Lecture 2
Machine Learning Review
CMSC 35246: Deep Learning

Shubhendu Trivedi
&
Risi Kondor

University of Chicago

March 29, 2017

- Things we will look at today
  - Formal Setup for Supervised Learning
  - Empirical Risk, Risk, Generalization
  - Define and derive a linear model for Regression
  - Revise Regularization
  - Define and derive a linear model for Classification
  - (Time permitting) Start with Feedforward Networks

Note: Most slides in this presentation are adapted from, or taken (with permission) from slides by Professor Gregory Shakhnarovich for his TTIC 31020 course

What is Machine Learning?

- Right question: What is learning?
- Tom Mitchell ("Machine Learning", 1997): "A Computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"
- Gregory Shakhnarovich: Make predictions and pay the price if the predictions are incorrect. Goal of learning is to reduce the price.
- How can you specify T, P and E?

Formal Setup (Supervised)

- Input data space $\mathcal{X}$
- Output (label) space $\mathcal{Y}$
- Unknown function $f : \mathcal{X} \to \mathcal{Y}$
- We have a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ with $\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}$
- Finite $\mathcal{Y} \implies$ Classification
- Continuous $\mathcal{Y} \implies$ Regression

Regression

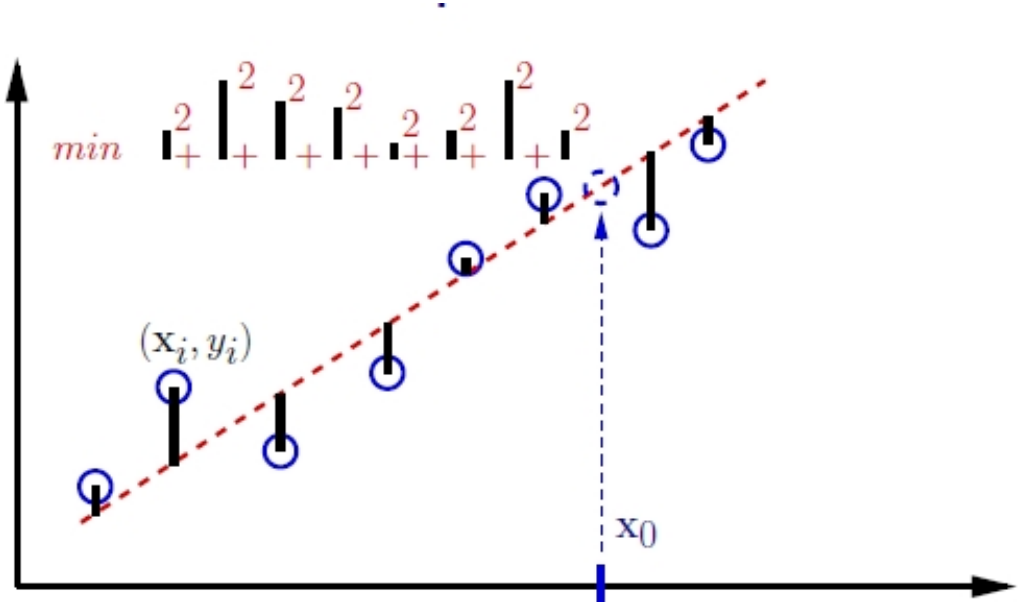
- We are given a set of N observations $(\mathbf{x}_i, y_i)$ with $i = 1, \ldots, N$ with $y_i \in \mathbb{R}$
- Example: Measurements (possibly noisy) of barometric pressure x versus liquid boiling point y
- Does it make sense to use learning here?

