CS 189/289

Today's lecture:

- Finish last lecture: Intro to ML
- Maximum likelihood estimation (MLE)

Announcement: for now, my office hours will be outside Haas at an outdoor table, after leaving main lecture hall exit. If rains, will try to find a spot indoors in Haas.

Assigned readings

Reminder: Lecture 1 Intro: 1-1.2.4

Today’s lecture:

- 2-2.1.2 – Rules of probability, sum, product
- 2.1.6 – Independent RVs
- 2.2-2.2.1 – Probability densities in continuous spaces.
- 2.3-2.3.2- Univariate Gaussian, Likelihood,
- 3-3.1.3- Discrete RV, Bernoulli, Binomial, Multinomial, MLE

CS 189/289

Today's lecture:

- Finish overall Intro to ML
- Maximum likelihood estimation (MLE)

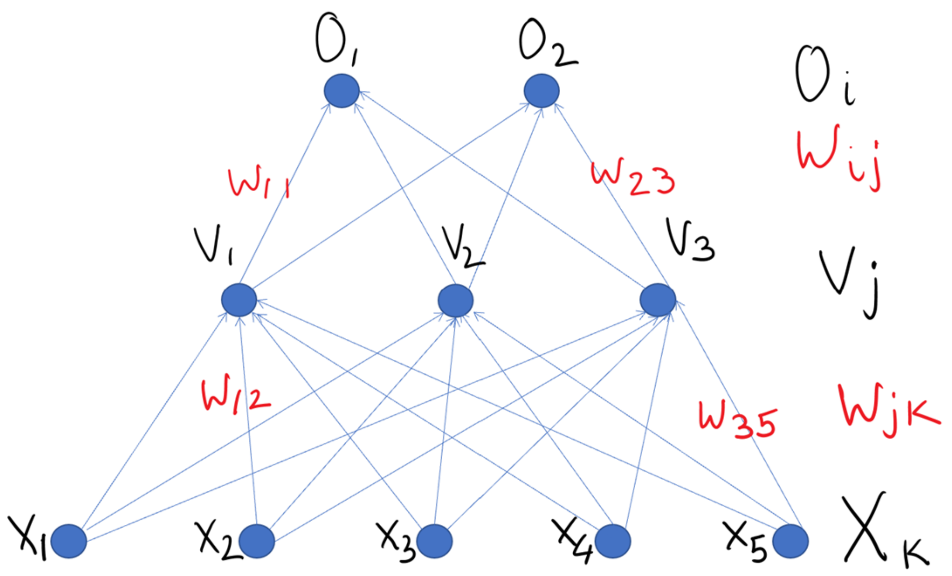
Training a neural network

Goal: Find W such that O_i is as close as possible to y_i (desired output)

Approach: • Define loss function $\mathcal{L}(W)$

• Compute $\nabla_W \mathcal{L}$

• $W_{new} \leftarrow W_{old} - \eta \nabla_W \mathcal{L}$



Training a single-layer neural network

- A good choice of loss function is the likelihood, (equivalent to cross- entropy).

$$\mathcal{L} = -\sum_{\text{input data}} \left(y_i \ln O_i + (1-y_i)\ln(1-O_i) \right)$$

- Model the activation function g as a sigmoid

$$g(z) = \frac{1}{1+\exp(-z)}$$

- Finding parameter w reduces to logistic regression!

We use *gradient descent.* $w_{new} \leftarrow w_{old} - \eta \nabla_w \mathcal{L}$

Training a 2-layer neural network

Use the same loss function, same activation functions, and still use gradient descent. All the same principles apply!

- Compute gradient wrt all weights across all layers.
- Loss function is no longer convex, so we typically find local minima.
- Time complexity of computing the gradient is naively quadratic in the # of weights.
- The *back-propagation algorithm* is a trick that enables it to be computed in linear time.
- This works for *any* number of layers.

What is the “best” classifier?

- If we knew the true probability distribution of the features conditioned on the classes, there is a “correct” answer – the *Bayes classifier*.
- The Bayes classifier minimizes the probability of misclassification.
- (unknown in practical situations)

Bayes Optimal Classifier

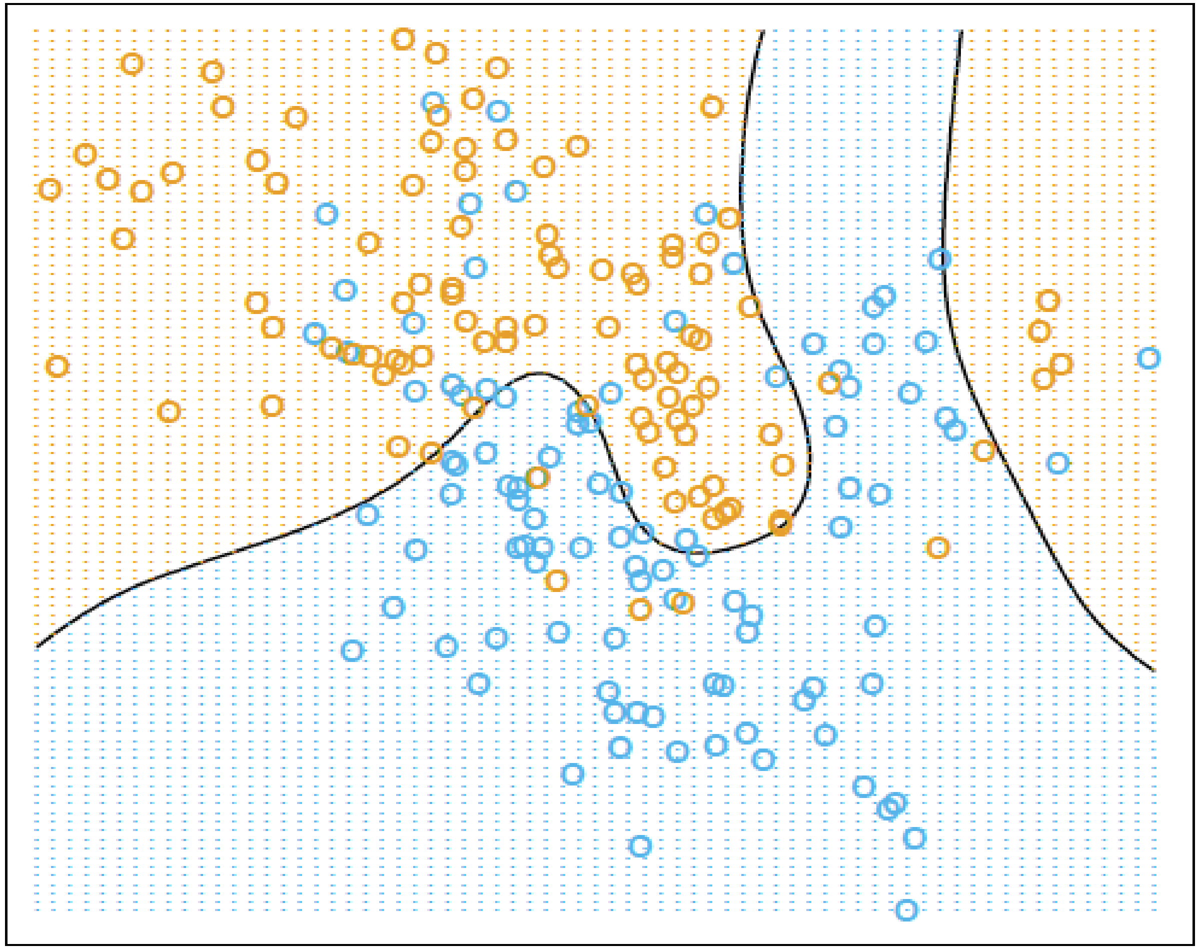


FIGURE 2.5. *The optimal Bayes decision boundary for the simulation example of Figures 2.1, 2.2 and 2.3. Since the generating density is known for each class, this boundary can be calculated exactly (Exercise 2.2).*

[Source : Hastie, Tibshirani, Friedman]

Two primary kinds of error

1. Training set error
   - We train a classifier to minimize training set error.
2. Test set error
   - At test time, we will take the trained classifier and use it to classify previously unseen examples for which we know the right answer. The error on these is called test set error.

