Machine Learning (CSE 446): Probabilistic Machine Learning MLE & MAP

Sham M Kakade

University of Washington
cse446-staff@cs.washington.edu

Announcements

- Homeworks
  - HW 3 posted. Get the most recent version.
  - You must do the regular probs before obtaining *any* extra credit.
  - Extra credit factored in after your scores are averaged together.
- Office hours today: 3-4p
- Today:
  - Review
  - Probabilistic methods

Review

SGD: How do we set the step sizes?

- Theory: If you turn down the step sizes at (some prescribed decaying method) then SGD will converge to the right answer.
  The "classical" theory doesn't provide enough practical guidance.
- Practice:
  - starting stepsize: start it "large":
    if it is "too large", then either you diverge (or nothing improves). set it a little less (like $1/4$) less than this point.
  - When do we decay it?
    When your training error stops decreasing "enough".
- HW: you'll need to tune it a little. (a slow approach: sometimes you can just start it somewhat smaller than the "divergent" value and you will find something reasonable.)

SGD: How do we set the mini-batch size m?

- Theory: there are diminishing returns to increasing m.
- Practice: just keep cranking it up and eventually you'll see that your code doesn't get any faster.

Regularization: How do we set it?

- Theory: really just says that λ controls your "model complexity".
  - we DO know that "early stopping" for GD/SGD is (basically) doing L2 regularization for us
  - i.e. if we don't run for too long, then $\|\mathbf{w}\|^2$ won't become too big.
- Practice:
  - Set with a dev set!
  - Exact methods (like matrix inverse/least squares): always need to regularize or something horrible happens....
  - GD/SGD: sometimes (often ?) it works just fine ignoring regularization

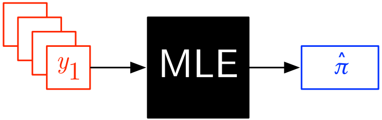
Today

There is no magic in vector derivatives: scratch space

There is no magic in vector derivatives: scratch space

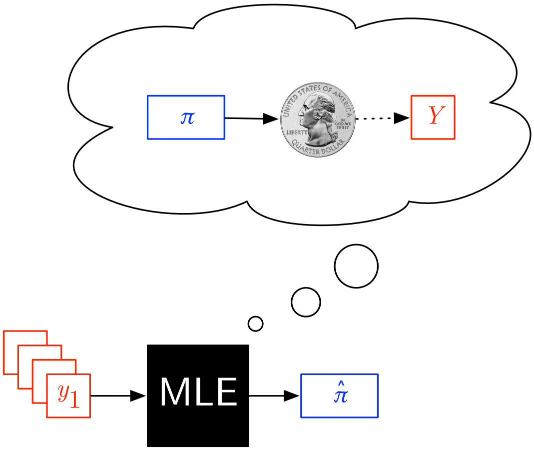
There is no magic in matrix derivatives: scratch space

Understanding MLE

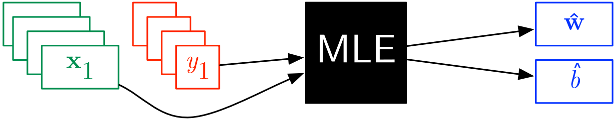


You can think of MLE as a "black box" for choosing parameter values.