Fitting Function to Data

- We will approach this in two steps:
  - Choose a *model class* of functions
  - Design a criteria to guide the selection of one function from the selected class
- Let us begin with considering one of the simpled model classes: linear functions

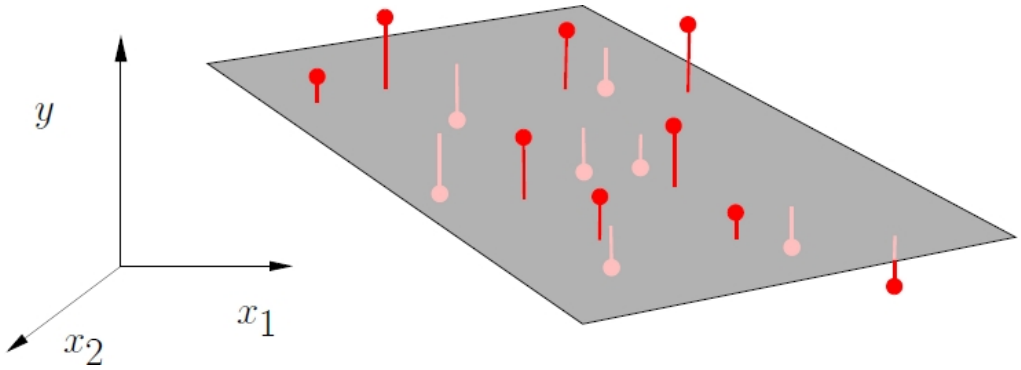
Linear Fitting to Data

- We want to fit a linear function to $(X, Y) = \{(\mathbf{x}_1, y_1) \dots (\mathbf{x}_N, y_N)\}$
- Fitting criteria: Least squares. Find the function that minimizes the sum of squared distances between actual ys in the training set
- We then use the fitted line as a predictor



Linear Functions

- General form: $f(\mathbf{x}; \theta) = \theta_0 + \theta_1 x_1 + \ldots \theta_d x_d$
- 1-D case: A line
- $\mathcal{X} \in \mathbb{R}^2$: a plane
- *Hyperplane* in general d-D case



Loss Functions

- Targets are in $\mathcal{Y}$
  - Binary Classification: $\mathcal{Y} = \{-1, +1\}$
  - Univariate Regression: $\mathcal{Y} \equiv \mathbb{R}$
- A **Loss Function** $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
- L maps decisions to costs. $L(\hat{y}, y)$ is the penalty for predicting $\hat{y}$ when the *correct* answer is y
- Standard choice for classification: 0/1 loss
- Standard choice for regression: $L(\hat{y}, y) = (\hat{y} - y)^2$

Empirical Loss

- Consider a *parametric* function $f(\mathbf{x}; \theta)$
- Example: Linear function - $f(\mathbf{x}; \theta) = \theta_0 + \sum_{j=1}^{d} \theta_j x_{ij} = \theta^T \mathbf{x}$
- Note: $x_{i0} \equiv 1$
- The *empirical loss* of function $y = f(\mathbf{x}; \theta)$ on a set $\mathbf{X}$:

$$L(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i; \theta), y_i)$$

- Least squares minimizes empirical loss for squared loss L
- We care about: *predicting* labels for *new* examples
- When does empirical loss minimization help us in doing that?

Loss: Empirical and Expected

- Basic Assumption: Example and label pairs $(\mathbf{x}, y)$ are drawn from an unknown distribution $p(\mathbf{x}, y)$
- Data are i.i.d: Same (abeit unknown) distribution for all $(\mathbf{x}, y)$ in both training and test data
- The empirical loss is measured on the training set:

$$L(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i; \theta), y_i)$$

- The **goal** is to minimize the *expected loss*, also known as **risk**:

$$R(\theta) = \mathbb{E}_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} [L(f(\mathbf{x}_0; \theta), y_0)]$$

Empirical Risk Minimization

- Empirical Loss:

$$L(\theta, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i; \theta), y_i)$$

- Risk:

$$R(\theta) = \mathbb{E}_{(\mathbf{x}_0, y_0) \sim p(\mathbf{x}, y)} [L(f(\mathbf{x}_0; \theta), y_0)]$$

