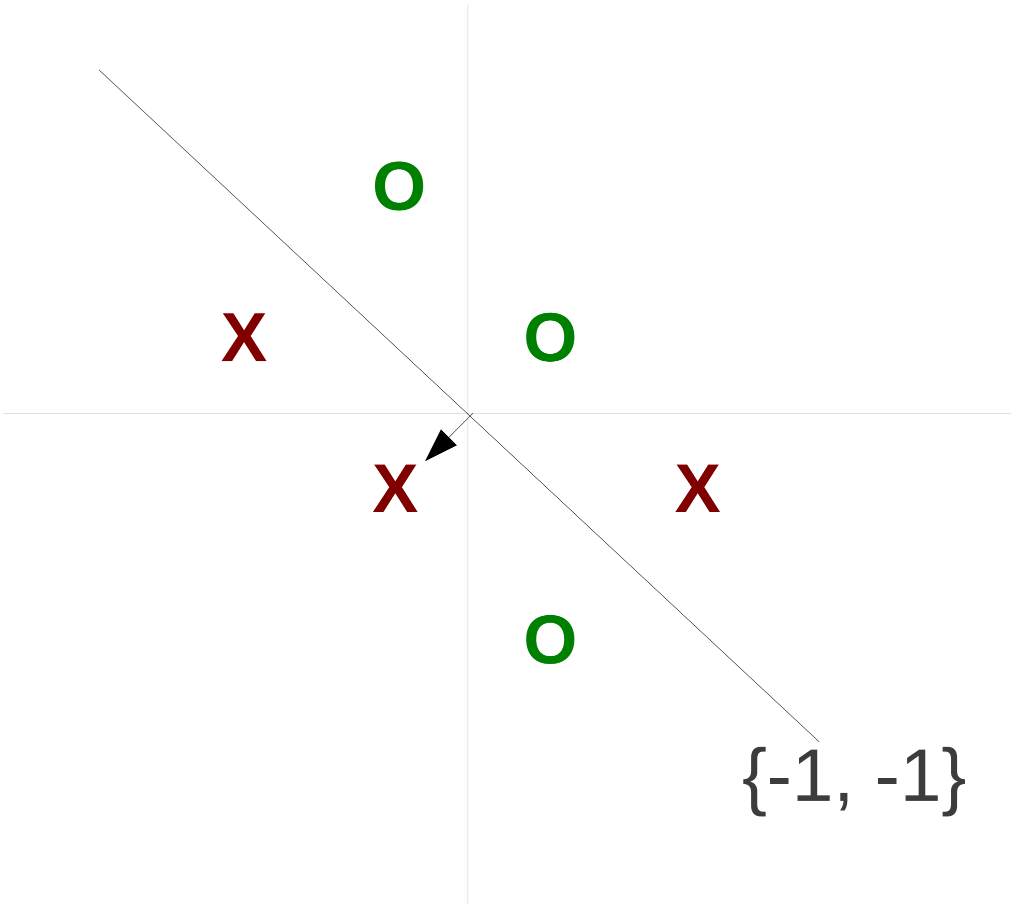
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