Validation and Cross-Validation

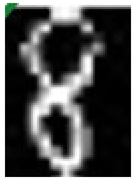
- If the test set error is much greater than training set error, this is called over-fitting.
- To avoid over-fitting, we can measure error on a held-out set of training data, called the validation set.
- We could divide the data into k-folds, use k-1 of these to train and "test"/validate on the remaining fold. This is cross-validation.

CS 189/289

Today's lecture:

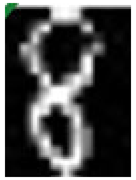
- Finish overall Intro to ML
- Maximum likelihood estimation (MLE)

Recall from last class:

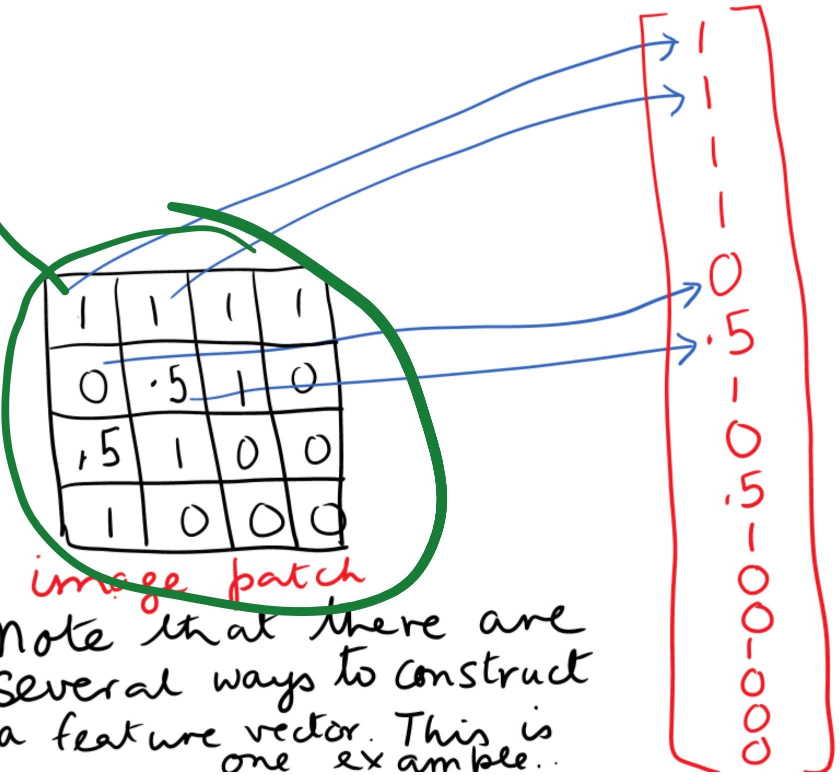
Problem of digit classification from handwriting: is  a “7”, yes or no?

- 60K training examples of digits (6K per class)
- Each digit is a 28 x 28 pixel grey level image.

Recall from last class:

Problem of digit classification from handwriting: is  a "7", yes or no?

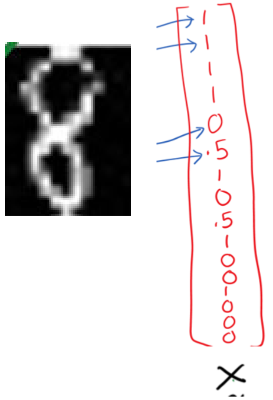
- 60K training examples of digits (6K per class)
- Each digit is a 28 x 28 pixel grey level image.

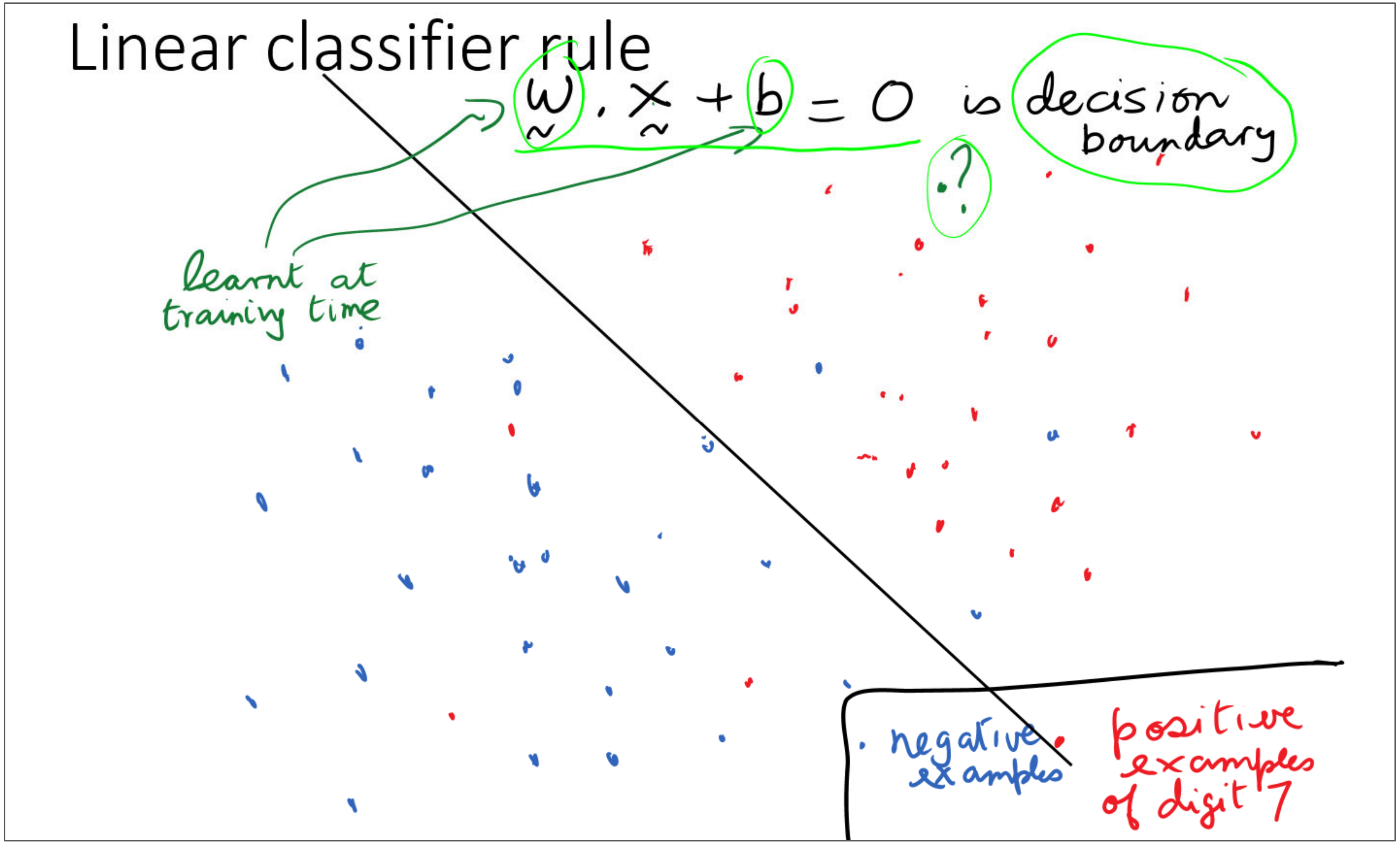


Note that there are several ways to construct a feature vector. This is one example.

