CS 360: Machine Learning

Sara Mathieson, Sorelle Friedler

Spring 2024

HAVERFORD
COLLEGE

Admin

- Sorelle office hours today **4-5pm in H110**
- **Lab 5** due TODAY
  - Grading will be generous, turn in what you have!
- Reminder about **extra credit for handout solutions** (see Piazza)
  - Deadline: 24 hours before the midterm

Midterm:
- In class on **Tuesday**
- You may bring a one page (front and back) “study sheet” (handwritten, created by you)

Feedback forms

Understand well

- KNN, KD-trees
- Ensembles, bootstrap
- AdaBoost
- Decision Trees
- Naïve Bayes
- Logistic Regression

Most confusing

- Continuous vs. binary features
- Bias-variance tradeoff
- Logistic regression
- Stochastic gradient descent
- Regularization
- ML pipeline
- Cost vs. loss functions
- AdaBoost

Other

- Varying opinions on pair programming
- Notes are helpful, most also take notes too
- Difficult if you didn’t take most recent version of CS260
- Overall course has been a lot of work
- Slow grading (I’m sorry! – Lab 2 is up now!)

Outline for Feb 29

- Recap fairness regularization
- Cost functions, different types of features/labels
- Gradient descent (Handout 11, Q3)
- ML pipeline and cross-validation
- Bias-variance tradeoff
- Ensembles and Boosting

Outline for Feb 29

- Recap fairness regularization
- Cost functions, different types of features/labels
- Gradient descent (Handout 11, Q3)
- ML pipeline and cross-validation
- Bias-variance tradeoff
- Ensembles and Boosting

SGD for logistic regression

- Hypothesis function (prediction)

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = p(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}\cdot\boldsymbol{x}}}$$

- Cost function (want to minimize)

$$J(\boldsymbol{w}) = -\sum_{i=1}^{n} y_i \log h_{\boldsymbol{w}}(\boldsymbol{x_i}) + (1-y_i)\log(1-h_{\boldsymbol{w}}(\boldsymbol{x_i}))$$

- Gradient of cost wrt single data point x_i

$$\nabla J_{\boldsymbol{x}_i}(\boldsymbol{w}) = (h_{\boldsymbol{w}}(\boldsymbol{x_i}) - y_i)\boldsymbol{x_i}$$

Lab 5 notation

sex

- `A = 1 (female, protected class)`
- `A = 0 (male, unprotected class)`

label

- `Y = 1 (income >= 50k)`
- `Y = 0 (income < 50k)`

Regularization ("default version")

- Regularization to minimize magnitude of weights

$$J^R(\vec{w}) = \underbrace{\left[-\sum_{i=1}^{n} y_i \log h(\vec{x}_i) + (1-y_i)\log(1+h(\vec{x}_i))\right]}_{\text{Original binary classification cost function}} + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{p} w_j^2}_{\text{Regularization term}}$$

Regularization ("default version")

- Regularization to minimize magnitude of weights

$$J^R(\vec{w}) = \underbrace{\left[-\sum_{i=1}^{n} y_i \log h(\vec{x}_i) + (1-y_i)\log(1+h(\vec{x}_i))\right]}_{\text{Original binary classification cost function}} + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{p} w_j^2}_{\text{Regularization term}}$$

- Take gradient of regularization term

$$\nabla J^R_{\vec{x}_i}(\vec{w}) = (h_{\vec{w}}(\vec{x}) - y_i)\vec{x}_i + \lambda\vec{w}^*$$

Regularization ("default version")

- Regularization to minimize magnitude of weights

$$J^R(\vec{w}) = \underbrace{\left[-\sum_{i=1}^{n} y_i \log h(\vec{x}_i) + (1-y_i)\log(1+h(\vec{x}_i))\right]}_{\text{Original binary classification cost function}} + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{p} w_j^2}_{\text{Regularization term}}$$

- Take gradient of regularization term

$$\nabla J^R_{\vec{x}_i}(\vec{w}) = (h_{\vec{w}}(\vec{x}) - y_i)\vec{x}_i + \lambda \vec{w}^*$$

- Note: don't regularize the bias term!

$$\vec{w}^* = \begin{bmatrix} 0 \\ w_1 \\ w_2 \\ . \\ . \\ . \\ w_p \end{bmatrix}$$