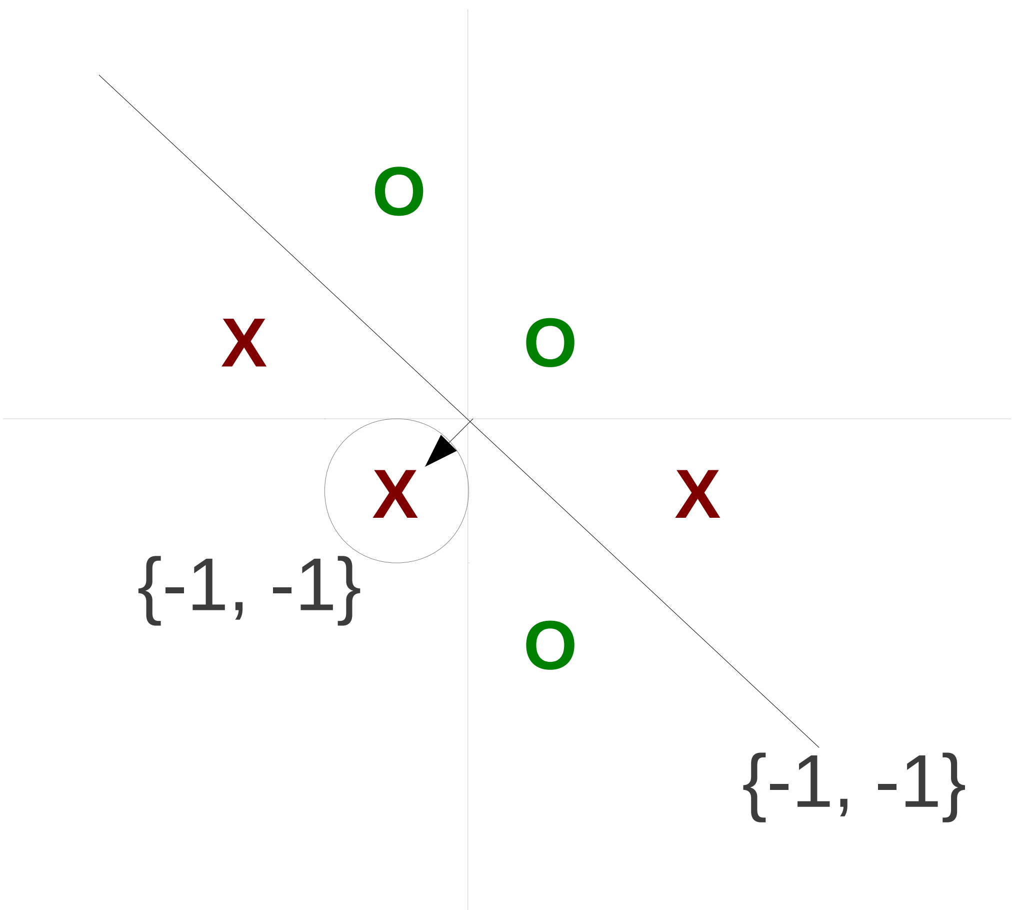
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



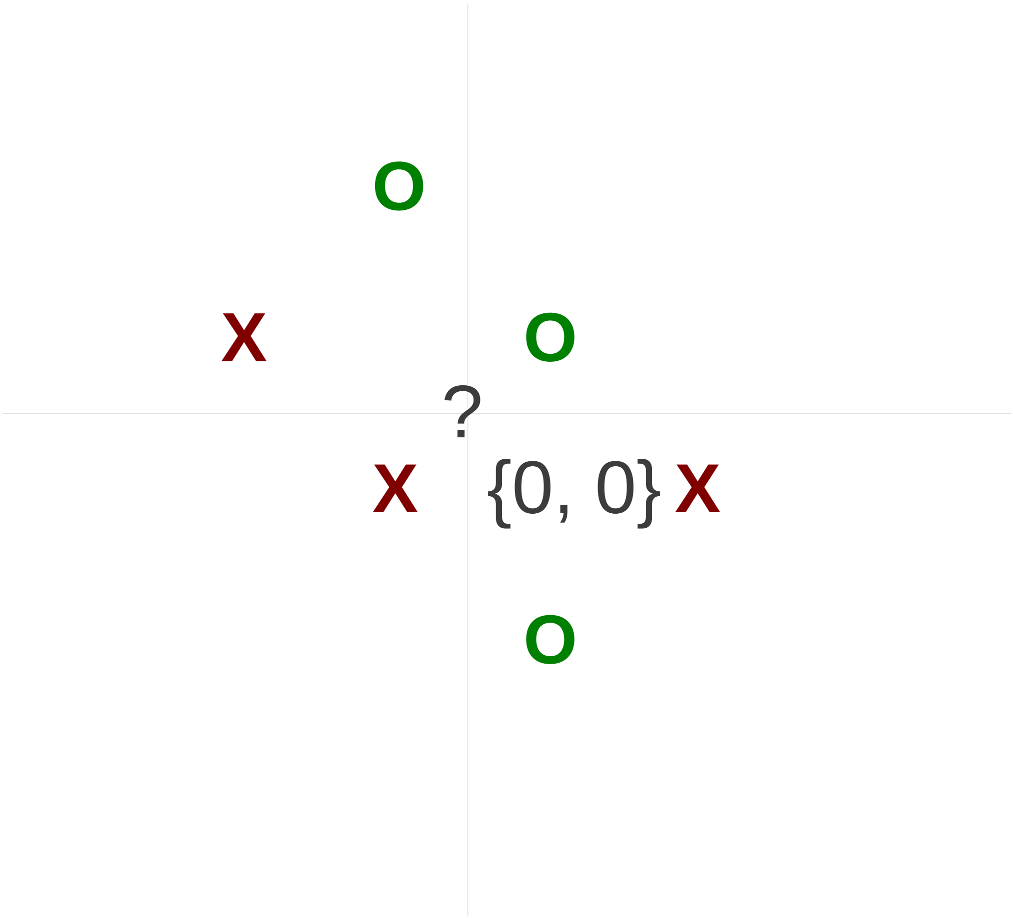
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



