Regression

Byungkon Kang
SUNY Korea, Dept. of CS
Week 2

Outline

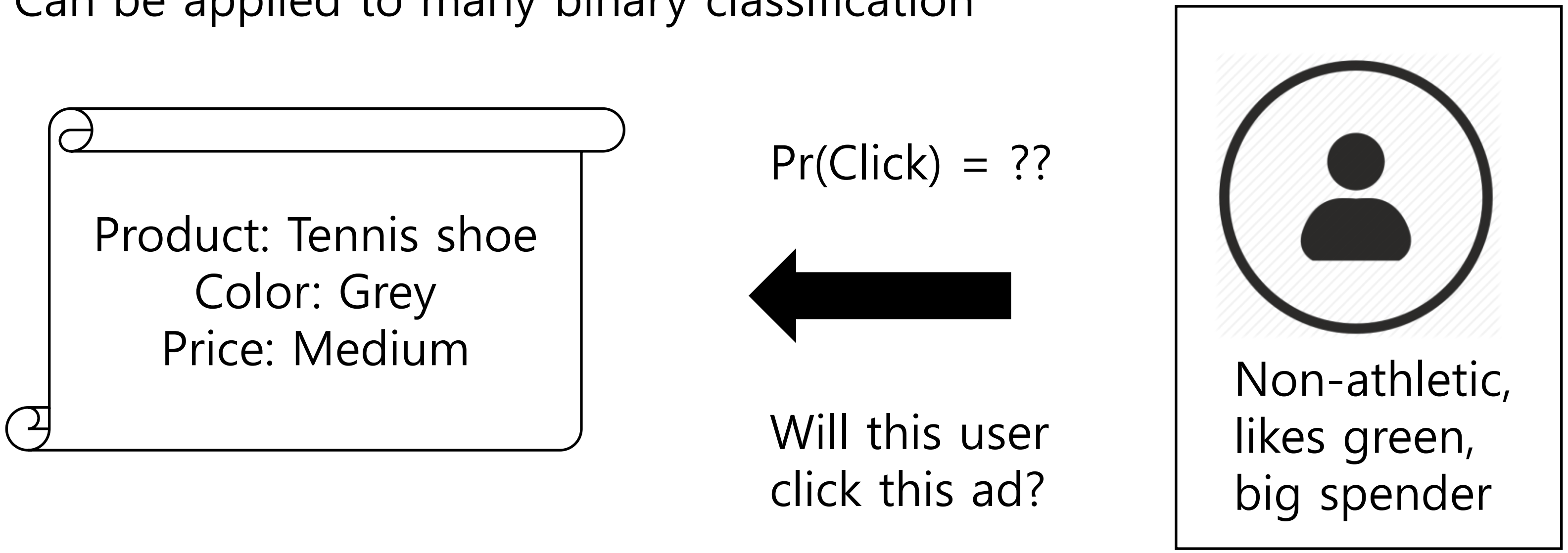
- Linear regression

- Logistic regression

Problem setting

- Example: How likely is a user clicking an online ad?
  - Inputs: User's profile, Ad's profile

- Regression on probability
  - But actually a classification problem
  - Can be applied to many binary classification



Product: Tennis shoe
Color: Grey
Price: Medium

Pr(Click) = ??

Will this user
click this ad?