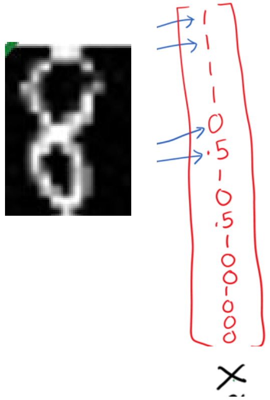
X

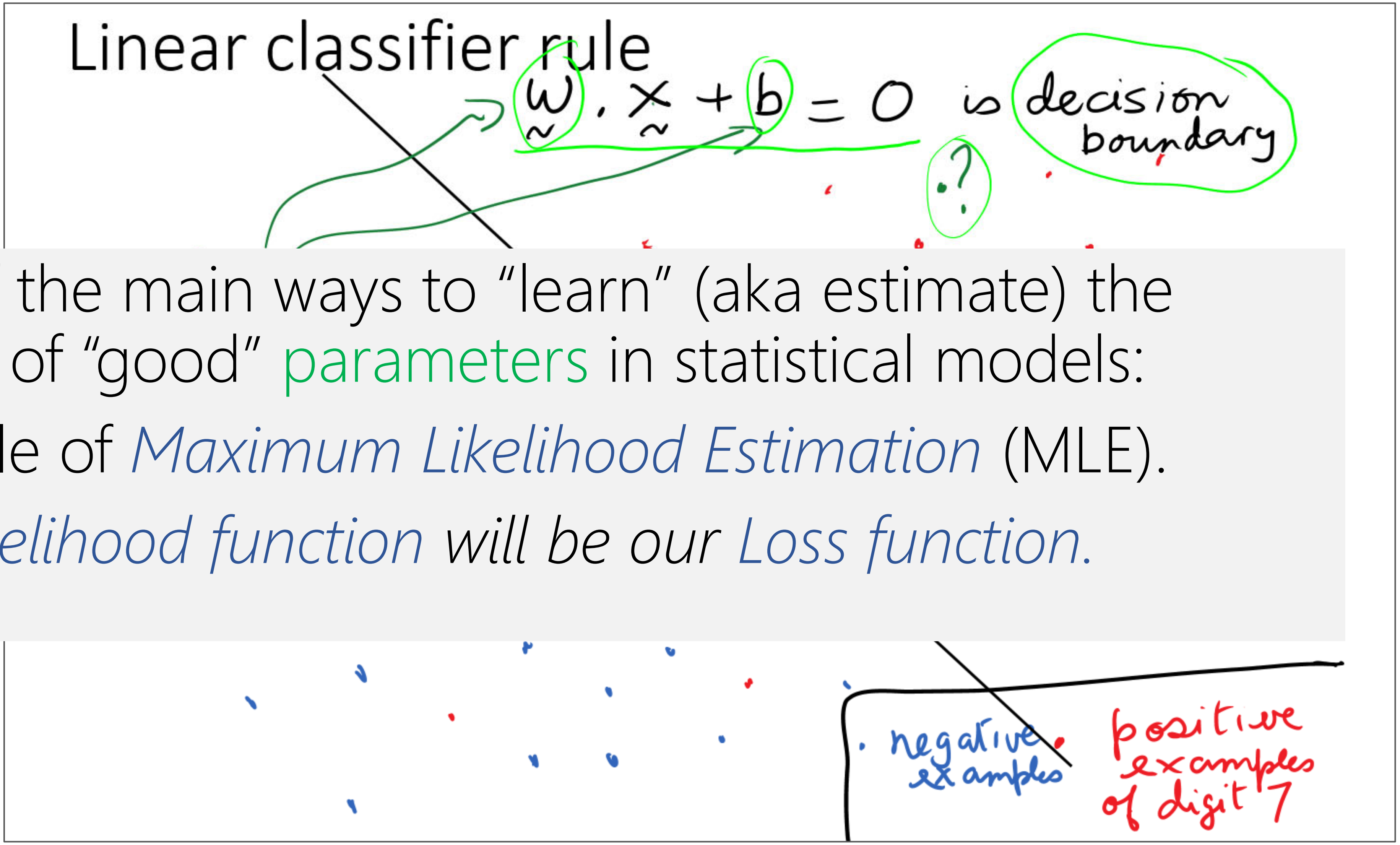
Recall from last class:





Recall from last class:





- One of the main ways to "learn" (aka estimate) the setting of "good" parameters in statistical models:
- Principle of *Maximum Likelihood Estimation* (MLE).
- *The Likelihood function will be our Loss function.*

ML: main concepts

- Training data set: $D = \{(x_i, y_i)\}_{i=1}^{N}$

$x_i \in R^D$

$y_i \in R$ or $y_i \in \{-1,1\}$

classification

regression

ML: main abstract ideas

- Training data set: $D = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x_i \in R^D$ $\quad y_i \in R$ or $y_i \in \{-1,1\}$

SUPERVISED

"label" provides supervision

$D = \{(x_i\ )\}_{i=1}^{N}$ $\quad x_i \in R^D$

UNSUPERVISED

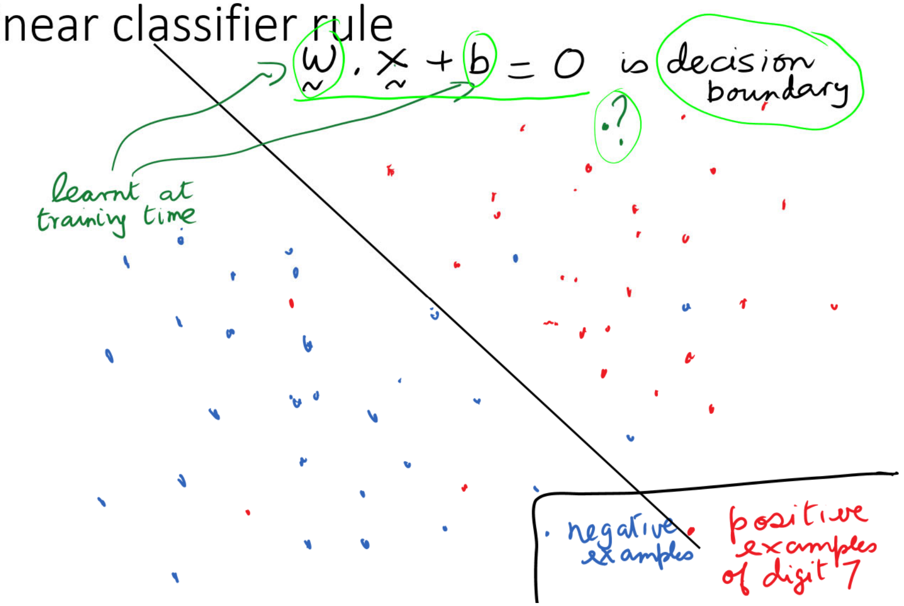
ML: main abstract ideas

- Training data set: $D = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x_i \in R^D$ $\quad y_i \in R$ or $y_i \in \{-1,1\}$