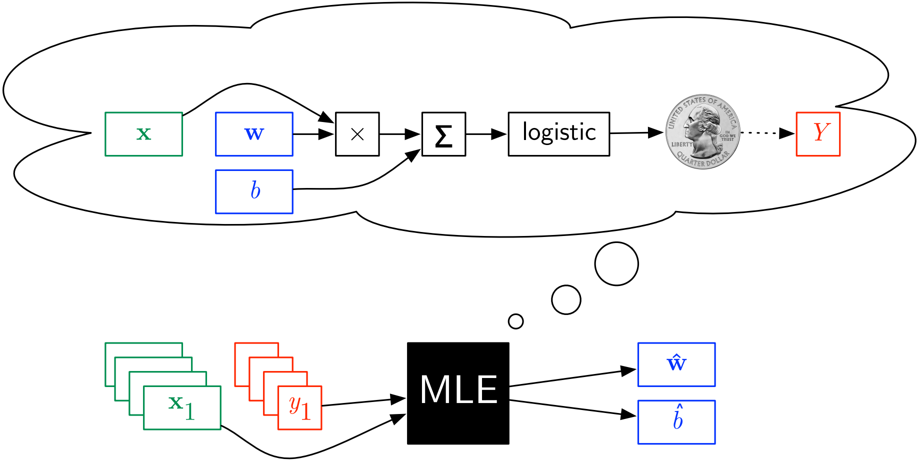
Understanding MLE



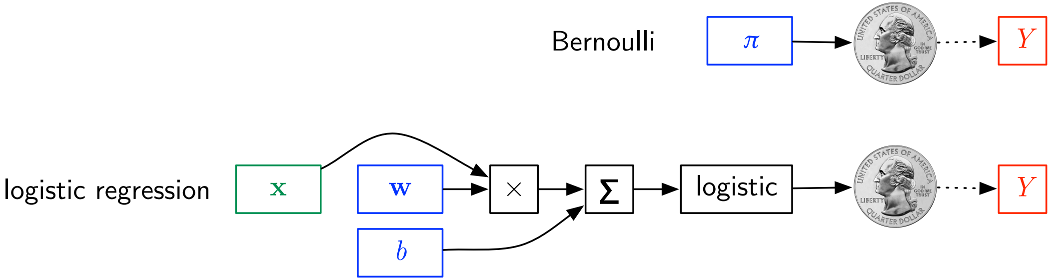
Understanding MLE



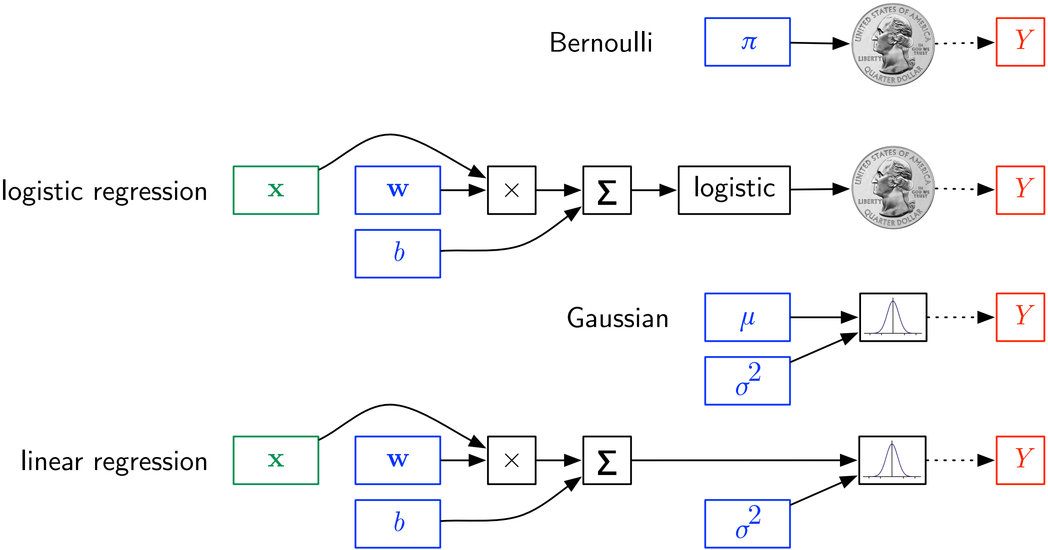
Understanding MLE



Probabilistic Stories



Probabilistic Stories



MLE example: estimating the bias of a coin

MLE example: estimating the bias of a coin

Then and Now

Before today, you knew how to do MLE:

- For a Bernoulli distribution: $\hat{\pi} = \frac{\text{count}(+1)}{\text{count}(+1)+\text{count}(-1)} = \frac{N^+}{N}$
- For a Gaussian distribution: $\hat{\mu} = \frac{\sum_{n=1}^{N} y_n}{N}$ (and similar for estimating variance, $\hat{\sigma}^2$).

Logistic regression and linear regression, respectively, generalize these so that the parameter is itself a function of $\mathbf{x}$, so that we have a **conditional model** of Y given X.

- The practical difference is that the MLE doesn't have a closed form for these models.
  (So we use SGD and friends.)