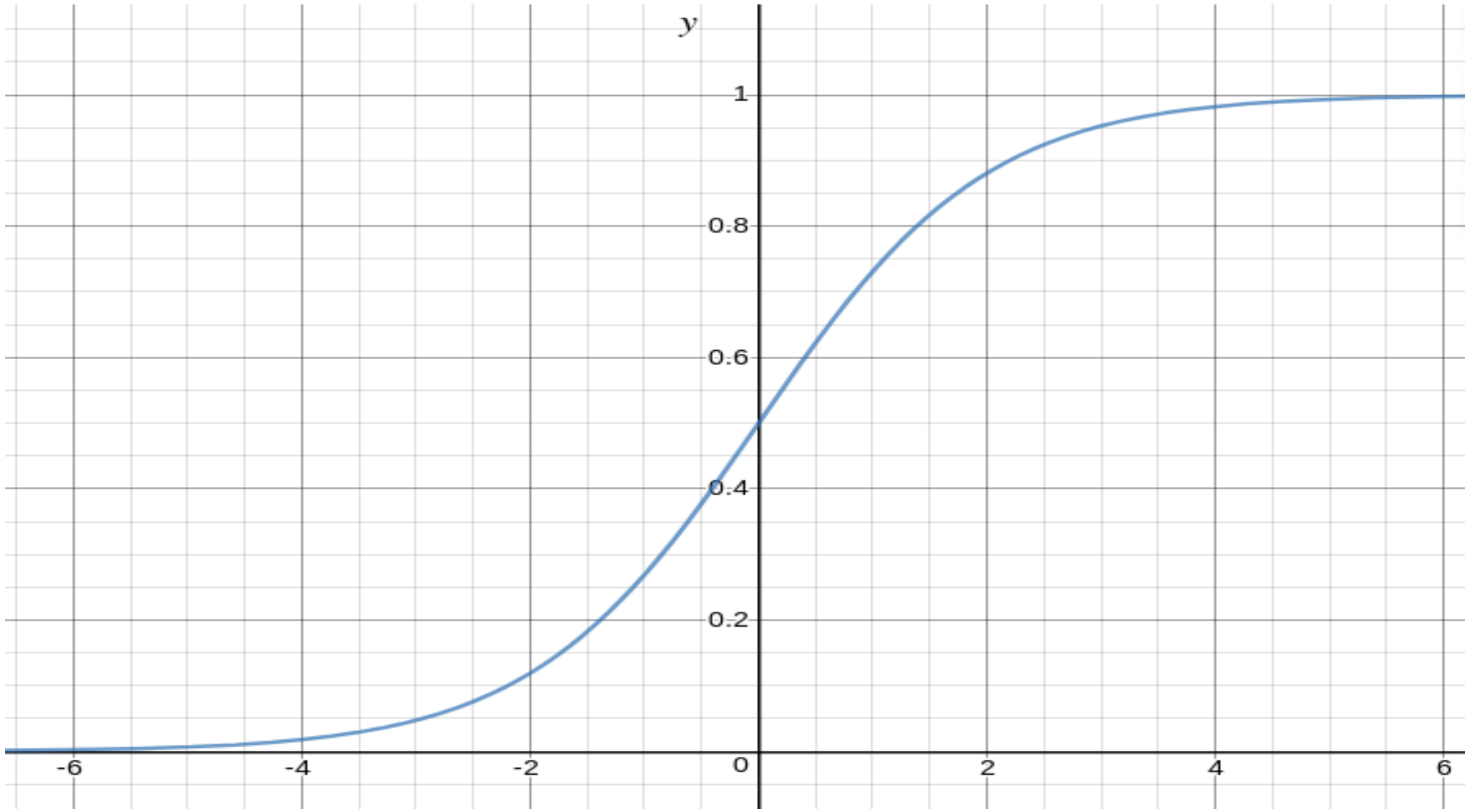
Non-athletic,
likes green,
big spender

Modeling probability

- What function ALWAYS produces value in range [0, 1]?

Can replace this with any function of x

$$\text{Logistic function } \sigma(x) = \frac{1}{1+\exp(-\boxed{x})} = \frac{\exp(x)}{1+\exp(x)}$$



Modeling probability

- Why not use a more expressive form for x?

$$\sigma(x) = \frac{1}{1 + \exp(-g(x))}, g(x) = \theta_1 x + \theta_0$$

- Multi-variate input x:

$$g(x) = \theta_1^T x + \theta_0$$

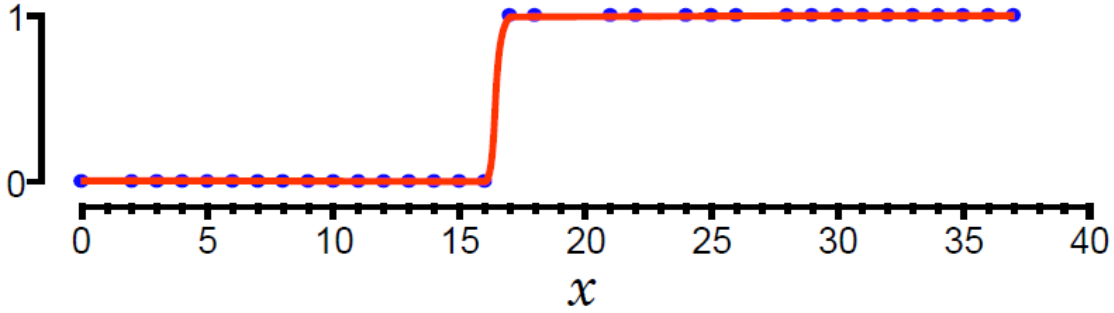
- Rearranging the terms, we can see

$$\log \frac{\sigma(x)}{1 - \sigma(x)} = \theta_1^T x + \theta_0$$

  - Linear regression on log-odds (well, sort of…)

How to train

- Typical regression algorithms are trained to minimize the residual
  - In logistic regression, the ground-truths are either 0 or 1
  - So should we just find the best-fit logistic curve using MSE?



- What’s wrong with this?
  - It’s not really regression if there are only two target values
  - MSE is not related to *accuracy*
  - We wish to model the *probability* of success: Need probabilistic interpretation of the model

Training via maximum likelihood

- Given the training data $\mathrm{D} = \{(x_i, y_i) | x_i \in \mathbb{R}, y_i \in \{0,1\}\}$
    - What is the log likelihood of D under our logistic regression model?

$$\log \Pr(D;\theta) = \sum_{i|y_i=1} \log \sigma(x_i) + \sum_{i|y_i=0} \log\left(1-\sigma(x_i)\right)$$

-
    - We should find the parameters that maximize this (log) likelihood

- Differentiate!

$$\Pr\left((x_i, y_i = 0)\right) = \Pr(y_i = 0 | x_i) = 1 - \sigma(\theta_1 x_i + \theta_0)$$

$$\left.\begin{aligned} \frac{\partial \log \Pr(D;\theta)}{\partial \theta_1} &= \sum_{y_i=1} \left(1-\sigma(x_i)\right) x_i + \sum_{y_i=0} \sigma(x_i) x_i = 0, \\ \frac{\partial \log \Pr(D;\theta)}{\partial \theta_0} &= \sum_{y_i=1} \left(1-\sigma(x_i)\right) + \sum_{y_i=0} \sigma(x_i) = 0 \end{aligned}\right\}$$

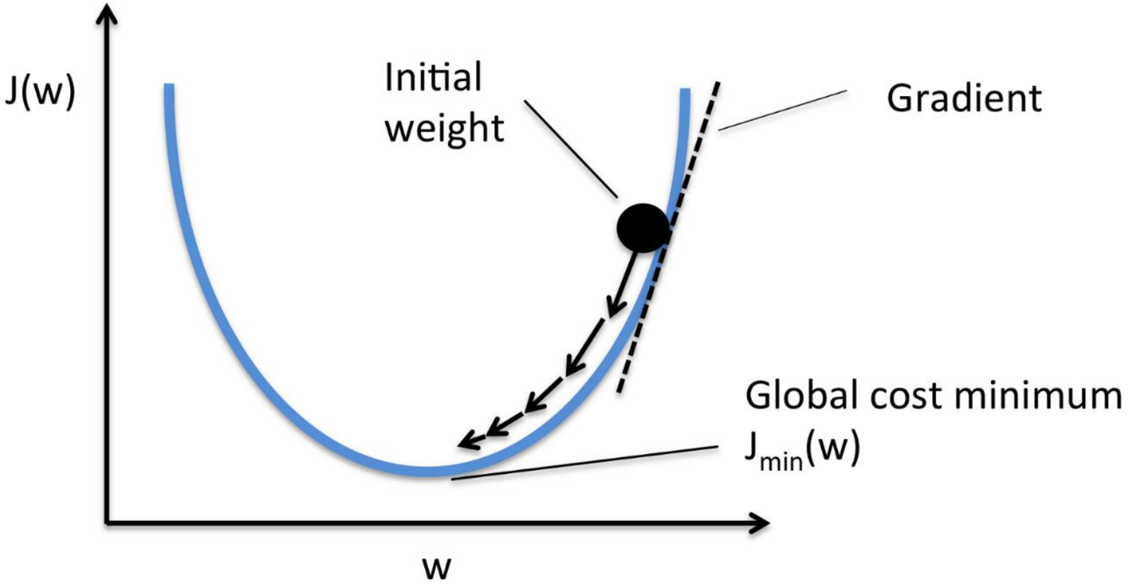
System of non-linear equations that don’t always yield closed-form solutions