- Model class: $f(x|w, b) = w^T x + b$ **Linear Models**

aka hypothesis class

Linear classifier rule



ML: main abstract ideas

- Training data set: $D = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x_i \in R^D \quad y_i \in R$ or $y_i \in \{-1,1\}$

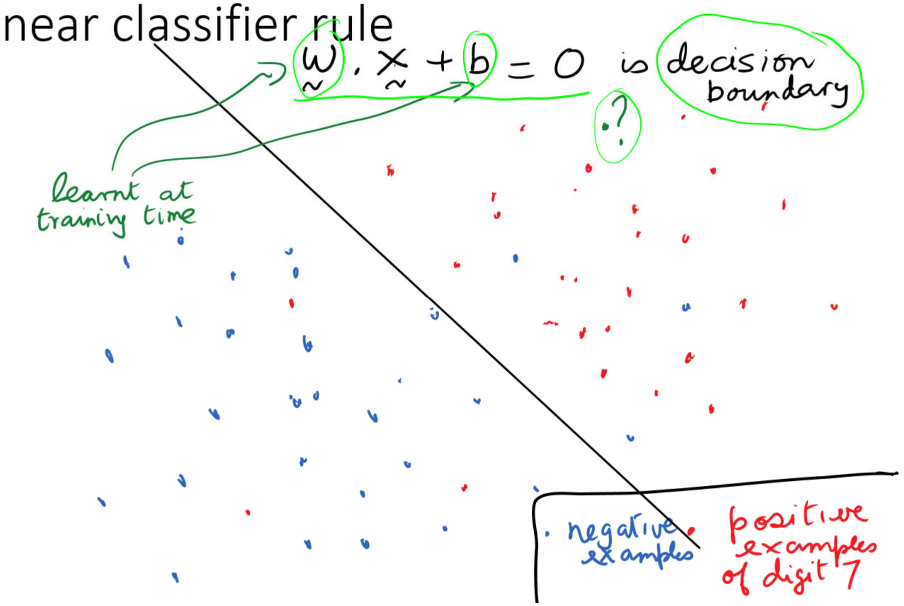
- Model class: $f(x|w, b) = w^T x + b$ **Linear Models**

aka hypothesis class

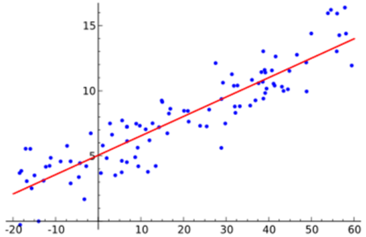
- **Optimization goal:** find "good" values of parameters (w, b).

  But was does "good" mean?

Linear classifier rule



ML: main abstract ideas



- Training data set: $D = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x_i \in R^D$ $\quad y_i \in R$ or $y_i \in \{-1,1\}$

- Model class: $f(x|w,b) = w^T x + b$ **Linear Models**

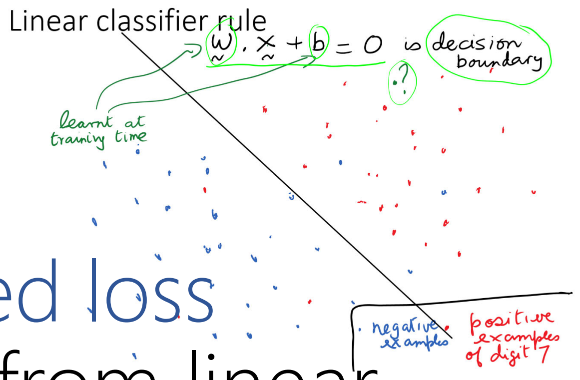
aka hypothesis class

- Loss Function: $L(a,b) = (a-b)^2$ **Squared Loss**

- Learning Objective: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$

Optimization Problem

Maximum Likelihood Estimation (MLE)



This principle gives a useful, principled and widely-used loss function to estimate parameters of statistical models (from linear regression, to neural networks, and beyond).

RVs!

- Training data set: $D = \{(x_i, y_i)\}_{i=1}^{N}$ $x_i \in R^D$, $y_i \in R$ or $y_i \in \{-1,1\}$
- Model class: $f(x|w,b) = w^T x + b$ **Linear Models**
  aka hypothesis class
- Loss Function: $L(a,b) = (a-b)^2$ **Squared Loss**
- Learning Objective: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, f(x_i | w, b))$

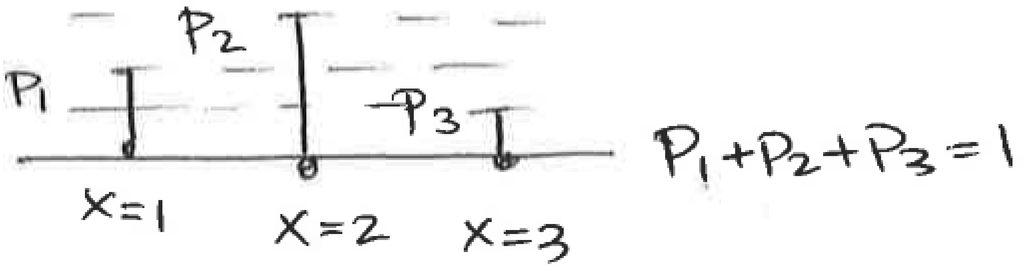
  Optimization Problem

Reminder: probability distributions

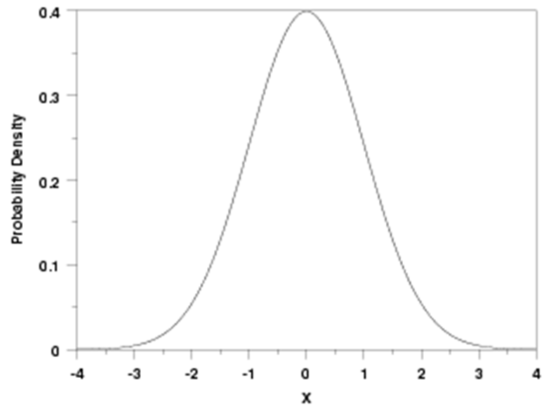
Random variable (RV) is a function: $\mathbf{x} \rightarrow \mathbb{R}$ e.g. $p(\text{T=heads}) = 0.5$

1. Discrete RV, e.g. coin toss heads/tails.
2. Continuous RV, e.g. height

Discrete RVs have a Probability Mass Function (PMF)



Continuous RVs have a Probability Density Function (PDF)



integrates to 1

e.g. distributions of discrete RVs

1. Bernoulli RV—model *one* toss of a coin that can be biased

   $P(heads) = p, \quad P(tails) = 1 - \boldsymbol{p}$, parameter is $\boldsymbol{p}$.