Regularization ("default version")

- Regularization to minimize magnitude of weights

$$J^R(\vec{w}) = \underbrace{\left[-\sum_{i=1}^{n} y_i \log h(\vec{x}_i) + (1-y_i)\log(1+h(\vec{x}_i))\right]}_{\text{Original binary classification cost function}} + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{p} w_j^2}_{\text{Regularization term}}$$

- Take gradient of regularization term

$$\nabla J^R_{\vec{x}_i}(\vec{w}) = (h_{\vec{w}}(\vec{x}) - y_i)\vec{x}_i + \lambda \vec{w}^*$$

- Note: don't regularize the bias term!

$$\vec{w}^* = \begin{bmatrix} 0 \\ w_1 \\ w_2 \\ . \\ . \\ . \\ w_p \end{bmatrix}$$

- Putting this all together: SGD

$$\vec{w} \leftarrow \vec{w} - \alpha\left[(h_{\vec{w}}(\vec{x}_i) - y_i)\vec{x}_i + \lambda \vec{w}^*\right]$$

Demographic Parity (DP) regularization

- Demographic parity definition

$$\boxed{DP = \frac{p(\hat{Y} = 1 | A = 1)}{p(\hat{Y} = 1 | A = 0}}$$

Demographic Parity (DP) regularization

- Demographic parity definition

$$\boxed{DP = \frac{p(\hat{Y} = 1 | A = 1)}{p(\hat{Y} = 1 | A = 0}}$$

- DP regularization term (note only includes the numerator)

$$\boxed{R(h_{\vec{w}}, D) = 1 - p(\hat{Y} = 1 | A = 1)}$$

Demographic Parity (DP) regularization

- Demographic parity definition

$$\boxed{DP = \frac{p(\hat{Y} = 1|A = 1)}{p(\hat{Y} = 1|A = 0}}$$

- DP regularization term (note only includes the numerator)

$$\boxed{R(h_{\vec{w}}, D) = 1 - p(\hat{Y} = 1|A = 1)}$$

- Add to the cost function

$$\boxed{J^R(\vec{w}) = J(\vec{w}) + R(h_{\vec{w}}, D)}$$

Demographic Parity (DP) regularization

- How do we compute the regularization term?

$$\boxed{1 - \frac{1}{|D_1|}\sum_{x \in D_1} p(\hat{y} = 1|\vec{x})}$$

Demographic Parity (DP) regularization

- How do we compute the regularization term?

$$\boxed{1 - \frac{1}{|D_1|} \sum_{x \in D_1} p(\hat{y} = 1 | \vec{x})}$$

- Logistic model tells us "prob pos"!

D_1 = # examples where A=1

$$\boxed{= 1 - \frac{1}{|D_1|} \sum_{\vec{x} \in D_1} h_{\vec{w}}(\vec{x})}$$

Demographic Parity (DP) regularization

- How do we compute the regularization term?

$$\boxed{1 - \frac{1}{|D_1|} \sum_{x \in D_1} p(\hat{y} = 1 | \vec{x})}$$

- Logistic model tells us "prob pos"!

D_1 = # examples where A=1

$$\boxed{= 1 - \frac{1}{|D_1|} \sum_{\vec{x} \in D_1} h_{\vec{w}}(\vec{x})}$$

- How to change the weight updates?

Gradient of the fairness regularization term

$$\vec{w} \leftarrow \begin{cases} \vec{w} - \alpha \Big[(h - y)\vec{x} \;\; - \frac{1}{|D_1|} \big(h_{\vec{w}}(\vec{x})(1 - h_{\vec{w}}(\vec{x}))\vec{x} \big) \Big] & \text{if } A = 1 \text{ for } x_i \\ \vec{w} - \alpha \Big[(h - y)\vec{x} \Big] & \text{if } A = 0 \text{ for } x_i \end{cases}$$

Error rate balance regularization

- Error rate balance definition

$$\boxed{\frac{p(\hat{Y} = 1 | A = 1, Y = y)}{p(\hat{Y} = 1 | A = 0, Y = y)} \text{ for } y \in \{0, 1\}}$$