- Need to use iterative methods (e.g., gradient descent)

Optimization

- Since we can’t find a closed-form solution, we use iterative methods
    - Also called “Numerical methods”, since relies on numerical analysis

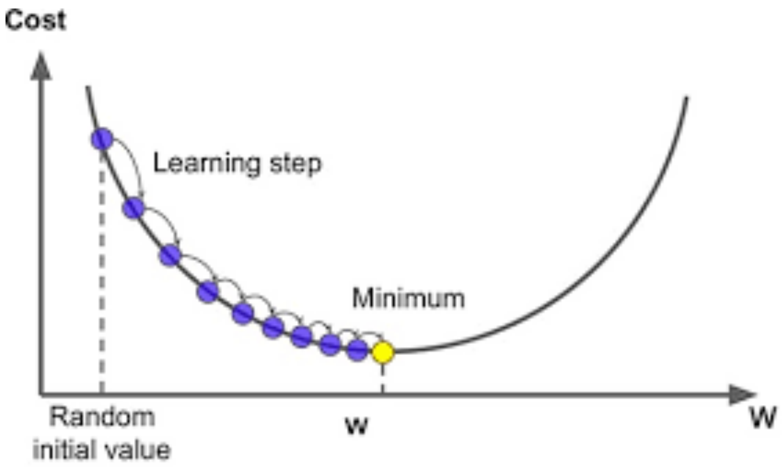
- Gradient descent
    - Gradually move in the direction of steepest descent (i.e., opposite of gradient)



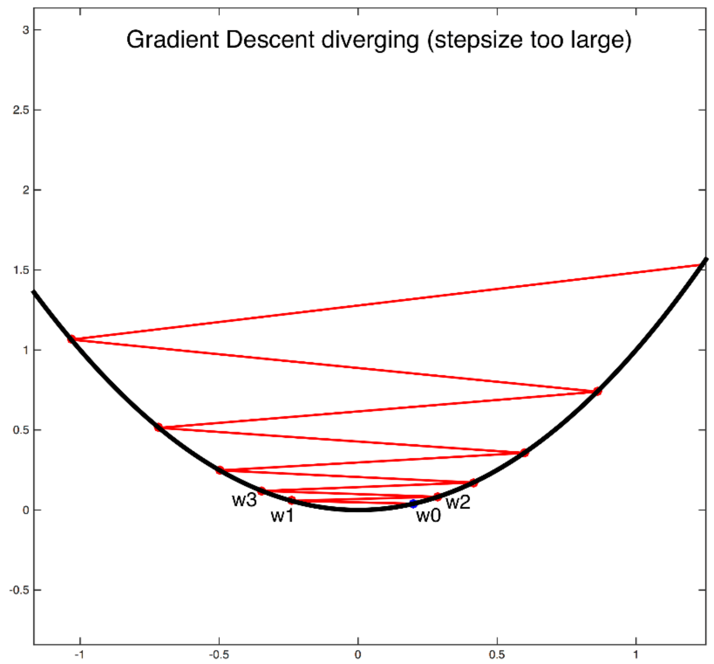
Gradient descent

- As always, we wish to find the minimum of our loss (cost) function

- If we move in the direction of gradient, we have better hope of finding the minimum
    - But if we move TOO much, we might overshoot