- Empirical Risk Minimization: If the training set is a representative of the underlying (unknown) distribution $p(\mathbf{x}, y)$, the empirical loss is a proxy for the risk
- In essence: Estimate $p(\mathbf{x}, y)$ by the *empirical distribution* of the data

Learning via Empirical Loss Minimization

- Learning is done in two steps:
  - Select a restricted class $\mathcal{H}$ of *hypotheses* $f : \mathcal{X} \to \mathcal{Y}$
    Example: linear functions parameterized by θ:
    $f(\mathbf{x}, y) = \theta^T \mathbf{x}$
  - Select a hypothesis $f^* \in \mathcal{H}$ based on the training set $\mathcal{D} = (X, Y)$
    Example: minimize empirical squared loss. That is, select $f(\mathbf{x}, \theta^*)$ such that:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} (y_i - \theta^T \mathbf{x}_i)^2$$

- How do we find $\theta^* = [\theta_0, \theta_1, \ldots, \theta_d]$?

Least Squares: Estimation

- Necessary condition to minimize L: $\frac{\partial L(\theta)}{\partial \theta_0}, \frac{\partial L(\theta)}{\partial \theta_1}, \ldots, \frac{\partial L(\theta)}{\partial \theta_d}$ must be set to zero
- Gives us $d+1$ linear equations in $d+1$ unknowns $\theta_0, \theta_1, \ldots, \theta_d$
- Let us switch to vector notation for convenience

Learning via Empirical Loss Minimization

- First let us write down least squares in matrix form:
  Predictions: $\hat{\mathbf{y}} = X\theta$; errors: $\mathbf{y} - X\theta$; empirical loss:

$$\begin{aligned} L(\theta, X) &= \frac{1}{N}(\mathbf{y} - X\theta)^T(\mathbf{y} - X\theta) \\ &= (\mathbf{y}^T - \theta^T X^T)(\mathbf{y} - X\theta) \end{aligned}$$

- What next? Take derivative of $L(\theta)$ and set it to zero

Derivative of Loss

- $L(\theta) = \frac{1}{N}(\mathbf{y}^T - \theta^T X^T)(\mathbf{y} - X\theta)$
- Use identities $\frac{\partial \mathbf{a}^T \mathbf{b}}{\partial \mathbf{a}} = \frac{\partial \mathbf{b}^T \mathbf{a}}{\partial \mathbf{a}} = \mathbf{b}$ and $\frac{\partial \mathbf{a}^T B \mathbf{a}}{\partial \mathbf{a}} = 2B\mathbf{a}$

$$\begin{aligned}
\frac{\partial L(\theta)}{\partial \theta} &= \frac{1}{N}\frac{\partial}{\partial \theta}[\mathbf{y}^T\mathbf{y} - \theta^T X^T \mathbf{y} - \mathbf{y}^T X\theta + \theta^T X^T X\theta] \\
&= \frac{1}{N}[0 - X^T\mathbf{y} - (\mathbf{y}^T X)^T + 2X^T X\theta] \\
&= -\frac{2}{N}(X^T\mathbf{y} - X^T X\theta)
\end{aligned}$$

Least Squares Solution

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{N}(X^T\mathbf{y} - X^T X\theta) = 0$$

$$X^T\mathbf{y} = X^T X\theta \implies \theta^* = (X^T X)^{-1}X^T\mathbf{y}$$

- $X^\dagger = (X^T X)^{-1}X^T$ is the Moore-Penrose pseudoinverse of X
- Linear regression infact has a closed form solution!
- Prediction: $\hat{y} = \theta^{*^T}\begin{bmatrix}1\\ \mathbf{x}_0\end{bmatrix} = \mathbf{y}^T X^{\dagger^T}\begin{bmatrix}1\\ \mathbf{x}_0\end{bmatrix}$