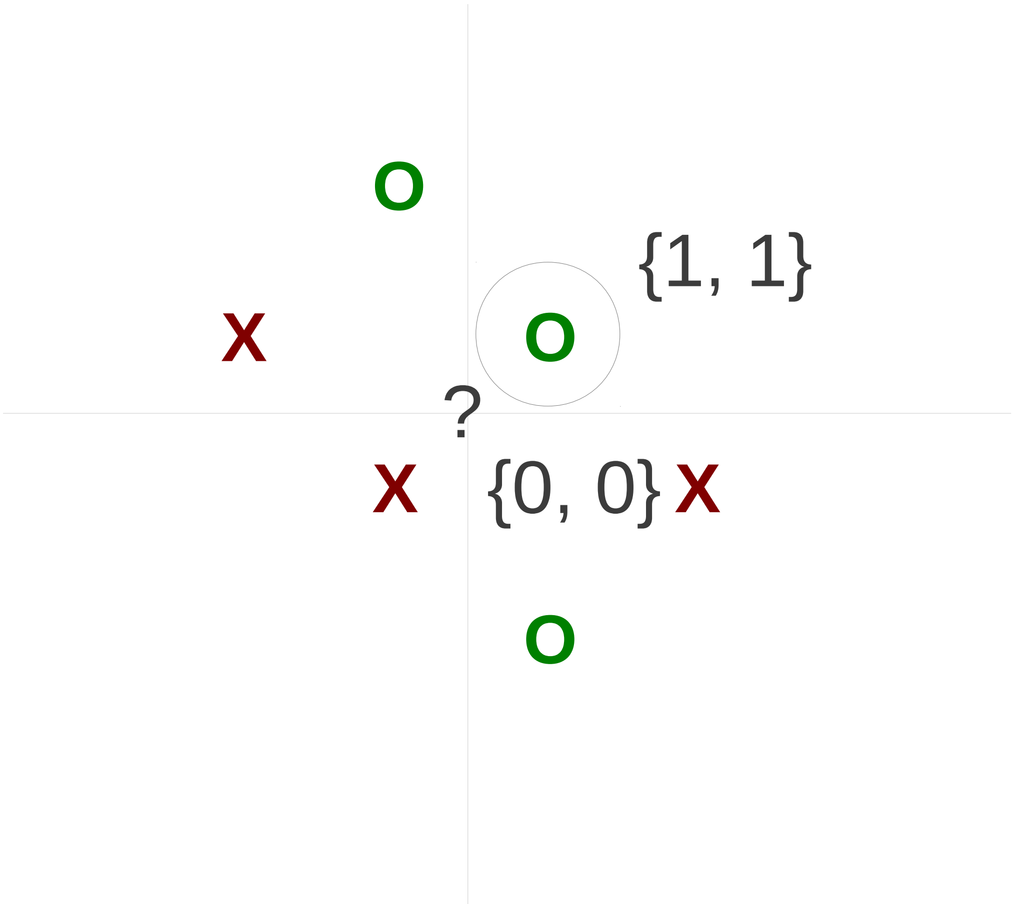
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