Error rate balance regularization

- Error rate balance definition

$$\boxed{\frac{p(\hat{Y} = 1 | A = 1, Y = y)}{p(\hat{Y} = 1 | A = 0, Y = y)} \text{ for } y \in \{0, 1\}}$$

- For us we will just use the y=1 term (TP)

Error rate balance regularization

- Error rate balance definition

$$\boxed{\frac{p(\hat{Y}=1|A=1,Y=y)}{p(\hat{Y}=1|A=0,Y=y)} \text{ for } y \in \{0,1\}}$$

- For us we will just use the y=1 term (TP)
- Same idea but use D_1^1

D_1^1 = # examples where A=1 *and* Y=1

$$\boxed{= 1 - \frac{1}{|D_1^1|} \sum_{\vec{x} \in D_1^1} h_{\vec{w}}(\vec{x})}$$

Outline for Feb 29

- Recap fairness regularization
- Cost functions, different types of features/labels
- Gradient descent (Handout 11, Q3)
- ML pipeline and cross-validation
- Bias-variance tradeoff
- Ensembles and Boosting

Loss Functions

- E.g., zero-one loss
  - Simple accuracy - is prediction right?
  - For binary or multi-class prediction

$$l(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$$

- E.g., squared loss
  - For regression

$$l(y, \hat{y}) = (y - \hat{y})^2$$

- Absolute loss (also for regression)

$$\ell(y, \hat{y}) = |y - \hat{y}|$$

"log loss" (binary cross entropy)

$$\ell(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

Slide: modified from Ameet Soni

Loss function vs. cost function

- Often used interchangeably

- Loss function usually refers to idea for one example

- Cost function (usually) sums up loss for all examples, plus regularization terms etc

Continuous -> Discrete (Features)

(do this for the TRAIN only!)

X	Y
10	Y
7	Y
8	N
3	Y
7	N
12	Y
2	Y

1) Sort examples based on given feature

2	3	7	7	8	10	12
Y	Y	Y	N	N	Y	Y

Continuous -> Discrete (Features)

(do this for the TRAIN only!)

X	Y
10	Y
7	Y
8	N
3	Y
7	N
12	Y
2	Y

1) Sort examples based on given feature

2	3	7	7	8	10	12
Y	Y	Y	N	N	Y	Y

2) Different label with same feature value, collapse to "None"

2	3	7	8	10	12
Y	Y	None	N	Y	Y

Continuous -> Discrete (Features)

(do this for the TRAIN only!)

X	Y
10	Y
7	Y
8	N
3	Y
7	N
12	Y
2	Y

1) Sort examples based on given feature

2	3	7	7	8	10	12
Y	Y	Y	N	N	Y	Y

2) Different label with same feature value, collapse to "None"

2	3	7	8	10	12
Y	Y	None	N	Y	Y

3) Whenever label changes, make a feature (use avg)