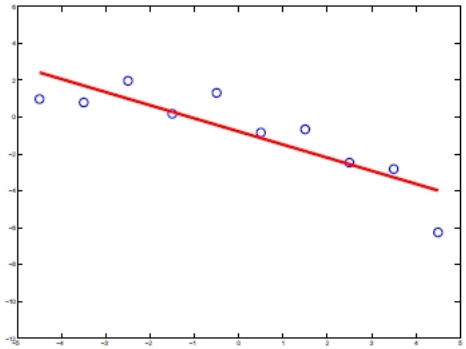
Polynomial Regression

- Transform $\mathbf{x} \rightarrow \phi(\mathbf{x})$
- For example consider 1D for simplicity:
- $f(x;\theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_m x^m$
- Above $\phi(\mathbf{x}) = [1, x, x^2, \ldots, x^m]$
- No longer linear in x, but still linear in θ!
- Back to familiar linear regression!
- Generalized Linear models:
  $f(\mathbf{x};\theta) = \theta_0 + \theta_1 \phi_1(\mathbf{x}) + \theta_2 \phi_2(\mathbf{x}) + \cdots + \theta_m \phi_m(\mathbf{x})$

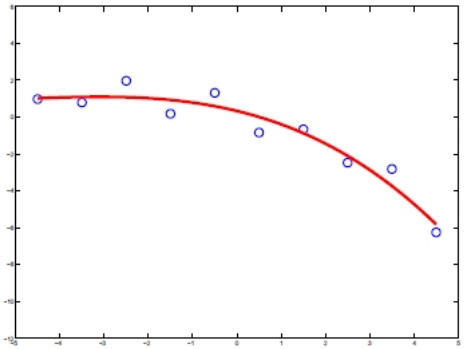
A Short Primer on Regularization

Model Complexity and Overfitting

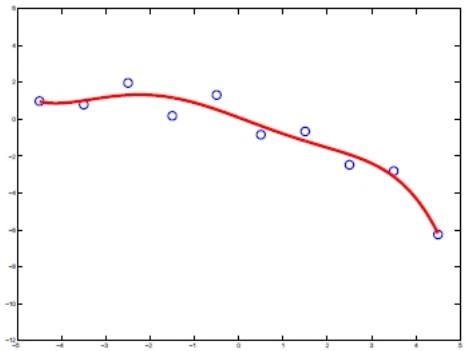
Consider data drawn from a 3rd order model:



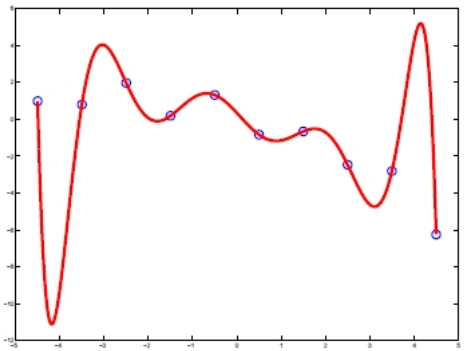
$m = 1$



$m = 3$



$m = 5$

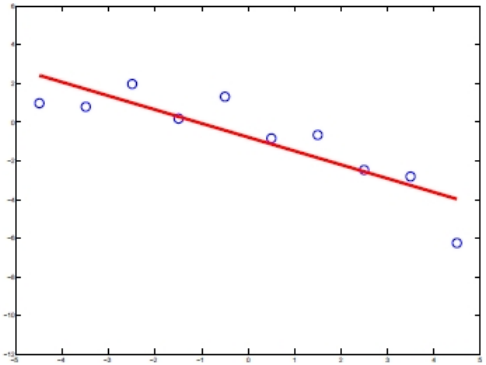


$m = 10$

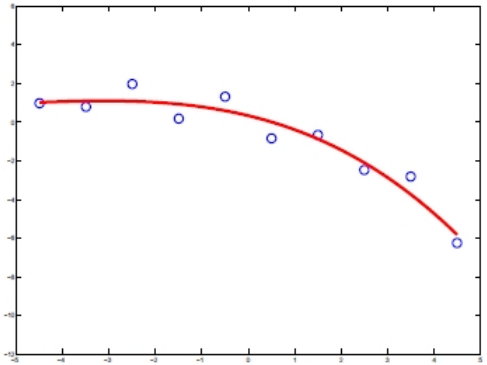
Avoiding Overfitting: Cross Validation

- If model overfits i.e. it is too sensitive to data: It will be unstable
- Idea: *hold out* part of the data, fit model on rest and test on held out set
- k-fold cross validation. Extreme case: *leave one out* cross validation
- What is the source of overfitting?