e.g. distributions of discrete RVs

1. <u>Bernoulli RV</u>—model *one* toss of a coin that can be biased
   $P(heads) = p, \quad P(tails) = 1 - p$, parameter is p.
2. <u>Binomial RV</u>—model *n coin tosses*, number of heads, $\mathbf{k}$

$$P(x=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

e.g. distributions of discrete RVs

1. Bernoulli RV—model *one* toss of a coin that can be biased
   $P(heads) = p, \quad P(tails) = 1 - p$, parameter is p.
2. Binomial RV—model *n coin tosses*, number of heads, $\mathbf{k}$

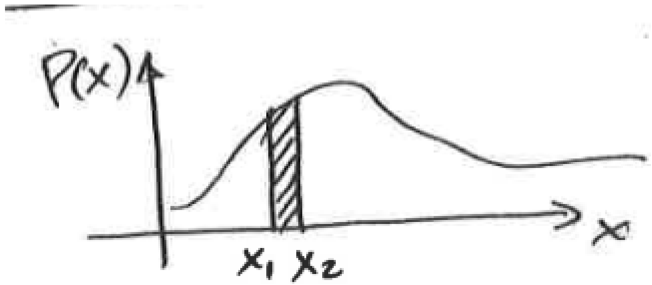
$$P(x=k) = \binom{n}{k} p^k (1-p)^{n-k}$$

3. Poisson RV– model number of mutations, k, occurring in a cell population with mean mutation rate, λ , over fixed time interval

$$P(x=k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

Distributions of continuous RVs

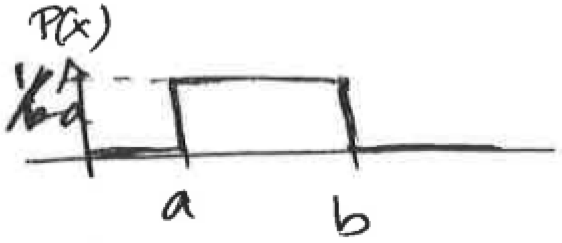
Continuous RVs have a Probability Density Function



$$\int_{x_1}^{x_2} P(x)\,dx = \text{Prob that the random variable } X \text{ has a value in the interval } [x_1 \ldots x_2]$$

Examples:

1. Uniform -



Parameters: a, b

2. Gaussian

$$P(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

Parameters: μ, σ

$$X \sim N(\mu, \sigma^2)$$

Multivariate distributions

Space of outcomes is a vector instead of a scalar:

<u>Multinomial</u> (generalization from binomial):

- urn with balls of different colors.
- Pick a ball at random.
- $\boldsymbol{p_1}$ it is green, $\boldsymbol{p_2}$ it is blue and $\boldsymbol{p_3}$ it is red

Multivariate distributions

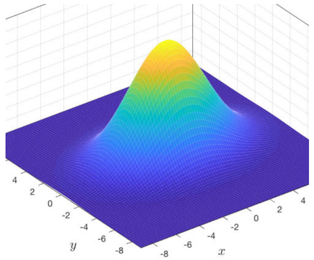
Space of outcomes is a vector instead of a scalar:

<u>Multinomial</u> (generalization from binomial):

- urn with balls of different colors.
- Pick a ball at random.
- $\boldsymbol{p_1}$ it is green, $\boldsymbol{p_2}$ it is blue and $\boldsymbol{p_3}$ it is red

<u>Multivariate Gaussian:</u>

- Mean is a vector, and variance becomes covariance.
- Will learn more about this next lecture.



The basic set-up of MLE

- Given data $D = \{x_i\}_{i=1}^{N}$ for $x_i \in R^d$
- Assume a set (family) of distributions on R^d, $\{p_\theta(x) | \theta \in \Theta\}$.

Same as $p(x|\theta)$

e.g. mean (μ) and variance (σ^2) for $x \in \mathbb{R}^1$

The basic set-up of MLE

Same as $p(x|\theta)$

- Given data $D = \{x_i\}_{i=1}^{N}$ for $x_i \in R^d$
- Assume a set (family) of distributions on R^d, $\{p_\theta(x)|\theta \in \Theta\}$.
- Assume D contains samples from one of these distributions:

$$x_i \sim p_{\hat{\theta}}(x)$$

- This assumes that each element of D is *identically and independently distributed* (iid).

The basic set-up of MLE

- Given data $D = \{x_i\}_{i=1}^{N}$ for $x_i \in R^d$
- Assume a set (family) of distributions on R^d, $\{p_\theta(x) | \theta \in \Theta\}$.
- Assume D contains samples from one of these distributions:

$$x_i \sim p_{\hat{\theta}}(x)$$

- This assumes that each element of D is *identically and independently distributed* (iid).

"likelihood function"

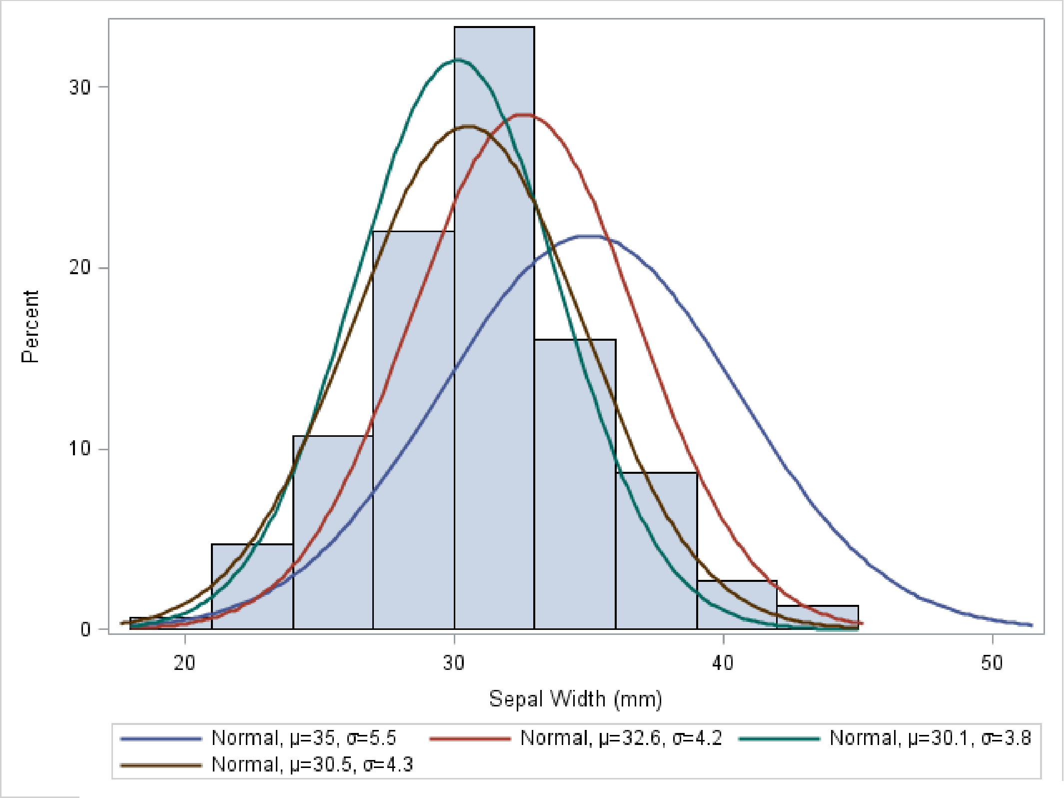
Goal of MLE: "learn"/estimate the value of θ that "pins down" the distribution from which the data came.

