A Review, then Sliding into Classification

Kenneth (Kenny) Joseph

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Announcements

- PA2 due Sunday night
- Quiz 4 is out, we will review Quiz 3 today
- PA 3 is out March 7th
  - Minimal coding, we'll be annotating data, calculating agreement statistics, and reflecting on the process
  - You have almost a month to do this (in other words, a break from programming assignments for a while
- Midterm is March 17th
  - In class, mostly
  - One page handwritten notes, front and back
  - Official Accessibility requests **due by next Tuesday**
- Questions?

Terms/concepts you now know/have seen

- "Review"
  - Probability distribution
  - Expected Values
  - Stats (e.g. mean/variance)
  - Python
  - Pandas/Numpy/Jupyter
- ML High-level ideas
  - Model class
  - Loss function
    - Squared Error
    - Regularization
  - Optimization algorithm
    - (Stochastic) Gradient Descent
    - Closed form solutions
  - Making Predictions
- Models
  - (Regularized) Linear Regression
  - Polynomial regression
  - Decision Tree Regression
  - kNN regression
  - ~~(Generalized) additive models~~
- Selection & Evaluation
  - 2/3-way holdout methods
  - K-fold cross validation
  - Bias/Variance tradeoff
  - Generalization error
  - The 3 sources of error
  - Over/underfitting

This week

- PA 1, Quiz 3
- A brief review of where we're at
  - Supervised learning – what's the point?
  - Where do features come from?
  - What does sklearn.linear_model.LinearRegression() actually do?
- A "new" setup from a *probabilistic* perspective
  - Maximum Likelihood Estimation
  - Using the probabilistic approach to re-derive OLS regression
- Intro to classification
  - Logistic Regression
  - Bayes Optimal Classifier
  - Naïve Bayes
- Potentially: SVMs & Kernels

Back to the beginning

- In Supervised ML, we have…

(x_i, y_i) where $x \in \mathbb{R}^d \leftarrow$ "features" $y \leftarrow$ label

$D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in \mathbb{R}^d \times \mathcal{C}$

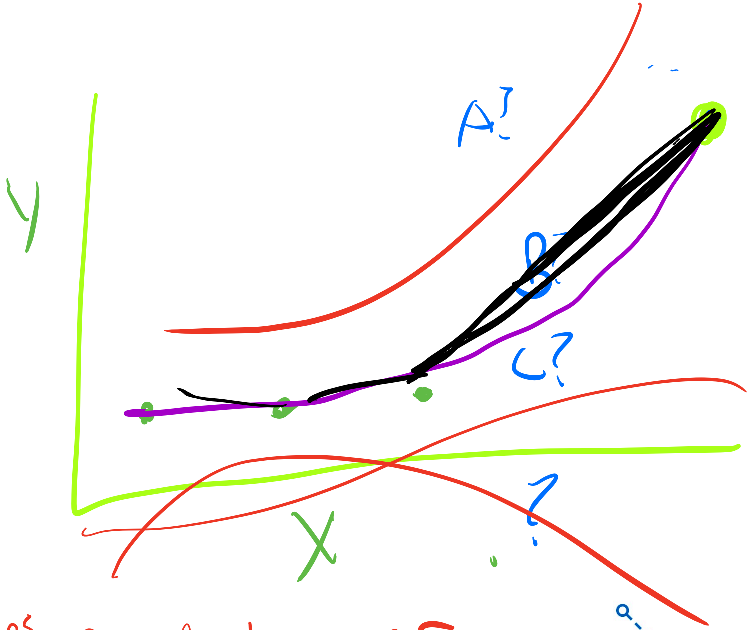
- We want to be able to get y when we only have **x**…

$h(x_i) \rightarrow y_i$

How do we find the “best model?

- We could just memorize the training data … right?
  - Training != Test, curse of dimensionality
- So… make assumptions (def’n model class)
- Then, find best model… 3 steps
  - Define best
  - Find best
  - Select/evaluate

minimizes Generalization error



- Linear Regression as an example…

How do we find the "best model?