https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/



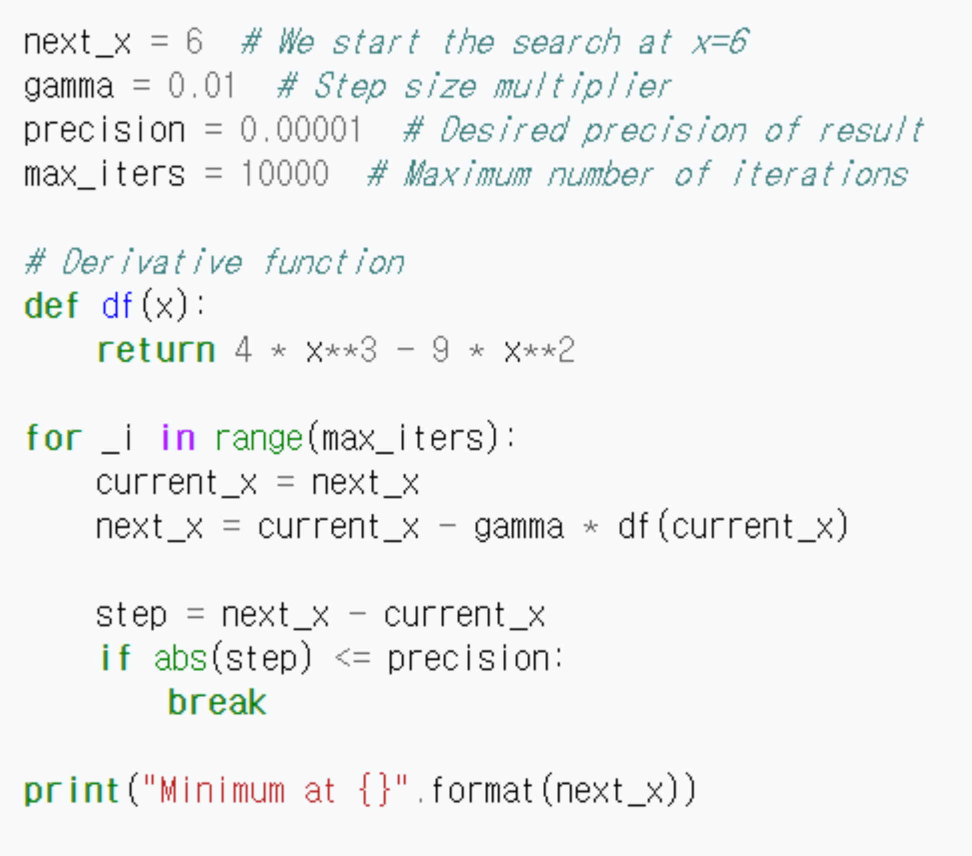
Gradient descent pseudocode

- In pseudocode:

```
1. Let J(θ) be the cost function
2. Initialize θ*
3. While not converged:
    1. $\theta^* \leftarrow \theta^* - \boxed{\gamma}\boxed{\frac{\partial J}{\partial \theta}}$
    2. Adjust step size $\gamma$ (if necessary)
```

//Update w* in the direction that decreases the loss the most, but only by a small amount

- Python example (Thank you Wikipedia)



```python
next_x = 6  # We start the search at x=6
gamma = 0.01  # Step size multiplier
precision = 0.00001  # Desired precision of result
max_iters = 10000  # Maximum number of iterations

# Derivative function
def df(x):
    return 4 * x**3 - 9 * x**2

for _i in range(max_iters):
    current_x = next_x
    next_x = current_x - gamma * df(current_x)

    step = next_x - current_x
    if abs(step) <= precision:
        break

print("Minimum at {}".format(next_x))
```

Where the decision boundary lies

- How do we predict?
  - If the probability is >0.5, we say "yes", otherwise, "no"

- To see the boundary, simply set the probability to 0.5, and see what happens:

$$\frac{1}{1+\exp(-x\theta_1-\theta_0)}=\frac{1}{2}$$

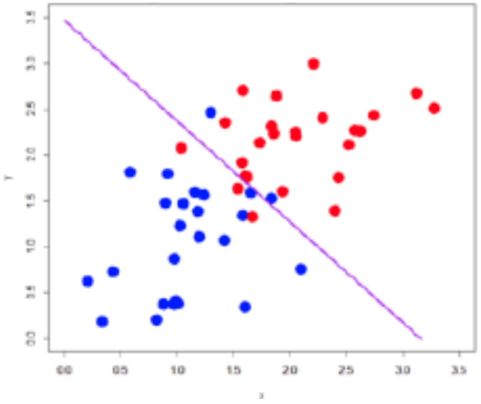
$$\Downarrow$$

$$\exp(-x\theta_1-\theta_0)=1$$

$$\Downarrow$$

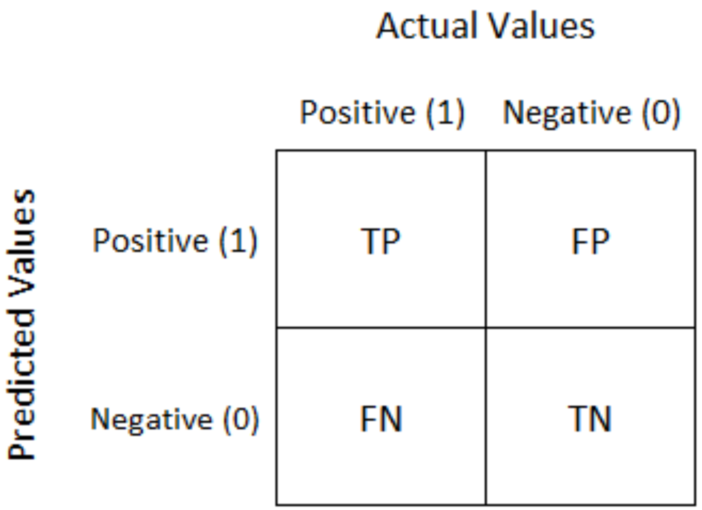
$$x\theta_1+\theta_0=0$$

  - This is just a linear function!
  - Logistic regression is a linear classifier



How well did we do?

- Sometimes, it's not all about getting the right answer
  - We also have to successfully reject the wrong answers



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

GT	PD	
1	0	FN
0	0	TN
0	0	TN
1	1	TP
1	1	TP
1	1	TP
1	0	FN
0	1	FP
0	0	TN
0	0	TN

Precision: 0.75
Recall: 0.6
F1: 0.667
Accuracy: 0.7

- Precision: How many of the found items are correct?
  - True positive / (True positive + False positive)

- Recall: How many of the relevant items did we find?
  - True positive / (False negative + True positive)

How well did we do?

- Useful for class imbalance
  - If 70% of your data has label ‘1’, you can still get 70% accuracy by just saying ‘1’
  - But that will drastically lower the precision

Precision = True positive / (True positive + False positive)