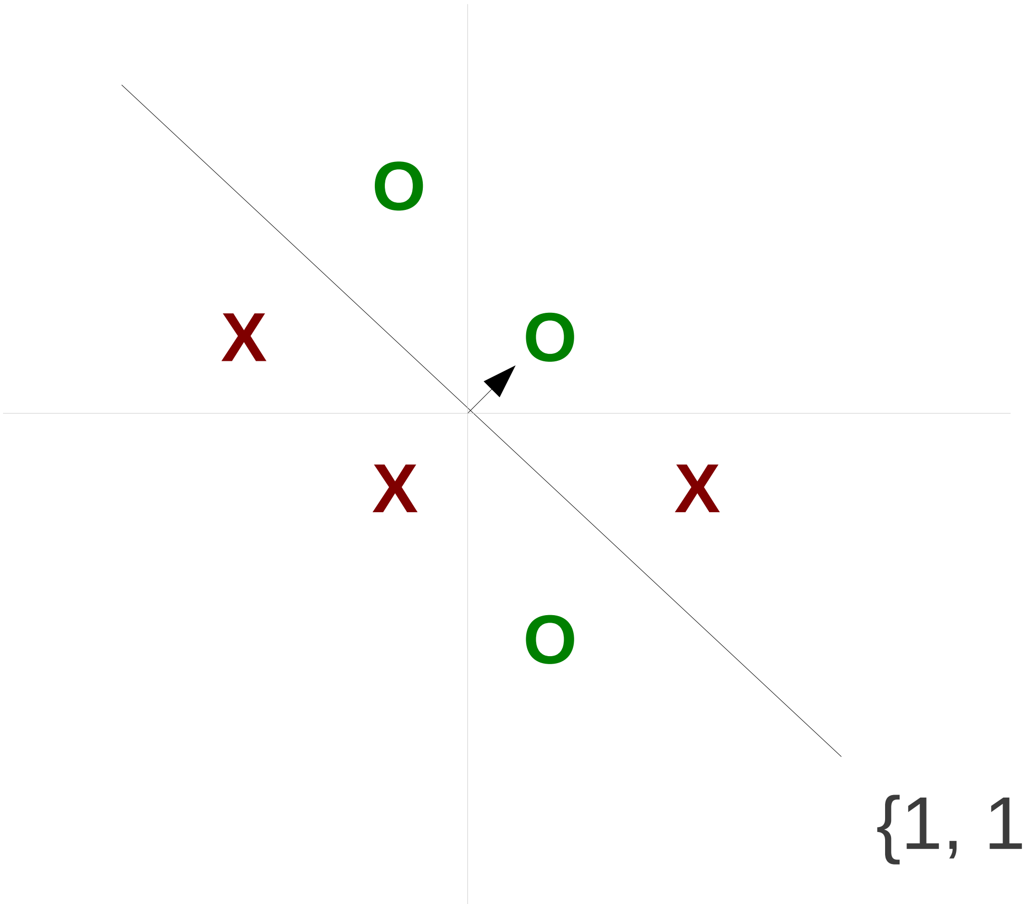
Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



Perceptron Instability

- Perceptron is instable with non-separable data
- Example:



{1, 1}

Result of Perceptron Training

- Long list of weights that never converges
- Accuracy greatly influenced by stopping point

{1, 1}
{2, -2}
{-1, -1}
{0, 0}
{1, 1}
{1, 1}
{1, 1}
{2, -2}
{-1, -1}
{0, 0}
{1, 1}
{1, 1}
{1, 1}
...

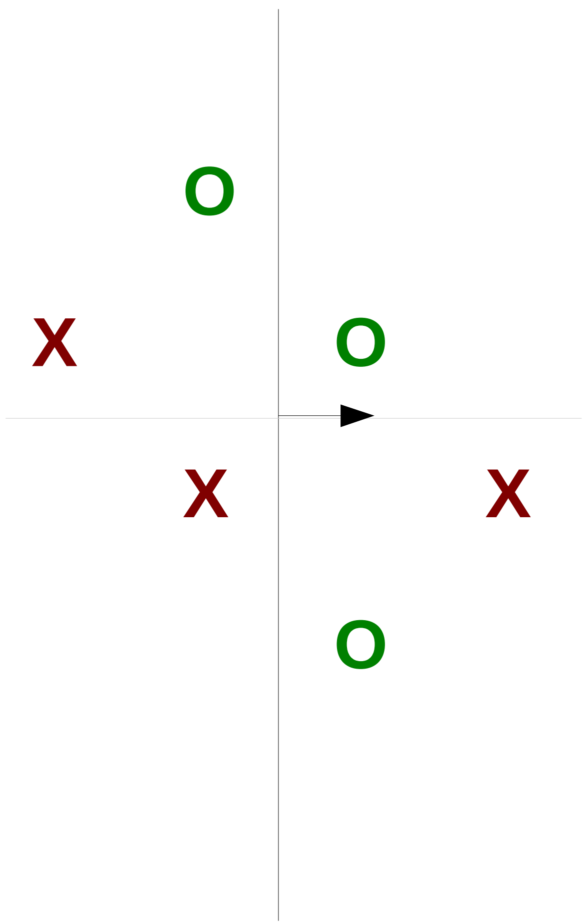
Not so bad...