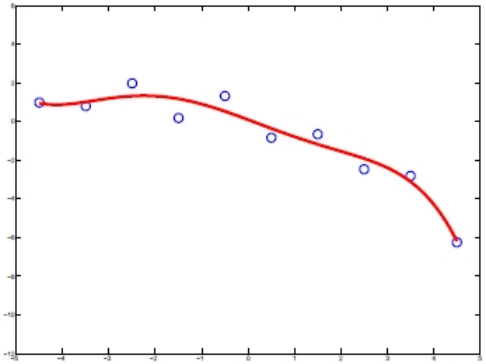
Cross Validation Example



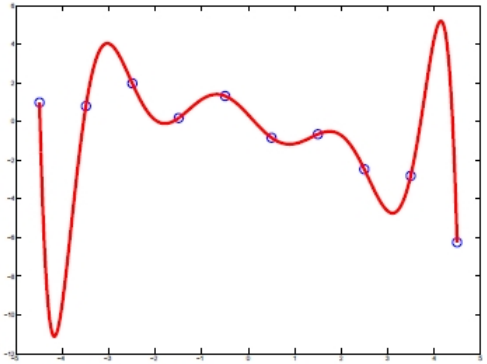
$m = 1 : L = 1.4, \hat{L}_{\text{cv}} = 2.6$



$m = 3 : L = 0.4, \hat{L}_{\text{cv}} = 1.3$



$m = 5 : L = 0.3, \hat{L}_{\text{cv}} = 2.7$



$m = 10 : L = 0, \hat{L}_{\text{cv}} = 4 \times 10^4$

Model Complexity

- Model complexity is the number of *independent* parameters to be fit ("degrees of freedom")
- Complex model $\implies$ more sensitive to data $\implies$ more likely to overfit
- Simple model $\implies$ more rigid $\implies$ more likely to underfit
- Find the model with the right "bias-variance" balance

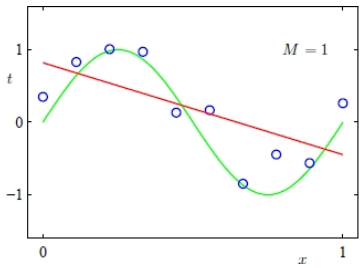
Penalizing Model Complexity

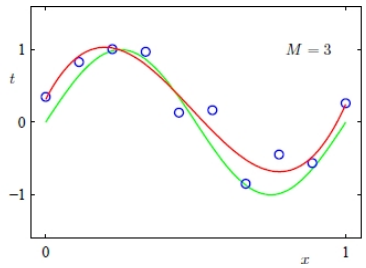
- Idea 1: Restrict model complexity based on amount of data
- Idea 2: Directly penalize by the number of parameters (called the Akaike Information criterion): minimize

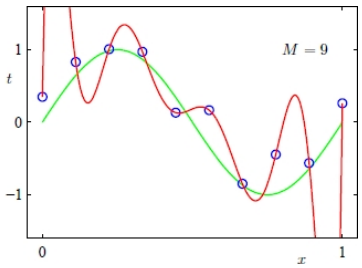
$$\sum_{i=1}^{N} L(f(x_i;\theta), y_i) + \#\text{params}$$

- Since the parameters might not be independent, we would like to penalize the complexity in a more sophisticated way

Problems







	$m = 0$	$m = 1$	$m = 3$	$m = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43