- Combined measure: F1 score

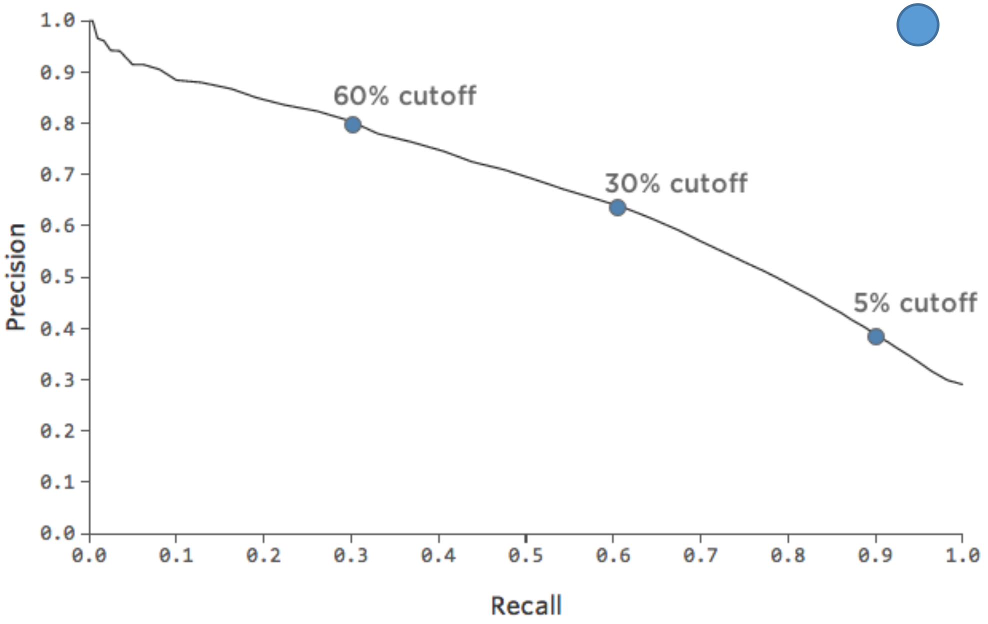
$$F1 = 2\frac{PR}{P + R}$$

  - Harmonic mean of precision and recall
  - Both precision and recall have to be high

How well did we do?

- Precision-recall curve
  - Recall that our decision rule is: IF p(x) > 0.5 THEN ‘1’ ELSE ‘0’
  - We can see how the results change if we vary the threshold 0.5
  - Collect and plot the (recall, precision) points for many different threshold values

Optimal point (100% recall and precision)



c.f., ROC curve
(Receiver operating
characteristic curve)

We want more than two

- How do we extend beyond binary classification? (K classes, K > 2)

- Setting: $\mathrm{D} = \{(x_i, y_i) | x_i \in \mathbb{R}, y_i \in [K]\}$

- Main idea:
  - Run multiple binary regressions on each class independently
  - i.e., set a parameter for each class (plus one for bias)

$$\Pr(Y_i = k | X_i) \propto \exp(\theta_k X_i + \theta_0)$$

  - Why do we use an exponential?
    i) good way to ensure non-negativity
    ii) Log-linear interpretation

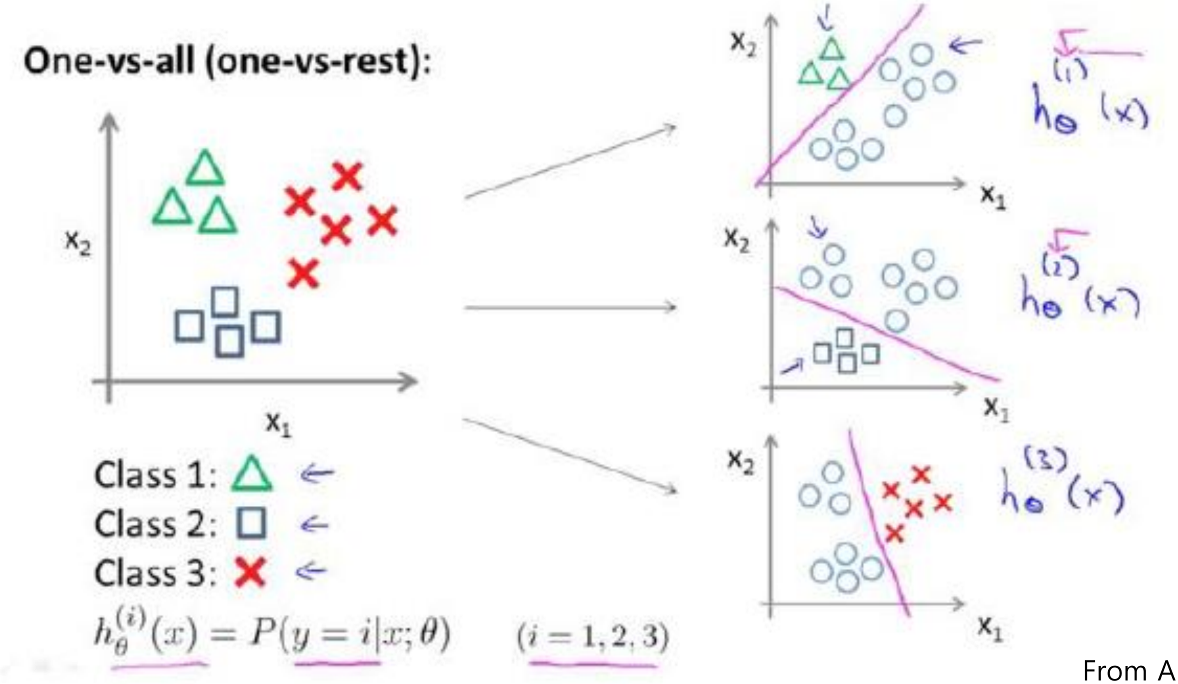
$$\Pr(Y_i = k) = \frac{\exp(\theta_k X_i + \theta_0)}{\sum_j \exp\left(\theta_j X_i + \theta_0\right)}$$

A.K.A., "Softmax regression"

We want more than two

- One-vs-All
    - Also common in multi-class classification problems

- To predict among K classes, train K binary logistic regressors
    - Choose the class that yields highest probability



From Andrew Ng's note

What about multi-class precision/recall? (Macro avg.)

- Construct a ***confusion matrix*** as follows
  - Each entry is the # of predictions made

		Ground-truth labels					
		L1	L2	L3	L4	L5	L6
Predicted labels	L1	3	3	4	1	3	3
	L2	0	15	2	0	0	0
	L3	3	2	10	0	2	0
	L4	1	1	1	14	0	0
	L5	0	0	4	1	12	0
	L6	0	0	0	0	0	17