Really bad!

Averaged Perceptron Idea

- Just take the average of the weights!

average({1, 1}
{2, -2}
{-1, -1}
{0, 0}
{1, 1}
{1, 1}
{1, 1}
{2, -2}
{-1, -1}
{0, 0}
{1, 1}
{1, 1}
{1, 1}
...) → {0.67, 0}



Averaged Perceptron in Code

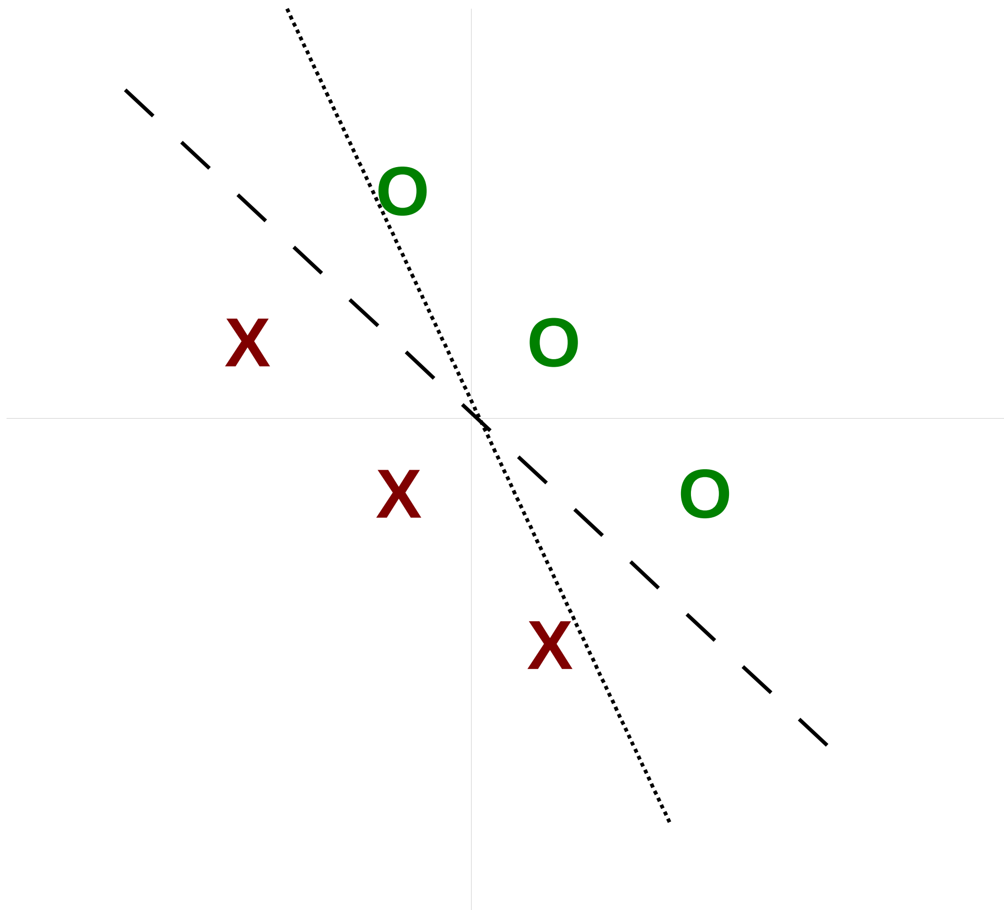
```
  create map w
★ create map avg
★ set updates = 0
  for I iterations
      for each labeled pair x, y in the data
          phi = CREATE_FEATURES(x)
          y' = PREDICT_ONE(w, phi)
          if y' != y
              UPDATE_WEIGHTS(w, phi, y)
★             updates += 1
★             avg = (avg * (updates-1) + w) / updates
```