Description Length

- Intuition: Should not penalize the parameters, but the number of bits needed to encode the parameters
- With a finite set of parameter values, these are equivalent. With an infinite set, we can limit the effective number of degrees of freedom by restricting the value of the parameters.
- Then we can have Regularized Risk minimization:

$$\sum_{i=1}^{N} L(f(x_i;\theta), y_i) + \Omega(\theta)$$

- We can measure "size" in different ways: L1, L2 norms etc. etc.
- Regularization is basically a way to implement Occam's Razor

Shrinkage Regression

- Shrinkage methods: We penalize the L2 norm

$$\theta^*_{ridge} = \arg\min_{\theta} \sum_{i=1}^{N} L(f(x_i;\theta), y_i) + \lambda \sum_{j=1}^{m} (\theta_j)^2$$

- If we use likelihood:

$$\theta^*_{ridge} = \arg\max_{\theta} \sum_{i=1}^{N} \log p(\text{data}_i;\theta) - \lambda \sum_{j=1}^{m} (\theta_j)^2$$

- This is called Ridge regression: Closed form solution for squared loss $\hat{\theta}_{ridge} = (\lambda I + X^T X)^{-1} X^T \mathbf{y}$!

LASSO Regression

- LASSO: We penalize the L1 norm

$$\theta^*_{lasso} = \arg\min_{\theta} \sum_{i=1}^{N} L(f(x_i;\theta), y_i) + \lambda \sum_{j=1}^{m} |\theta_j|$$

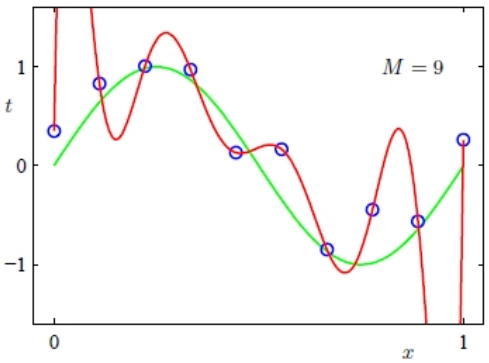
- If we use likelihood:

$$\theta^*_{ridge} = \arg\max_{\theta} \sum_{i=1}^{N} \log p(\text{data}_i;\theta) - \lambda \sum_{j=1}^{m} |\theta_j|$$

- This is called LASSO regression: No closed form solution!
- Still convex, but no longer smooth. Solve using Lagrange multipliers!