Prec(L1) = 3 / (3+3+4+1+3+3) = 3/17

Rec(L1) = 3 / (3+0+3+1+0+0) = 3/7

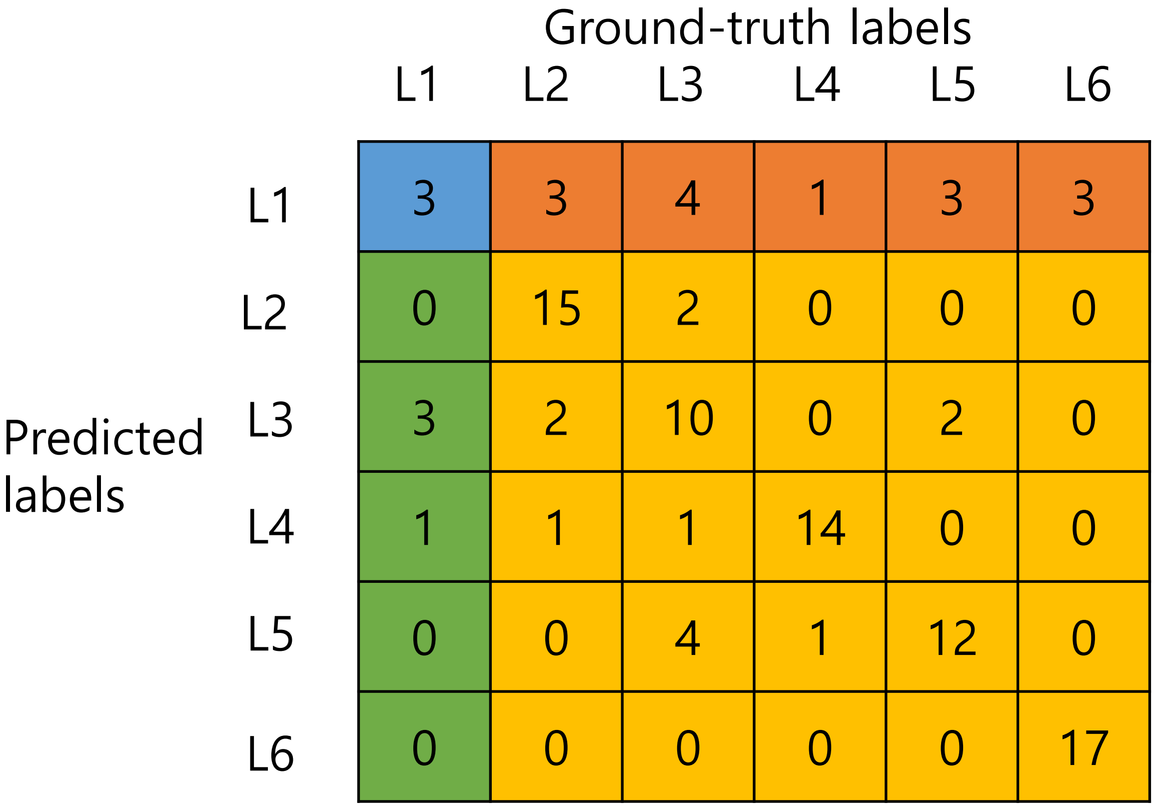
Simply average row-by-row and column-by-column

Then average over all labels for the final precision/recall

Prec = Avg(Prec(L1), … , Prec(L6))
Rec = Avg(Rec(L1), …, Rec(L6))

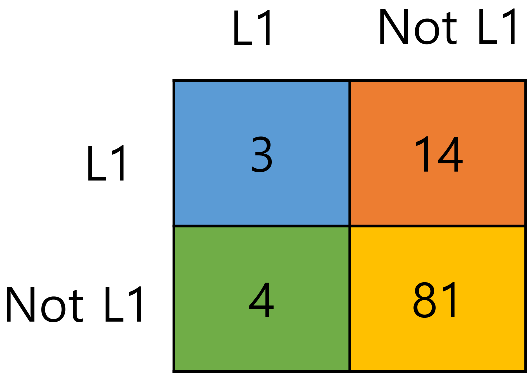
What about multi-class precision/recall? (Micro avg.)

- Construct a ***confusion matrix*** as follows
    - Each entry is the # of predictions made



		Ground-truth labels					
		L1	L2	L3	L4	L5	L6
Predicted labels	L1	3	3	4	1	3	3
	L2	0	15	2	0	0	0
	L3	3	2	10	0	2	0
	L4	1	1	1	14	0	0
	L5	0	0	4	1	12	0
	L6	0	0	0	0	0	17