- Change the average after every update

Classification Margins

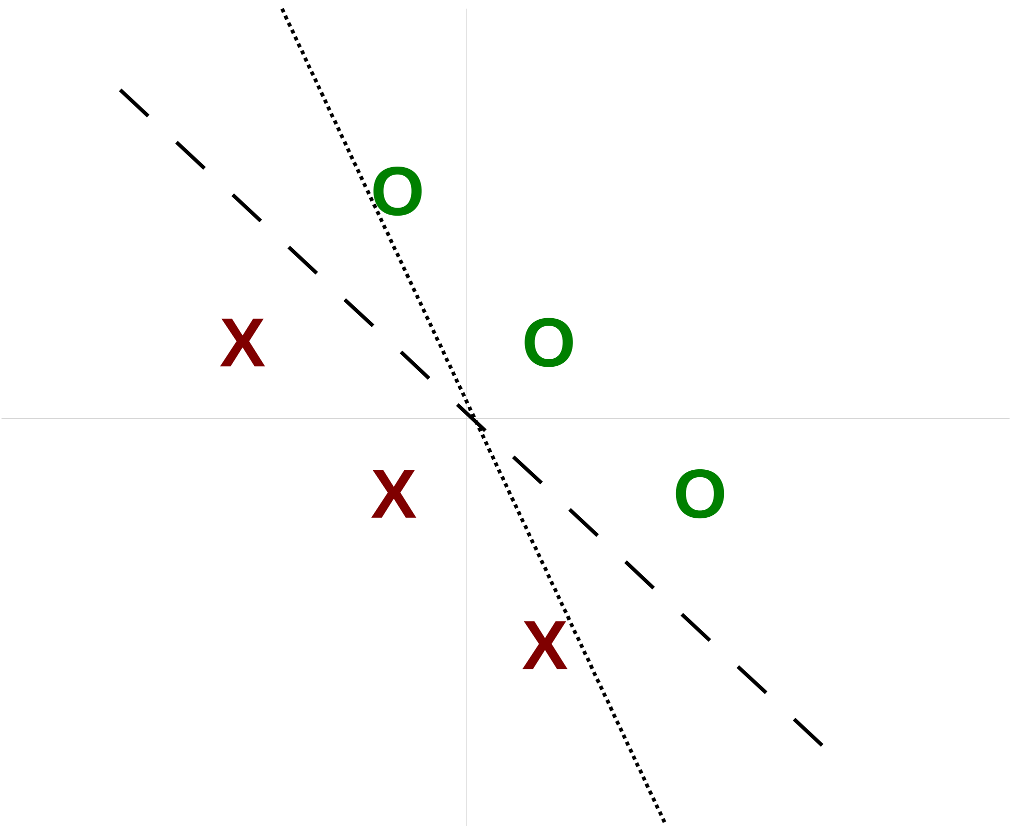
Choosing between Equally Accurate Classifiers

- Which classifier is better? Dotted or Dashed?