Definition: θ_{MLE} is a MLE for θ with respect to the data and family of distributions, if $\theta_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(D|\theta)$.

function of θ.

The basic set-up of MLE

$$\boxed{\theta_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(D|\theta)}$$



$$D = \{x_i\}_{i=1}^{N} = \{20.1, 33.8, 34.6, 36.2, \ldots\}$$

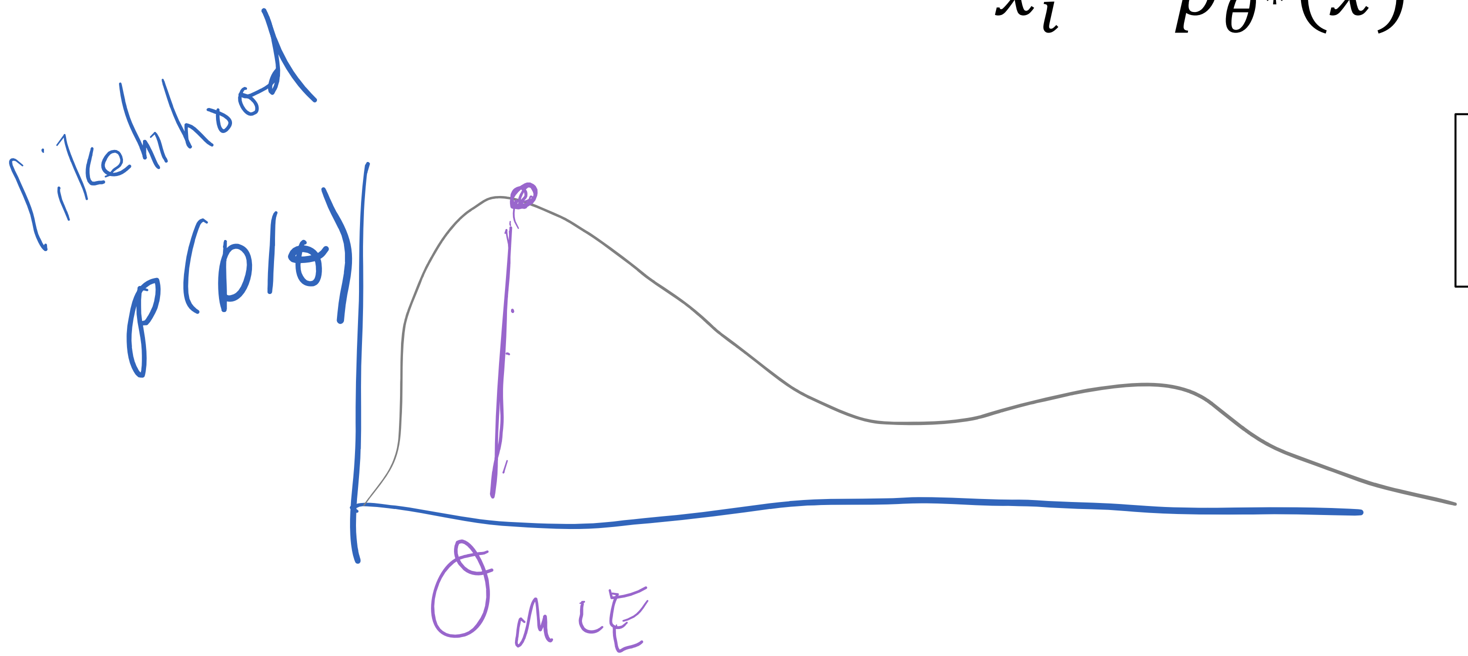
Note that $p(D|\theta) = p\left(\{x_i\}_{i=1}^{N} | \theta\right) = \prod_{i=1}^{N} p(x_i|\theta)$

because iid

The basic set-up of MLE

- Given data $D = \{x_i\}_{i=1}^{N}$ for $x_i \in R^d$
- Assume a set (family) of distributions on R^d, $\{p_\theta(x) | \theta \in \Theta\}$.
- Assume D contains samples from one of these distributions:

$$x_i \sim p_{\theta^*}(x)$$



$$\boxed{\theta_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(D|\theta)}$$

Is there always one unique MLE parameter value?

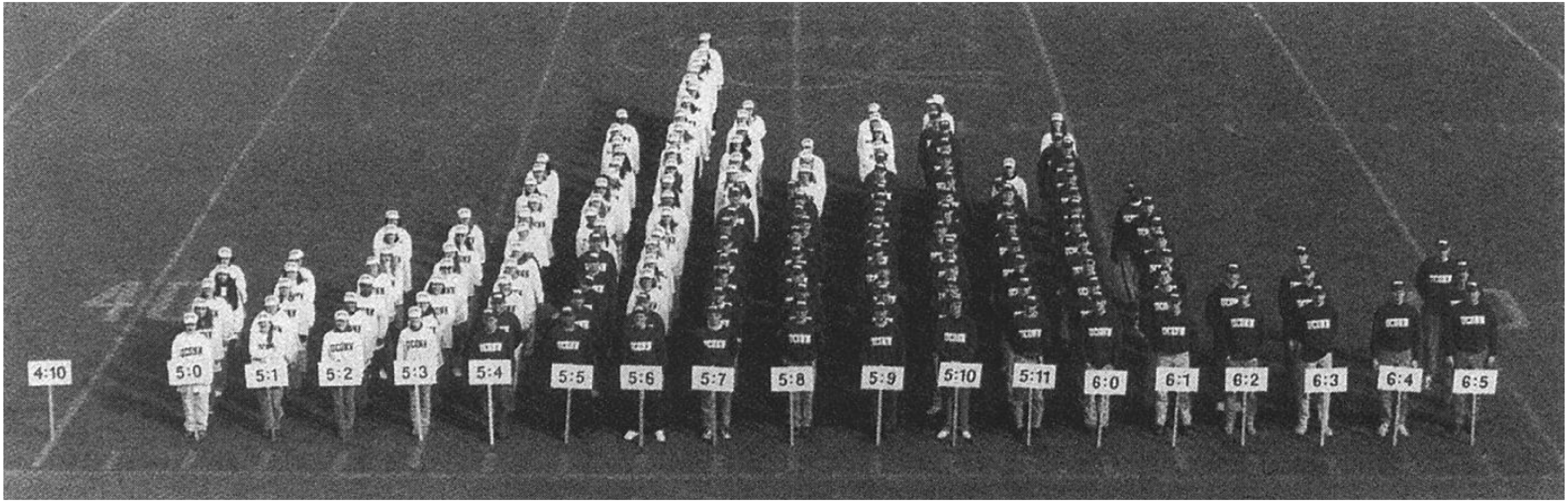
Some properties of MLE

eg $N(x \mid \mu, \sigma)$
vs
$N(x \mid 2+\mu, \sigma)$

- *Consistency*: as we get more and more data (drawn from one distribution in our family), then we converge to estimating the true value of $\boldsymbol{\theta}$ for $\boldsymbol{D}$.
- *Statistically efficient*: making good use of the data ("least variance" parameter estimates).
- The value of $p(D|\theta_{MLE})$ is invariant to re-parameterization.