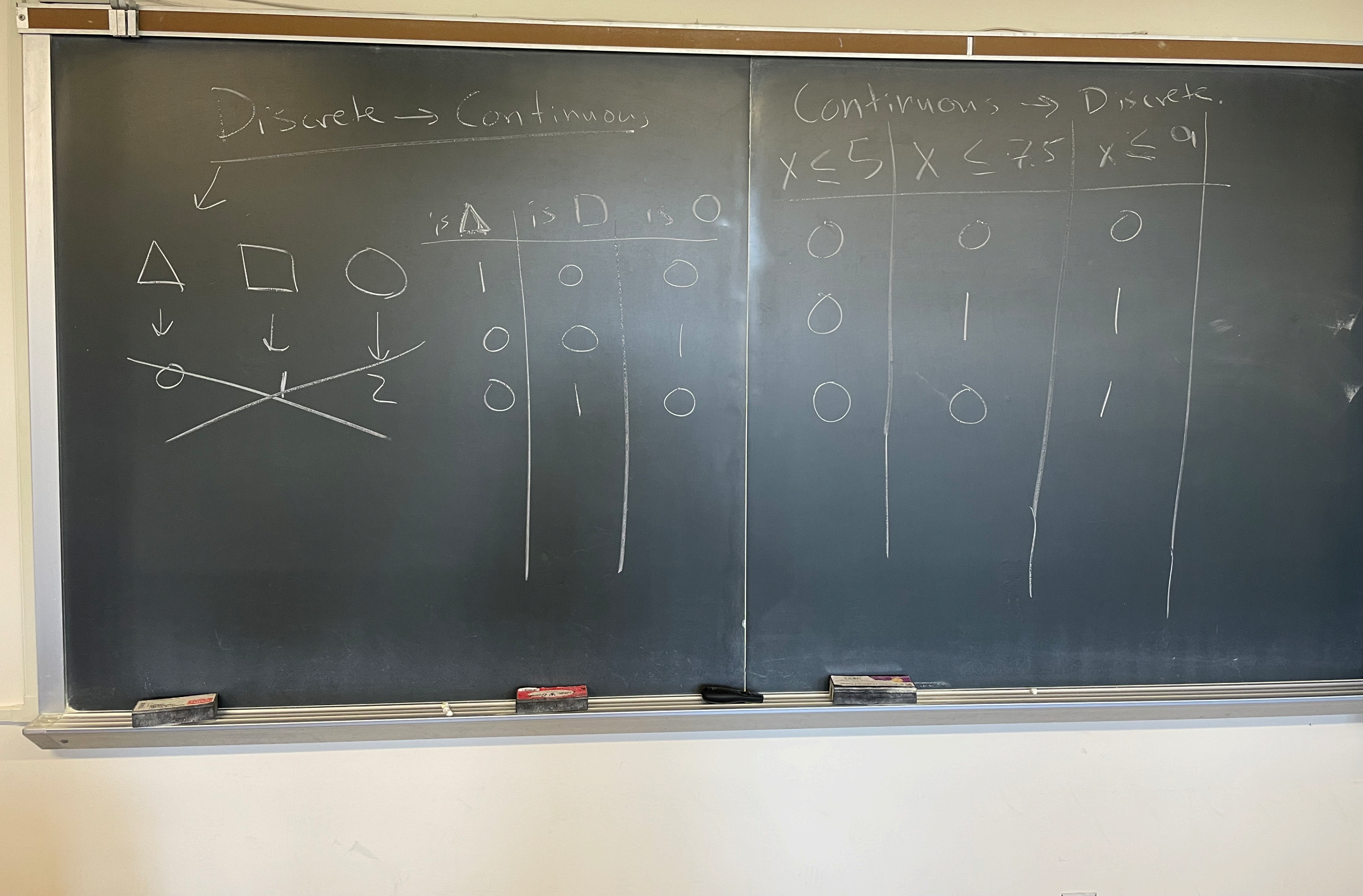
2	3	7	8	10	12
Y	Y	None	N	Y	Y

X <= 5

X <= 7.5

X <= 9

Discrete <-> Continuous (Features)



Outline for Feb 29

- Recap fairness regularization
- Cost functions, different types of features/labels
- Gradient descent (Handout 11, Q3)
- ML pipeline and cross-validation
- Bias-variance tradeoff
- Ensembles and Boosting

Handout 11, $\boxed{Q2}$

$\boxed{\text{goal}}$ $w = 3$

$\boxed{\alpha = 0.1}$

$$w \leftarrow w - \alpha(2w - 6)$$

$$0 - 0.1(0 - 6)$$

$$\boxed{w = 0.6}$$

$$w \leftarrow 0.6 - 0.1(2 \cdot 0.6 - 6)$$

$$0.6 + .48 = \boxed{1.08}$$

$$L(\vec{w}) = \prod_{i=1}^{n} h(x_i)^{y_i}\left(1-h(x_i)\right)^{1-y_i}$$

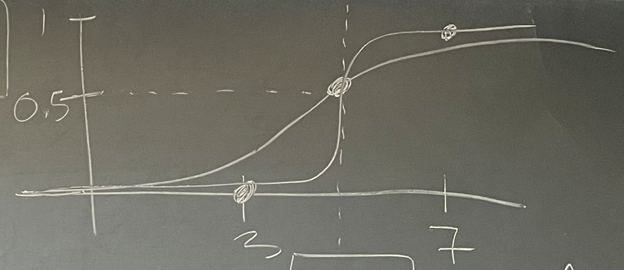
$$= \cancel{h(3)^{0}}\left(1-h(3)\right)^{1-0}$$

$$\cdot\, h(7)^{1}\cancel{\left(1-h(7)\right)^{1-1}}$$

$$= \left(1-h(3)\right)h(7)$$

$\boxed{Q3}$

(a)