Choosing between Equally Accurate Classifiers

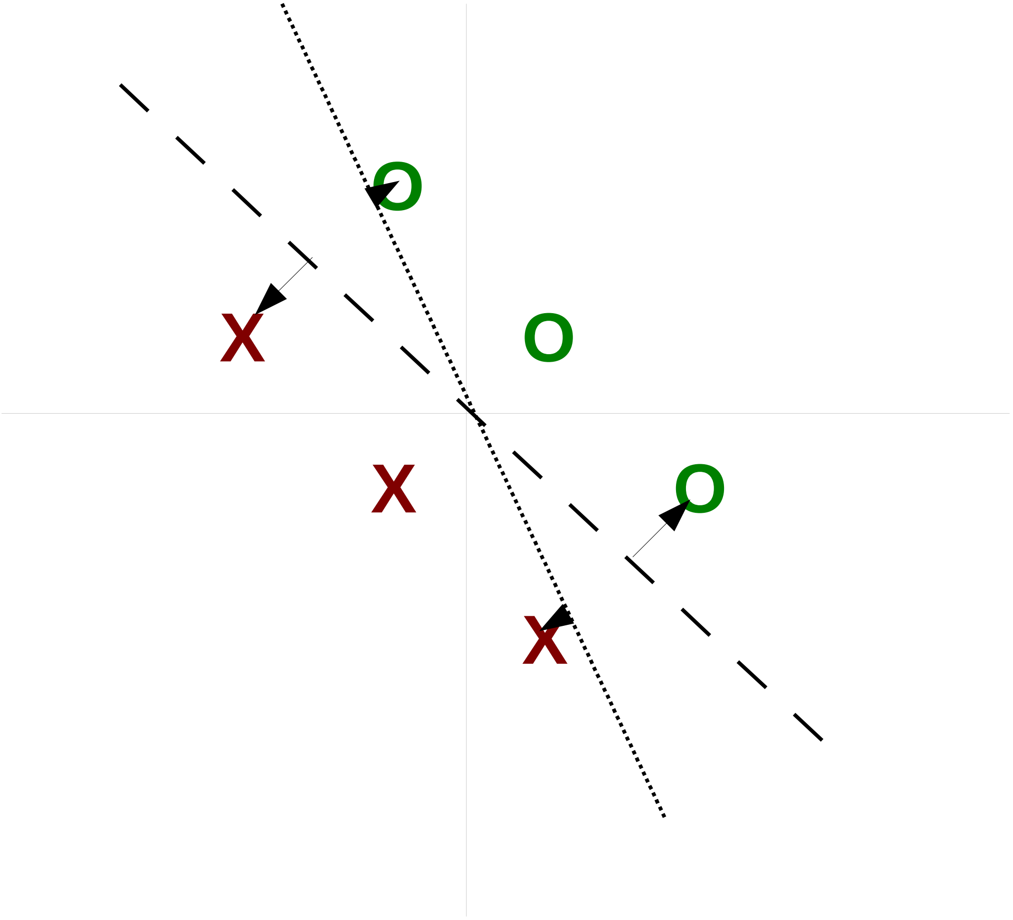
- Which classifier is better? Dotted or Dashed?



- Answer: Probably the dashed line.
- Why?: It has a larger margin.

What is a Margin?

- The distance between the classification plane and the nearest example:



Support Vector Machines

- Most famous margin-based classifier
  - Hard Margin: Explicitly maximize the margin
  - Soft Margin: Allow for some mistakes
- Usually use batch learning
  - Batch learning: slightly higher accuracy, more stable
  - Online learning: simpler, less memory, faster convergence
- Learn more about SVMs: http://disi.unitn.it/moschitti/material/Interspeech2010-Tutorial.Moschitti.pdf
- Batch learning libraries: LIBSVM, LIBLINEAR, SVMLite

Online Learning with a Margin

- Penalize not only mistakes, but also correct answers under a margin

```
create map w
for I iterations
    for each labeled pair x, y in the data
        phi = CREATE_FEATURES(x)
        val = w * phi * y
★       if val <= margin
            UPDATE_WEIGHTS(w, phi, y)
```

(A correct classifier will always make w * phi * y > 0)
If margin = 0, this is the perceptron algorithm

Regularization

Cannot Distinguish Between Large and Small Classifiers

- For these examples:

-1	he saw a bird in the park
+1	he saw a robbery in the park

- Which classifier is better?

Classifier 1	Classifier 2
he +3	bird -1
saw -5	robbery +1
a +0.5	
bird -1	
robbery +1	
in +5	
the -3	
park -2	

Cannot Distinguish Between Large and Small Classifiers

- For these examples:

-1	he saw a bird in the park
+1	he saw a robbery in the park

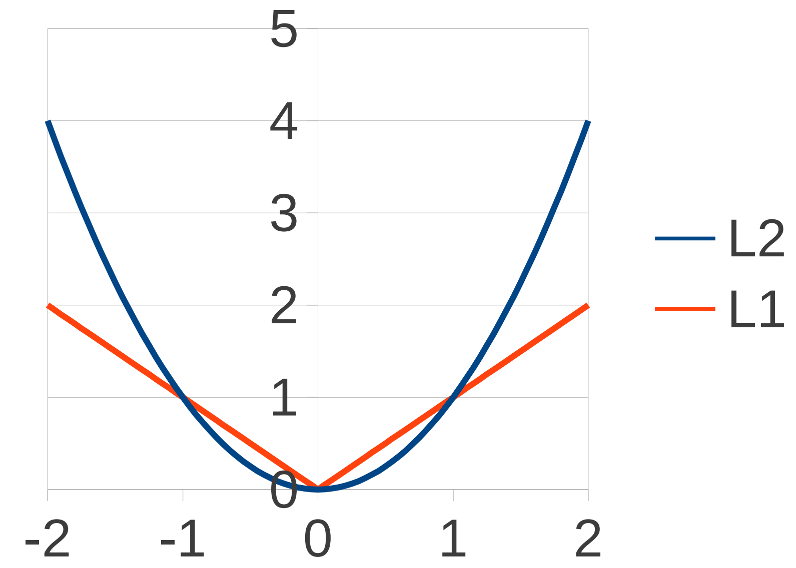
- Which classifier is better?

Classifier 1	Classifier 2
he +3	bird -1
saw -5	robbery +1
a +0.5	
bird -1	
robbery +1	
in +5	
the -3	
park -2	

Probably classifier 2!
It doesn't use
irrelevant information.

Regularization

- A penalty on adding extra weights
- L2 regularization:
  - Big penalty on large weights, small penalty on small weights
  - High accuracy
- L1 regularization:
  - Uniform increase whether large or small
  - Will cause many weights to become zero → small model



L1 Regularization in Online Learning

- After update, reduce the weight by a constant c

```
UPDATE_WEIGHTS(w, phi, y, c)
★   for name, value in w:
★       if ABS(value) < c:
★           w[name] = 0
★       else:
★           w[name] -= SIGN(value) * c
    for name, value in phi:
        w[name] += value * y
```

If abs. value < c, set weight to zero

If value > 0, decrease by c
If value < 0, increase by c

Example

- Every turn, we Regularize, Update, Regularize, Update

Regularization: c=0.1
Updates: {1, 0} on 1st and 5th turns
{0, -1} on 3rd turn

	R_1	U_1	R_2	U_2	R_3	U_3
Change:	{0, 0}	{1, 0}	{-0.1, 0}	{0, 0}	{-0.1, 0}	{0, -1}
w:	{0, 0}	{1, 0}	{0.9, 0}	{0.9, 0}	{0.8, 0}	{0.8, -1}

	R_4	U_4	R_5	U_5	R_6	U_6
Change:	{-0.1, 0.1}	{0, 0}	{-0.1, 0.1}	{1, 0}	{-0.1, 0.1}	{0, 0}
w:	{0.7, -0.9}	{0.7, -0.9}	{0.6, -0.8}	{1.6, -0.8}	{1.5, -0.7}	{1.5, -0.7}

Efficiency Problems

- Typical number of features:
  - Each sentence (phi): 10~1000
  - Overall (w): 1,000,000~100,000,000

```
UPDATE_WEIGHTS(w, phi, y, c)
  for name, value in w:
    if ABS(value) <= c:
      w[name] = 0
    else:
      w[name] -= SIGN(value) * c
  for name, value in phi:
    w[name] += value * y
```

This loop is VERY SLOW!

Efficiency Trick

- Regularize only when the value is used!

```
GETW(w, name, c, iter, last)
    if iter != last[name]:          # regularize several times
        c_size = c * (iter - last[name])
        if ABS(w[name]) <= c_size:
            w[name] = 0
        else:
            w[name] -= SIGN(w[name]) * c_size
        last[name] = iter
    return w[name]
```

- This is called “lazy evaluation”, used in many applications

Choosing the Regularization Constant

- The regularization constant c has a large effect
- Large value
  - small model
  - lower score on training set
  - less overfitting
- Small value
  - large model
  - higher score on training set
  - more overfitting
- Choose best regularization value on development set
  - e.g. 0.0001, 0.001, 0.01, 0.1, 1.0

Exercise

Exercise

- Write program:
  - train-svm: Creates an svm model with L1 regularization constant 0.001 and margin 1
- Train a model on data-en/titles-en-train.labeled
- Predict the labels of data-en/titles-en-test.word
- Grade your answers and compare them with the perceptron
  - script/grade-prediction.py data-en/titles-en-test.labeled your_answer
- Extra challenge:
  - Try many different regularization constants
  - Implement the efficiency trick

Thank You!