Remember: Linear Regression as a Probabilistic Model

Linear regression defines $p_{\mathbf{w}}(Y \mid X)$ as follows:

1. Observe the feature vector $\mathbf{x}$; transform it via the activation function:

$$\mu = \mathbf{w} \cdot \mathbf{x}$$

2. Let μ be the mean of a normal distribution and define the density:

$$p_{\mathbf{w}}(Y \mid \mathbf{x}) = \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(Y - \mu)^2}{2\sigma^2}$$

3. Sample Y from $p_{\mathbf{w}}(Y \mid \mathbf{x})$.

Remember: Linear Regression-MLE is (Unregularized) Squared Loss Minimization!

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} -\log p_{\mathbf{w}}(y_n \mid \mathbf{x}_n) \equiv \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \underbrace{(y_n - \mathbf{w} \cdot \mathbf{x}_n)^2}_{SquaredLoss_n(\mathbf{w},b)}$$

Where did the variance go?

Adding a "Prior" to the Probabilistic Story

Probabilistic story:

- For $n \in \{1, \ldots, N\}$:
  - Observe $\mathbf{x}_n$.
  - Transform it using parameters $\mathbf{w}$ to get $p(Y = y \mid \mathbf{x}_n, \mathbf{w})$.
  - Sample $y_n \sim p(Y \mid \mathbf{x}_n, \mathbf{w})$.

Adding a “Prior” to the Probabilistic Story

Probabilistic story:

- For $n \in \{1, \ldots, N\}$:
  - Observe $\mathbf{x}_n$.
  - Transform it using parameters $\mathbf{w}$ to get $p(Y = y \mid \mathbf{x}_n, \mathbf{w})$.
  - Sample $y_n \sim p(Y \mid \mathbf{x}_n, \mathbf{w})$.

Probabilistic story with a “prior”:

- Use hyperparameters α to define a **prior** distribution over random variables W, $p_\alpha(\boldsymbol{W})$.
- Sample $\mathbf{w} \sim p_\alpha(W = w)$.
- For $n \in \{1, \ldots, N\}$:
  - Observe $\mathbf{x}_n$.
  - Transform it using parameters $\mathbf{w}$ and b to get $p(Y \mid \mathbf{x}_n, \mathbf{w})$.
  - Sample $y_n \sim p(Y \mid \mathbf{x}_n, \mathbf{w})$.

MLE vs. Maximum a Posteriori (MAP) Estimation

- Review: MLE
  - We have a model $\Pr(\text{Data}|\mathbf{w})$.
  - Find $\mathbf{w}$ which maximizes the probability of the data you have observed:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \Pr(\text{Data}|\mathbf{w})$$

- New: Maximum a Posterior Estimation
  - Also have a **prior** $\Pr(W = \mathbf{w})$
  - Now we a have **posterior** distribution:

$$\Pr(\mathbf{w}|\text{Data}) = \frac{\Pr(\text{Data}|\mathbf{w}) \Pr(W = \mathbf{w})}{\Pr(\text{Data})}$$

  - Now suppose we are asked to provide our "best guess" at $\mathbf{w}$. What should we do?

Maximum a Posteriori (MAP) Estimation and Regularization

- MAP estimation:

$$\operatorname*{argmax}_{\mathbf{w}} \Pr(\mathbf{w} \mid \text{Data})$$

- In many settings, this leads to

$$(\hat{\mathbf{w}}) = \operatorname*{argmax}_{\mathbf{w}} \underbrace{\log p_{\alpha}(\mathbf{w})}_{\text{log prior}} + \underbrace{\sum_{n=1}^{N} \log p_{\mathbf{w}}(y_n \mid \mathbf{x}_n)}_{\text{log likelihood}}$$

Maximum a Posteriori (MAP) Estimation and Regularization

- MAP estimation:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \Pr(\mathbf{w} \mid \text{Data})$$

- In many settings, this leads to

$$(\hat{\mathbf{w}}) = \underset{\mathbf{w}}{\operatorname{argmax}} \underbrace{\log p_\alpha(\mathbf{w})}_{\text{log prior}} + \underbrace{\sum_{n=1}^{N} \log p_{\mathbf{w}}(y_n \mid \mathbf{x}_n)}_{\text{log likelihood}}$$

Option 1: let $p_\alpha(W)$ be a zero-mean Gaussian distribution with standard deviation α.