Construct a confusion matrix for each label with 1-vs-all criterion

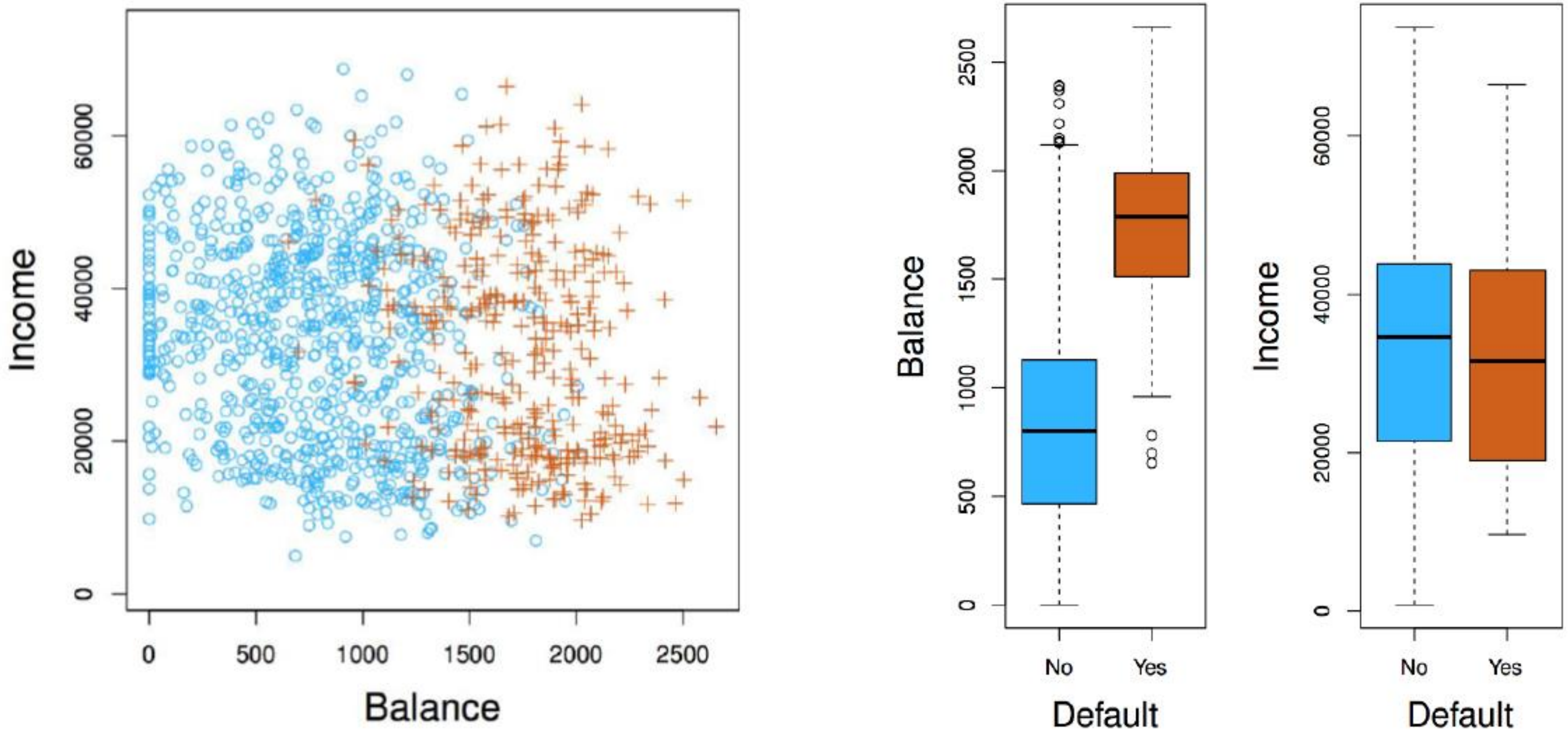


	L1	Not L1
L1	3	14
Not L1	4	81

Then construct the aggregate confusion matrix by summing over all confusion matrices before computing the prec./rec. in the usual manner.

Example: Credit card default prediction

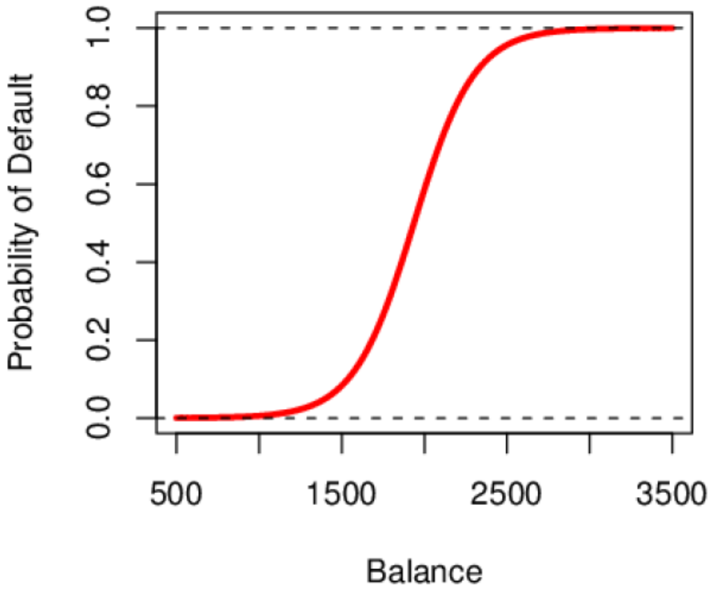
- Will a given customer default on his/her payment?
  - Customer features: Annual income, or monthly credit card balance
  - Output: Yes / No



Example: Credit card default prediction

- Since ‘balance’ is more decisive, we’ll use that as an input

- Learned model:



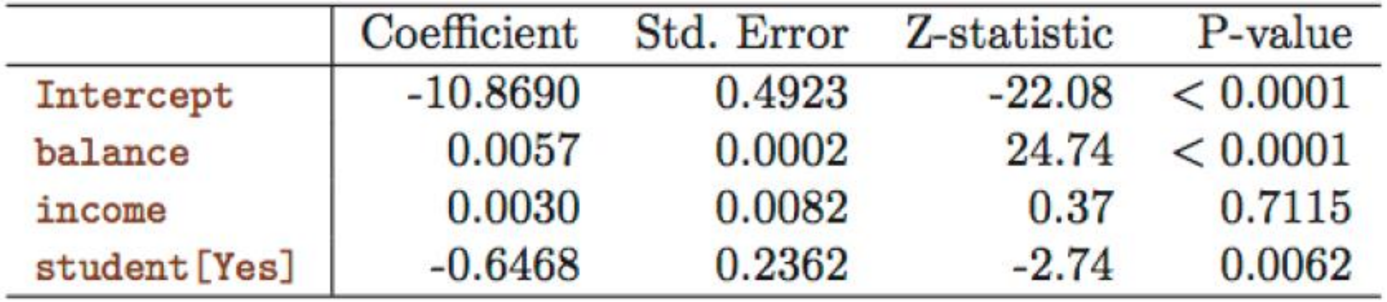
	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.6513	0.3612	-29.5	< 0.0001
`balance`	0.0055	0.0002	24.9	< 0.0001

Coefficient for ‘balance’ is positive
→ default probability increases with balance

Example: Credit card default prediction

- Let's use more information
  - Balance, income, and student-ness
  - Student-ness is a binary variable that's 1 for student, 0 for not

- Learned model

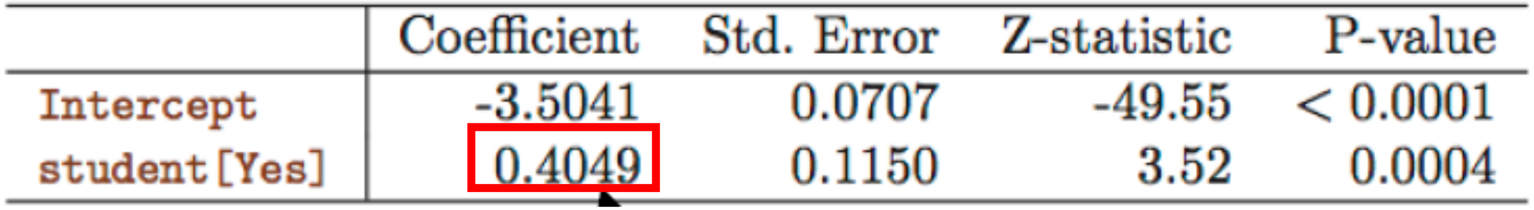


	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.8690	0.4923	-22.08	< 0.0001
`balance`	0.0057	0.0002	24.74	< 0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	-0.6468	0.2362	-2.74	0.0062

Example: Credit card default prediction

- What if we isolate 'student'?