- MLE can still yield a parameter estimate even when the data were not generated from that family (phew & caveat emptor).

e.g. MLE for univariate Gaussian



- Arguments can be made from the Central Limit Theorem that height is normally distributed.
- Suppose you were given a set if height measurements, $\{x_i\}$, how would you derive the estimate for the mean and variance, using MLE?

e.g. MLE for univariate Gaussian

Goal: $\theta_{MLE} = \underset{\theta \in \Theta}{\mathbf{argmax}}\, p(D|\theta)$ from set of data $D = \{x_i\}_{i=1}^{N}$

- Assume data are generated as $X \sim N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp - \frac{(x-\mu)^2}{2\sigma^2}$
- So assume MLE family of distributions, $p(X = x|\theta) = N(X|\mu, \sigma^2)$.
- Now our goal is to find $\theta_{MLE} = (\mu_{MLE}, \sigma^2_{MLE}) = \underset{\theta \in \Theta}{\mathbf{argmax}}\, p(D|\mu, \sigma^2)$.
- First step, write down the likelihood function:
  - $p(D|\theta) = p(x_1, x_2, \ldots x_N|\mu, \sigma^2) = \prod_{i=1}^{N} p(x_i|\mu, \sigma^2)$.
- The product of the terms is a little inconvenient to work with, so we will take the log to get a sum.

e.g. MLE for univariate Gaussian

- Likelihood: $p(x_1, x_2, \dots x_N | \mu, \sigma^2) = \prod_{i=1}^{N} p(x_i | \mu, \sigma^2)$.

- *Log likelihood ("LL")* is a monotonically increasing function of the likelihood.

$$\log p(D|\theta) = \sum_{i=1}^{N} \log p(x_i | \mu, \sigma^2)$$

- Therefore $\theta_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}}\, p(D|\theta) = \underset{\theta \in \Theta}{\operatorname{argmax}}\, \log p(D|\theta)$

e.g. MLE for univariate Gaussian

- Now we have a concrete optimization problem to work with:

$$\mu_{MLE}, \sigma^2_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}} \log p(D|\theta) = \underset{\mu, \sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log p\left(x_i \middle| \mu, \sigma^2\right)$$

- How will we solve this optimization problem?
- Find a setting of the parameters for which the partial derivatives are 0 (*i.e.*, a stationary point).
- Then check whether the setting is a maximum (negative second derivative), a minimum, etc. (first year calculus).
- (if #params>1, check if Hessian is negative definite; for 1D Gaussian Hessian is diagonal, so can check each separately).

e.g. MLE for univariate Gaussian

- Find the setting of the parameters that set the partial derivatives to zero:

$$\mu_{MLE}, \sigma^2_{MLE} = \underset{\theta\in\Theta}{\operatorname{argmax}} \log p(D|\theta) = \underset{\mu,\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log p\left(x_i \middle| \mu, \sigma^2\right)$$

- Lets expand out so we can take the derivative:

$$\frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log p(x_i | \mu, \sigma^2) = \sum_i \frac{\partial}{\partial \mu} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (x_i - \mu)^2 \right) \right]$$

e.g. MLE for univariate Gaussian

- Find the setting of the parameters that set the partial derivatives to zero:

$$\mu_{MLE}, \sigma^2_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}} \log p(D|\theta) = \underset{\mu,\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(x_i|\mu, \sigma^2)$$

- Lets expand out so we can take the derivative:

$$\frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log p(x_i|\mu, \sigma^2) = \sum_i \frac{\partial}{\partial \mu} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)\right]$$

$$= \sum_i \frac{\partial}{\partial \mu}\left[\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2}(x_i-\mu)^2\right]$$

$$= \sum_i \left[0 + \frac{1}{\sigma^2}(x_i-\mu)\right] \xRightarrow{\text{set to zero}} \sum_i x_i = \sum_i \mu$$

$$\sum_i x_i = N\mu$$

$$\Rightarrow \mu = \frac{\sum_i x_i}{N}$$

e.g. MLE for univariate Gaussian

$\frac{\partial^2 (LL)}{\partial \mu^2} =$

- Find the setting of the parameters that set the partial derivatives to zero:

$$\mu_{MLE}, \sigma^2_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}} \log p(D|\theta) = \underset{\mu,\sigma^2}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(x_i|\mu, \sigma^2)$$

- Lets expand out so we can take the derivative:

$$\frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log p(x_i | \mu, \sigma^2) = \sum_i \frac{\partial}{\partial \mu} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (x_i - \mu)^2\right) \right]$$

$$= \sum_i \frac{\partial}{\partial \mu} \left[-\log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} (x_i - \mu)^2 \right]$$

$$= \sum_i \left[0 + \frac{1}{\sigma^2} (x_i - \mu) \right] \Rightarrow$$

set to zero

$$\sum_i x_i = \sum_i \mu \Rightarrow \sum_i x_i = N\mu$$

$$\Rightarrow \mu = \frac{\sum_i x_i}{N}$$

e.g. MLE for univariate Gaussian

$$\frac{\partial^2 (LL)}{\partial \mu^2} = \sum_i \frac{1}{\sigma^2} \cdot (-1) = -\frac{N}{\sigma^2} < 0$$

$\Rightarrow$ max

MLE

$$\frac{\partial}{\partial \mu} \sum_{i=1}^{N} \log p(x_i \mid \mu, \sigma^2) = \sum_i \frac{\partial}{\partial \mu} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right)\right]$$

$$= \sum_i \frac{\partial}{\partial \mu}\left[\log\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2}(x_i-\mu)^2\right]$$

$$= \sum_i \left[0 + \frac{1}{\sigma^2}(x_i-\mu)\right]$$

set to zero

$$\Rightarrow \sum_i x_i = \sum_i \mu \Rightarrow \sum_i x_i = N\mu \Rightarrow \mu = \frac{\sum_i x_i}{N}$$

e.g. MLE for univariate Gaussian

$$N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp - \frac{(x-\mu)^2}{2\sigma^2}$$

$$\mu_{MLE}, \sigma^2_{MLE} = argmax \sum_{i=1}^{N} \log N(x_i|\mu,\sigma^2)$$

- Again, but this time for $\boldsymbol{\sigma^2}$:

$$\frac{\partial \mathcal{LL}}{\partial \sigma^2} = \frac{\partial}{\partial \sigma^2}\left(-\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right)$$

$$= -\frac{N}{2}\cdot\frac{\partial}{\partial\sigma^2}\left(\log(2\pi\sigma^2)\right) + \frac{\partial}{\partial\sigma^2}\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right)$$

$$= -\frac{N}{2}\cdot\frac{1}{2\pi\sigma^2}\cdot 2\pi + \frac{\partial}{\partial\sigma^2}\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right)$$

$$= -\frac{N}{2\sigma^2} + \sum_{n=1}^{N}\left(-\frac{1}{2}\cdot -1\cdot(\sigma^2)^{-2}\cdot 1\cdot(x_n-\mu)^2\right)$$

$$= -\frac{N}{2\sigma^2} + \sum_{n=1}^{N}\left(\frac{1}{2\sigma^4}\cdot(x_n-\mu)^2\right)$$

$$0 = \frac{1}{2\sigma^2}\left(-N + \frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right)$$