$$\log p_\alpha(\mathbf{w}) = -\frac{1}{2\alpha^2}\|\mathbf{w}\|_2^2 + \text{constant}$$

Maximum a Posteriori (MAP) Estimation and Regularization

- MAP estimation:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \Pr(\mathbf{w} \mid \text{Data})$$

- In many settings, this leads to

$$(\hat{\mathbf{w}}) = \underset{\mathbf{w}}{\operatorname{argmax}} \underbrace{\log p_\alpha(\mathbf{w})}_{\text{log prior}} + \underbrace{\sum_{n=1}^{N} \log p_{\mathbf{w}}(y_n \mid \mathbf{x}_n)}_{\text{log likelihood}}$$

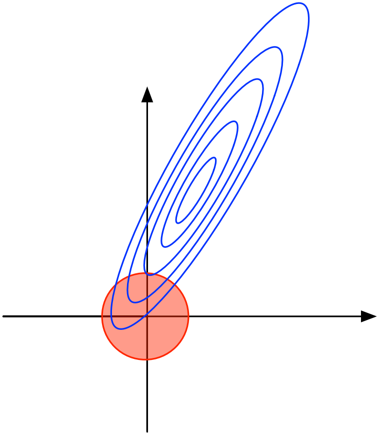
Option 1: let $p_\alpha(W)$ be a zero-mean Gaussian distribution with standard deviation α.

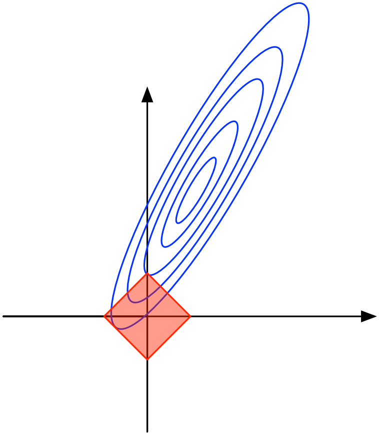
$$\log p_\alpha(\mathbf{w}) = -\frac{1}{2\alpha^2}\|\mathbf{w}\|_2^2 + \text{constant}$$

Option 2: let $p_\alpha(W_j)$ be a zero-location "Laplace" distribution with scale α.

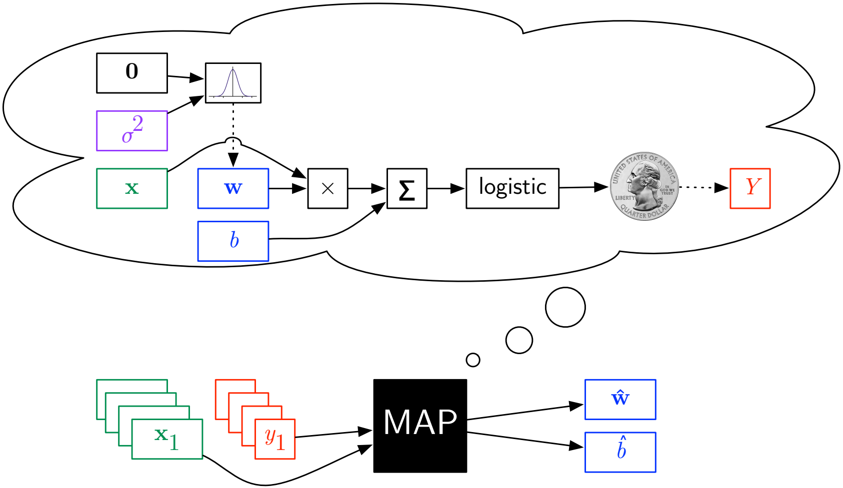
$$\log p_\alpha(\mathbf{w}) = -\frac{1}{\alpha}\|\mathbf{w}\|_1 + \text{constant}$$

L_2 v.s. L_1-Regularization





Probabilistic Story: L_2-Regularized Logistic Regression



Why Go Probabilistic?

- Interpret the classifier's activation function as a (log) probability (density), which encodes uncertainty.
- Interpret the regularizer as a (log) probability (density), which encodes uncertainty.
- Leverage theory from statistics to get a better understanding of the guarantees we can hope for with our learning algorithms.
- Change your assumptions, turn the optimization-crank, and get a new machine learning method.

The key to success is to tell a probabilistic story that's reasonably close to reality, including the prior(s).