$\boxed{x \geq 5} \Longrightarrow \hat{y} = 1$

(b)

$$L(\vec{w}) = \underbrace{\left(1-h_{\vec{w}}(3)\right)}_{\text{prob of 0 (high for } x_1)}\ \underbrace{h_{\vec{w}}(7)}_{\text{high for } x_2}$$

want high

(c) $\vec{w} \leftarrow \begin{bmatrix}0\\0\end{bmatrix} - 0.1\left(h_{\vec{w}}(7) - 1\right)\begin{bmatrix}1\\7\end{bmatrix}$ $\left.\begin{matrix}\leftarrow \text{fake } 1\\ \leftarrow x_2\end{matrix}\right\}\vec{X}_2$

$$= \begin{bmatrix}.05\\.35\end{bmatrix}$$

$$h_w(7) = \frac{1}{1+e^{-(0+7\cdot 0)}} = \frac{1}{1+1} = \boxed{0.5}$$

$$\vec{x}_1 = \begin{bmatrix}1\\3\end{bmatrix}$$

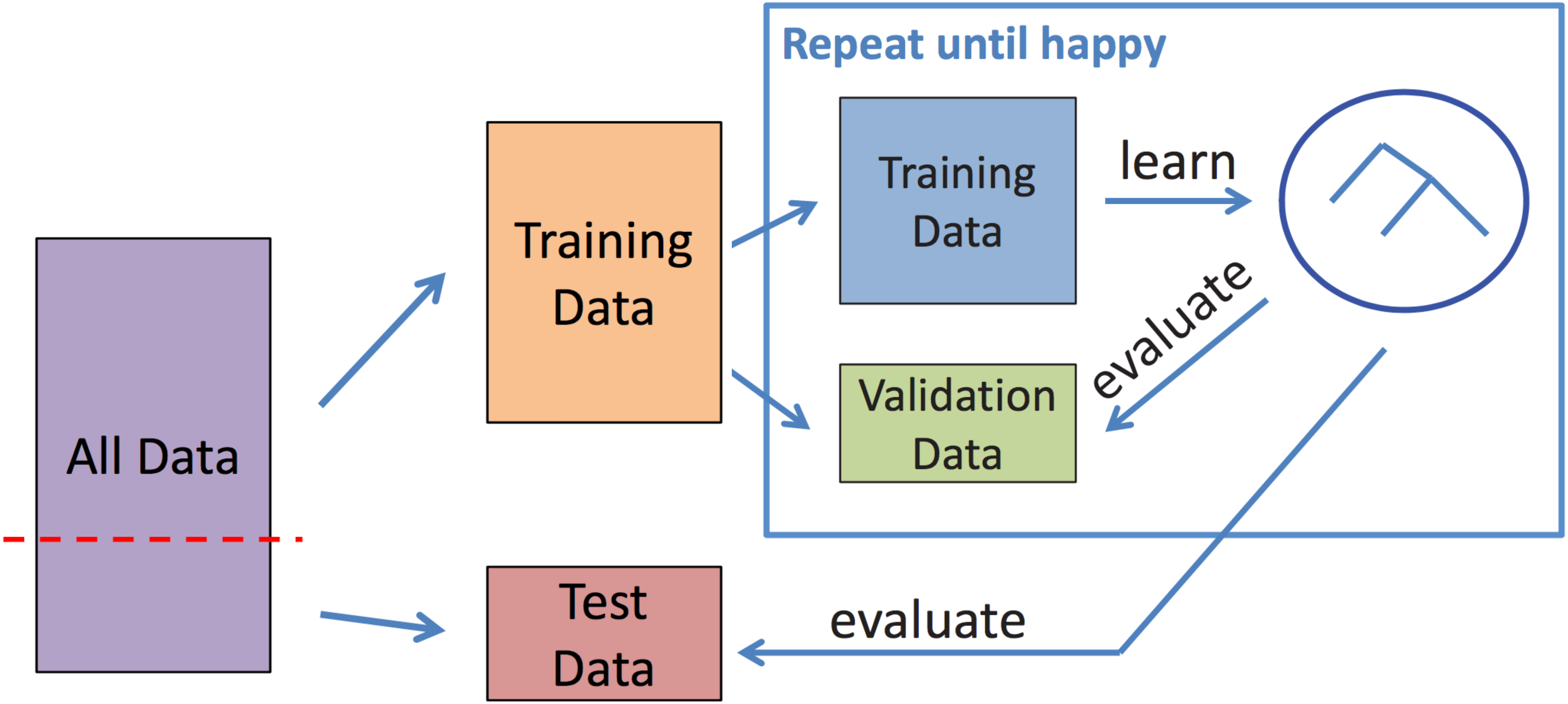
(d) $\vec{w} \leftarrow \begin{bmatrix}0.05\\0.35\end{bmatrix} - 0.1\left(h_{\vec{w}}(3) - 0\right)\begin{bmatrix}1\\3\end{bmatrix}$