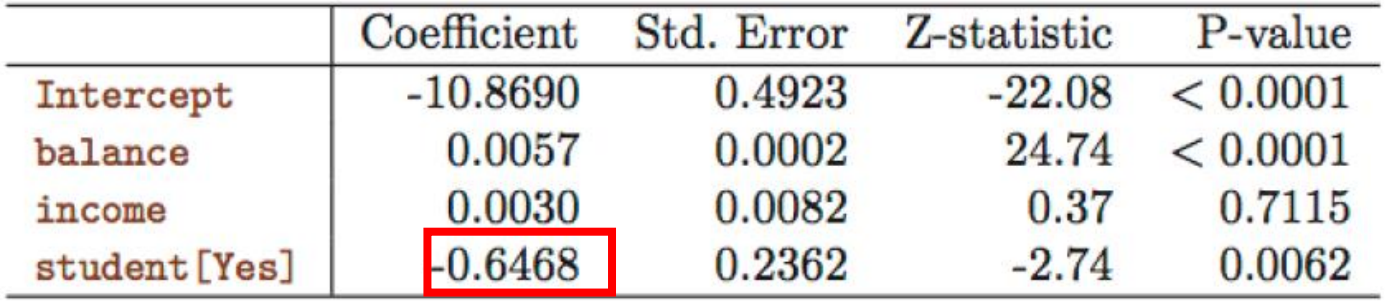
- Learned model



	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-3.5041	0.0707	-49.55	< 0.0001
student[Yes]	0.4049	0.1150	3.52	0.0004

Positive

- But wait…

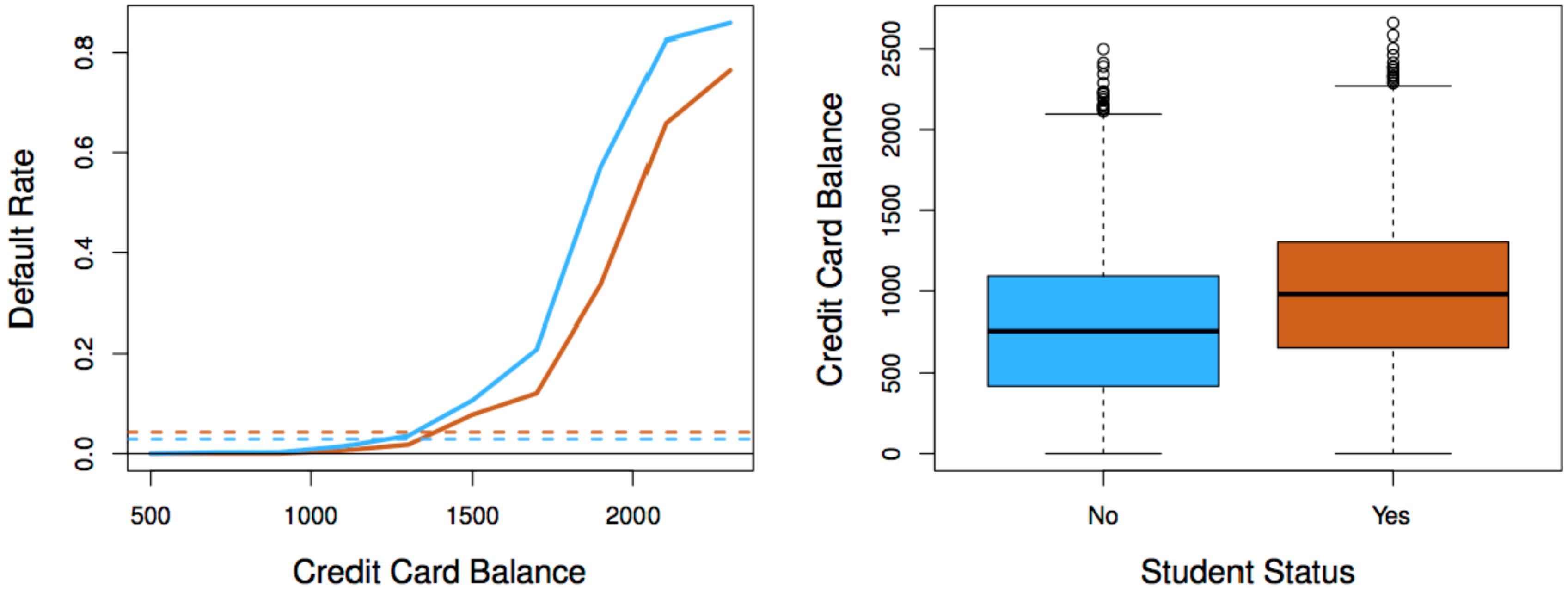


	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-10.8690	0.4923	-22.08	< 0.0001
balance	0.0057	0.0002	24.74	< 0.0001
income	0.0030	0.0082	0.37	0.7115
student[Yes]	-0.6468	0.2362	-2.74	0.0062

Negative

Example: Credit card default prediction

- Student vs. non-student



Default more likely for non-students
(Possibly because of more spending?)

Summary

- Regression is used to predict continuous outcomes for each input
  - Classification is used to predict categorical outcomes

- Linear regression assumes a linear relationship between input-output
  - 'Y = aX + b', where the trainable parameters are {a, b}
  - Admits closed-form solution (OLS)
  - General enough to model many other functions (polynomial, sinusoidal, etc.)

- Logistic regression tries to predict the *probability* of a categorical outcome
  - Mostly used as a classification model, despite its name
  - Doesn't have closed-form solution (must rely on iterative numerical methods)
  - Produces linear decision boundary