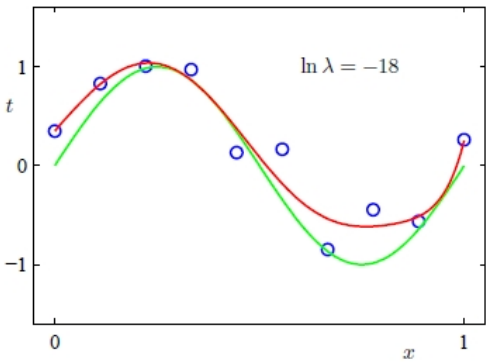
Effect of λ

$\lambda = 0$



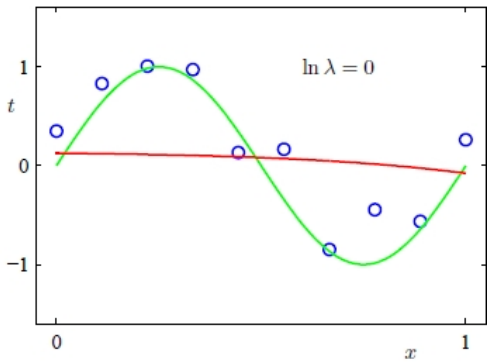
$\|\mathbf{w}^*\|^2 > 10^{12}$

$\lambda = e^{-18}$



$\|\mathbf{w}^*\|^2 \approx 21595$

$\lambda = 1$



$\|\mathbf{w}^*\|^2 \approx 0.027$

The Principle of Maximum Likelihood

- Suppose we have N data points $X = \{x_1, x_2, \ldots, x_N\}$ (or $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$)
- Suppose we know the probability distribution function that describes the data $p(x; \theta)$ (or $p(y|x; \theta)$)
- Suppose we want to determine the parameter(s) θ
- Pick θ so as to *explain* your data best
- What does this mean?
- Suppose we had two parameter values (or vectors) θ_1 and θ_2.
- Now suppose you were to *pretend* that θ_1 was really the true value parameterizing p. What would be the probability that you would get the dataset that you have? Call this $P1$
- If $P1$ is very small, it means that such a dataset is very unlikely to occur, thus perhaps θ_1 was not a good guess

The Principle of Maximum Likelihood

- We want to pick θ_{ML} i.e. the best value of θ that explains the data you *have*
- The plausibility of given data is measured by the "likelihood function" $p(x;\theta)$
- Maximum Likelihood principle thus suggests we pick θ that maximizes the likelihood function
- The procedure:
  - Write the log likelihood function: $\log p(x;\theta)$ (we'll see later why log)
  - Want to maximize - So differentiate $\log p(x;\theta)$ w.r.t θ and set to zero
  - Solve for θ that satisfies the equation. This is θ_{ML}

The Principle of Maximum Likelihood

- As an aside: Sometimes we have an initial guess for θ BEFORE seeing the data
- We then use the data to *refine* our guess of θ using Bayes Theorem
- This is called MAP (Maximum a posteriori) estimation (we'll see an example)
- Advantages of ML Estimation:
  - Cookbook, "turn the crank" method
  - "Optimal" for large data sizes
- Disadvantages of ML Estimation
  - Not optimal for small sample sizes
  - Can be computationally challenging (numerical methods)

Linear Classifiers

$$\hat{y} = h(\mathbf{x}) = \text{sign}(\theta_0 + \theta^T \mathbf{x})$$

- We need to find the (direction) θ and (the location) θ_0
- Want to minimize the expected 0/1 loss for classifier $h : \mathcal{X} \to \mathcal{Y}$

$$L(h(\mathbf{x}), y) = \begin{cases} 0, & \text{if } h(\mathbf{x}) = y \\ 1, & \text{if } h(\mathbf{x}) \neq y \end{cases}$$

Risk of a Classifier

- The risk (expected loss) of a C-way classifier $h(\mathbf{x})$

$$\begin{aligned} R(\mathbf{x}) &= \mathbb{E}_{\mathbf{x},y}[L(h(\mathbf{x}), y)] \\ &= \int_{\mathbf{x}} \sum_{c=1}^{C} L(h(\mathbf{x}), c) p(\mathbf{x}, y = c) d\mathbf{x} \\ &= \int_{\mathbf{x}} \left[\sum_{c=1}^{C} L(h(\mathbf{x}), c) p(y = c|\mathbf{x}) \right] p(\mathbf{x}) d\mathbf{x} \end{aligned}$$

- Clearly, it suffices to minimize the conditional risk:

$$R(h|\mathbf{x}) = \sum_{c=1}^{C} L(h(\mathbf{x}), c) p(y = c|\mathbf{x})$$

Conditional Risk of a Classifier

$$
\begin{aligned}
R(h|\mathbf{x}) &= \sum_{c=1}^{C} L(h(\mathbf{x}), c) p(y = c|\mathbf{x}) \\
&= 0 \times p(y = h(\mathbf{x})|\mathbf{x}) + 1 \times \sum_{c \neq h(\mathbf{x})} p(y = c|\mathbf{x}) \\
&= \sum_{c \neq h(\mathbf{x})} p(y = c|\mathbf{x}) = 1 - p(y = h(\mathbf{x})|\mathbf{x})
\end{aligned}
$$