$$\begin{bmatrix}0.05\\0.35\end{bmatrix}$$

Outline for Feb 29

- Recap fairness regularization
- Cost functions, different types of features/labels
- Gradient descent (Handout 11, Q3)
- ML pipeline and cross-validation
- Bias-variance tradeoff
- Ensembles and Boosting

Better: use a *validation* dataset



Modified from Jessica Wu

k-fold Cross Validation

Evaluation

Accuracy(P_1) *= 11/20*

Accuracy(P_2) *= 17/20*

Accuracy(P_3) *= 16/20*

Accuracy(P_4) *= 13/20*

Accuracy(P_5) *= 16/20*

Generalization: average accuracy across all folds = 73/100 = 73%

Slide: Ameet Soni

sklearn example of cross-validation

```python
from sklearn.model_selection import cross_val_score

tree_rmses = -cross_val_score(tree_reg, housing, housing_labels,
                              scoring="neg_root_mean_squared_error", cv=10)
```

count	10.000000
mean	66868.027288
std	2060.966425
min	63649.536493
25%	65338.078316
50%	66801.953094
75%	68229.934454
max	70094.778246

count	10.000000
mean	47019.561281
std	1033.957120
min	45458.112527
25%	46464.031184
50%	46967.596354
75%	47325.694987
max	49243.765795

```python
from sklearn.ensemble import RandomForestRegressor

forest_reg = make_pipeline(preprocessing,
                           RandomForestRegressor(random_state=42))
forest_rmses = -cross_val_score(forest_reg, housing, housing_labels,
                                scoring="neg_root_mean_squared_error", cv=10)
```

Finding hyper-parameters

- Grid search
- Random search

```python
from sklearn.model_selection import GridSearchCV

full_pipeline = Pipeline([
    ("preprocessing", preprocessing),
    ("random_forest", RandomForestRegressor(random_state=42)),
])
param_grid = [
    {'preprocessing__geo__n_clusters': [5, 8, 10],
     'random_forest__max_features': [4, 6, 8]},
    {'preprocessing__geo__n_clusters': [10, 15],
     'random_forest__max_features': [6, 8, 10]},
]
grid_search = GridSearchCV(full_pipeline, param_grid, cv=3,
                           scoring='neg_root_mean_squared_error')
grid_search.fit(housing, housing_labels)
```

n_clusters	max_features	split0	split1	split2	mean_test_rmse
15	6	43460	43919	44748	44042
15	8	44132	44075	45010	44406
15	10	44374	44286	45316	44659
10	6	44683	44655	45657	44999
10	6	44683	44655	45657	44999

Bias-Variance

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\underbrace{\hat{y}_i}_{\hat{f}(x)} - y_i)^2$$

$$E[MSE] = \underbrace{E[(\hat{f}-f)^2]}_{\text{reducible error}} + \underbrace{Var(\varepsilon)}_{\text{irreducible error}}$$

$$= \boxed{\underbrace{\underbrace{bias(\hat{f})^2}\ +\ \underbrace{Var(\hat{f})}}_{\text{bias-variance tradeoff}} + Var(\varepsilon)}$$

$$\underbrace{y}_{\text{true}} = \underbrace{f(x)}_{\text{"true model"}} + \underbrace{\varepsilon}_{\text{error}}$$

Handout 12, Q1

as model complexity ↑

$$\boxed{\begin{array}{l} \text{bias} \downarrow \\ \text{Variance} \uparrow \end{array}}$$

Handout 12, Q1