- We could just memorize the training data … right?
  - Training != Test, curse of dimensionality
- So… make assumptions (def'n model class)
- Then, find best model… 3 steps
  - Define best
  - Find best
  - Select/evaluat
- Linear Regression as an example…

How do we find the "best model?

- We could just memorize the training data … right?
  - Training != Test, curse of dimensionality
- So… make assumptions (def'n model class)
- Then, find best model… 3 steps
  - Define best
  - Find best
  - Select/evaluat
- Linear Regression as an example…

LR

Model class? $f(x) = w^T x$

Loss fn? SSE $(w^T x - y)^2$

Optimize? GD / closed form

Selection/evaluation? RMSE

How do we find the "best model?

- We could just memorize the training data … right?
  - Training != Test, curse of dimensionality
- So… make assumptions (def'n model class)
- Then, find best model… 3 steps
  - Define best
  - Find best
  - Select/evaluat
- Linear Regression as an example…

LinearRegression().fit(X_train, y_train)

1 model class

2 Loss fn

3 optimization fn

Re-introducing probability…

- Some holes in this “optimization” story…
  - What was all that business about “expectations”?
  - What about “training data as a random sample”? Of what?
  - Why SSE?
- Remembering the **probabilistic** part…

$$(x_i, y_i) \sim P(X, Y)$$

Implications of probabilistic framing

- Goal changes slightly – find h(x) approx. y
- Re-specifying "the best we can do"…
- Re-explaining "training data as a random sample"…
- But what about the SSE optimization part?

$$h(x) \approx y$$

$$E_{(x,y)\sim p}\left[L(x,y \mid h^*(\cdot)) \right]$$

$$(x_i, y_i) \sim p$$

Maximum Likelihood estimation

- P(coin) is heads, when D={H,T,T,H,H,H,T,T,T,T}?
- More formal derivation?
- Use **MLE**
  - Specify parameterized distribution
  - Find parameters that make observed data most likely
- For coin toss…

$P(H) = \frac{n_H}{n_H + n_T} = .4$

$P(D|\theta) = \binom{n_H + n_T}{n_H} \theta^{n_H} (1-\theta)^{n_T}$

$\hat{\theta} = \underset{\theta}{\mathrm{argmax}}\, P(D;\theta)$

$\hat{\theta} = \underset{\theta}{\mathrm{argmax}}$

Example from: https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote06.html

Linking back to the optimization view

- If we have a good model of the distrn. from which (**x**, y) is drawn, we can use it to put forward a good guess as to E[Y | X=x]
- MLE gets us the best estimate of this probability distribution, **given a particular parameterized form**…
- Actually, two kinds of models
  - Generative
  - Discriminative

$(\mathbf{x}, y) \sim P$

$P(y \mid x)$

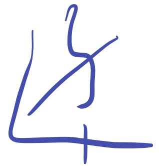
$P(x, y)$

$E_{p(x,y)}[Y \mid X=x]$

A "new strategy" for ML

1. Define p(y | **x**) in terms of some parameterized model "model class"
2. Write down (log) likelihood function "loss fn."
3. Find the parameters that maximize the probability of the observed data

$$p(y_i | x_i) \sim N(w^T x, 1)$$



$$= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i^T w - y_i)^2}{2\sigma^2}}$$

likelihood fn: $\prod p(y_i | x_i; w)$

$\underset{w}{\text{argmax}}$

$\log \prod p(y_i | x_i; w)$

$\sum \log p(y_i, x_i; w)$

Trying our "new strategy" for linear regression

$$\underset{w}{\operatorname{argmax}} \sum \log p(y_i \mid x_i, w)$$

$$\underbrace{\sum \log\left(\frac{1}{\sqrt{2\pi\sigma}}\right)} + \log\left(e^{-\dots}\right)$$

$$\sum \log\left(e^{-\frac{(x_i^T w - y_i)^2}{2\sigma^2}}\right)$$

$$\underset{w}{\operatorname{argmax}} - \sum_i (x^T w - y_i)^2$$

$$\underset{w}{\operatorname{argmin}} \sum (x_i^T w - y_i)^2$$

See Sections 1 and 2 of https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote08.html for the full derivation!