- To minimize the conditional risk given $\mathbf{x}$, the classifier must decide

$$h(\mathbf{x}) = \arg\max_{c} p(y = c|\mathbf{x})$$

Log Odds Ratio

- Optimal rule $h(\mathbf{x}) = \arg\max_c p(y = c|\mathbf{x})$ is equivalent to:

$$h(\mathbf{x}) = c^* \iff \frac{p(y = c^*|\mathbf{x})}{p(y = c|\mathbf{x})} \geq 1 \forall c$$

$$\iff \log \frac{p(y = c^*|\mathbf{x})}{p(y = c|\mathbf{x})} \geq 0 \forall c$$

- For the binary case:

$$h(\mathbf{x}) = 1 \iff \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} \geq 0$$

The Logistic Model

- The unknown decision boundary can be modeled directly:

$$\frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \theta_0 + \theta^T \mathbf{x} = 0$$

- Since $p(y=1|\mathbf{x}) = 1 - p(y=0|\mathbf{x})$, exponentiating, we have:

$$\begin{aligned}
\frac{p(y=1|\mathbf{x})}{1-p(y=1|\mathbf{x})} &= \exp(\theta_0 + \theta^T \mathbf{x}) = 1 \\
\implies \frac{1}{p(y=1|\mathbf{x})} &= 1 + \exp(-\theta_0 - \theta^T \mathbf{x}) = 2 \\
\implies p(y=1|\mathbf{x}) &= \frac{1}{1+\exp(-\theta_0 - \theta^T \mathbf{x})} = \frac{1}{2}
\end{aligned}$$

The Logistic Function

$$p(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\theta_0 - \theta^T \mathbf{x})}$$

- Properties?
- With linear logistic model we get a linear decision boundary

Likelihood under the Logistic Model

$$p(y_i|\mathbf{x};\theta) = \begin{cases} \sigma(\theta_0 + \theta^T \mathbf{x}_i) \text{ if } y_i = 1 \\ 1 - \sigma(\theta_0 + \theta^T \mathbf{x}_i) \text{ if } y_i = 0 \end{cases}$$

- We can rewrite this as:

$$p(y_i|\mathbf{x};\theta) = \sigma(\theta_0 + \theta^T \mathbf{x}_i)^{y_i} (1 - \sigma(\theta_0 + \theta^T \mathbf{x}_i))^{1-y_i}$$

- The log-likelihood of θ:

$$\log p(Y|X;\theta) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i;\theta)$$

$$= \sum_{i=1}^{N} y_i \log \sigma(\theta_0 + \theta^T \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\theta_0 + \theta^T \mathbf{x}_i))$$

The Maximum Likelihood Solution

$$\log p(Y|X;\theta) = \sum_{i=1}^{N} y_i \log \sigma(\theta_0 + \theta^T \mathbf{x}_i) + (1-y_i) \log(1-\sigma(\theta_0 + \theta^T \mathbf{x}_i))$$

- Setting derivatives to zero:

$$\frac{\partial \log p(Y|X;\theta)}{\partial \theta_0} = \sum_{i=1}^{N} (y_i - \sigma(\theta_0 + \theta^T \mathbf{x}_i)) = 0$$

$$\frac{\partial \log p(Y|X;\theta)}{\partial \theta_j} = \sum_{i=1}^{N} (y_i - \sigma(\theta_0 + \theta^T \mathbf{x}_i)) \mathbf{x}_{i,j} = 0$$

- Can treat $y_i - p(y_i|\mathbf{x}_i) = y_i - \sigma(\theta_0 + \theta^T \mathbf{x}_i)$ as the prediction error

Finding Maxima

- No closed form solution for the Maximum Likelihood for this model!
- But $\log p(Y|X;\mathbf{x})$ is jointly concave in all components of θ
- Or, equivalently, the error is convex
- Gradient Descent/ascent (descent on $-\log p(y|\mathbf{x};\theta)$, log loss)

Next time

- Feedforward Networks
- Backpropagation