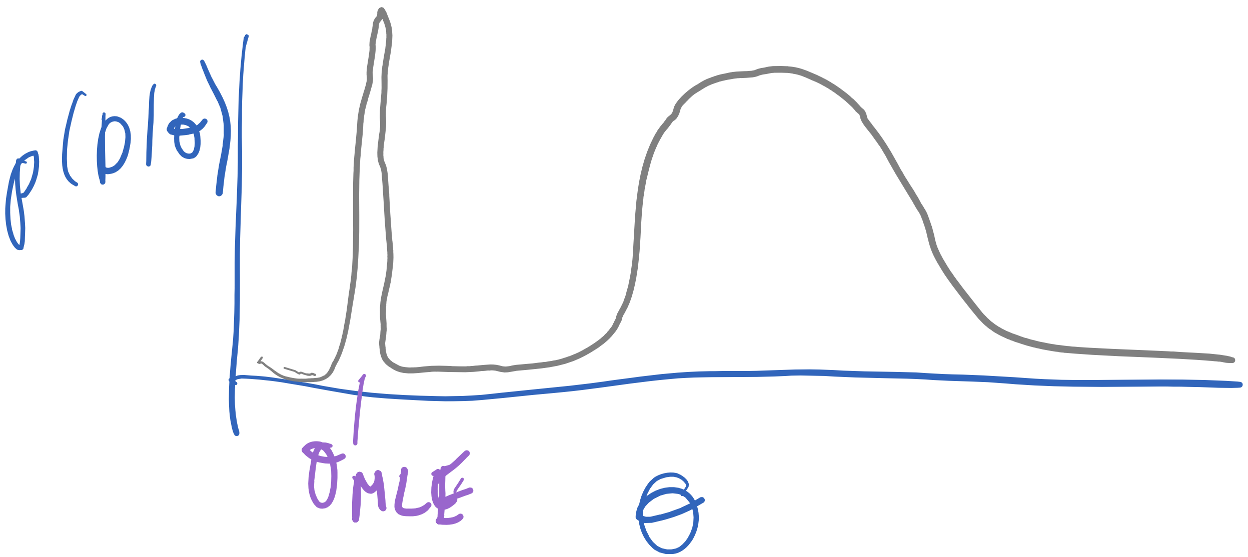
$$0 = -N + \frac{1}{\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2$$

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)^2$$

σ^2_{MLE}

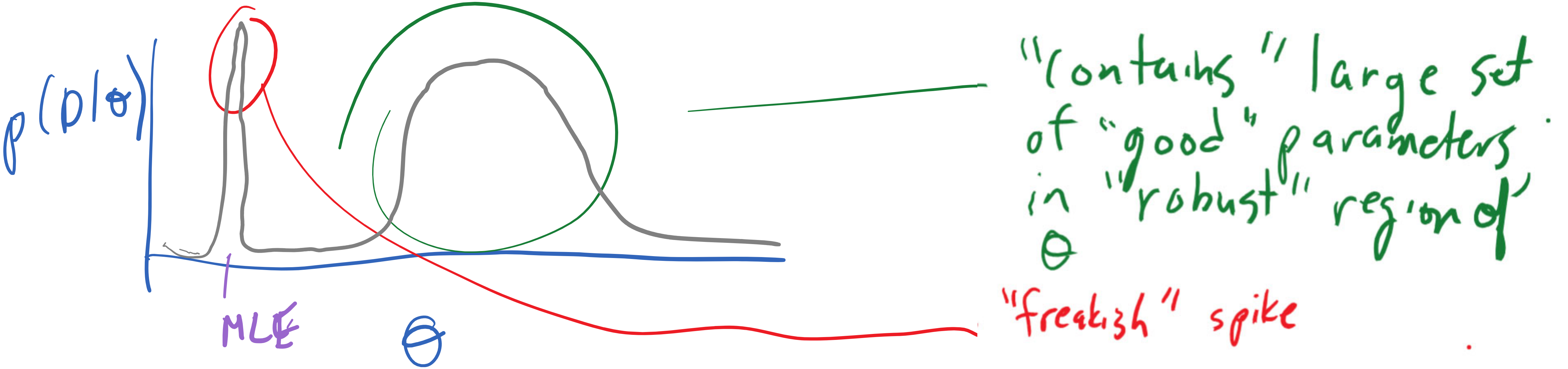
MLE yields a "point estimate" of our parameter

- When we perform MLE, we get just one single estimate of the parameter, $\boldsymbol{\theta}$, rather than a distribution over it (which captures uncertainty).
- In Bayesian statistics, we obtain a (posterior) distribution over $\boldsymbol{\theta}$. We will touch more on this in a few lectures.



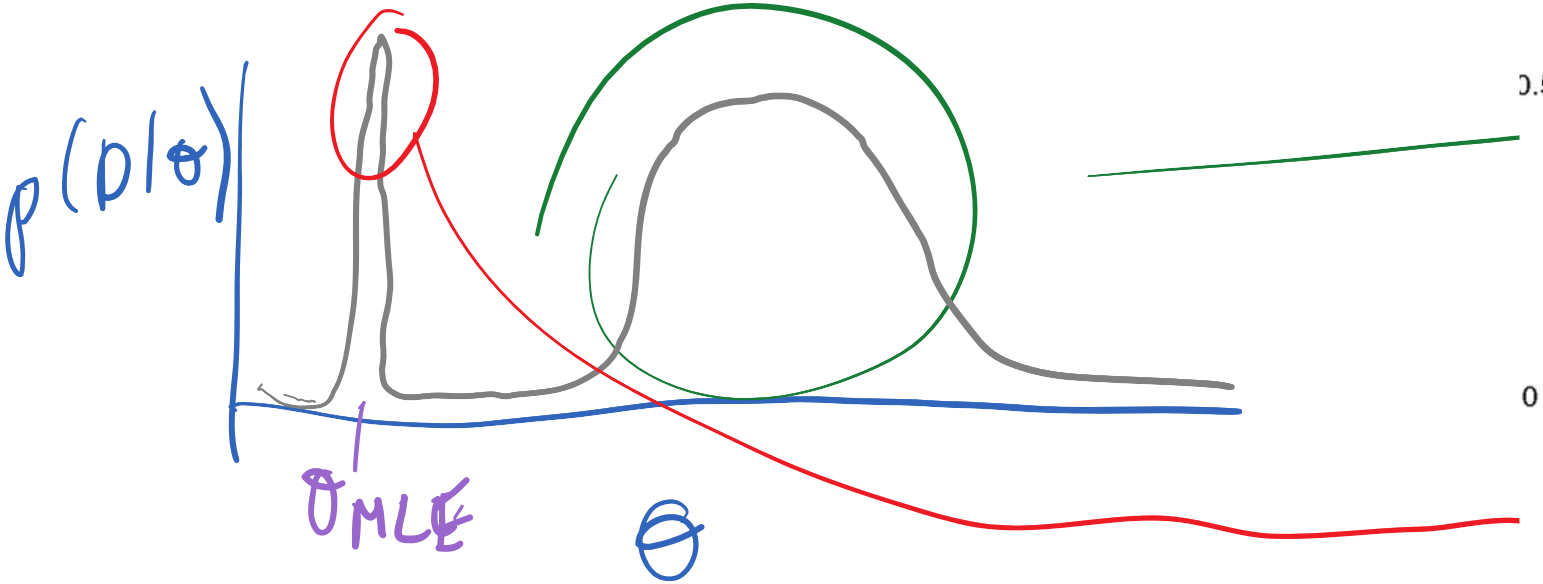
MLE yields a “point estimate” of our parameter

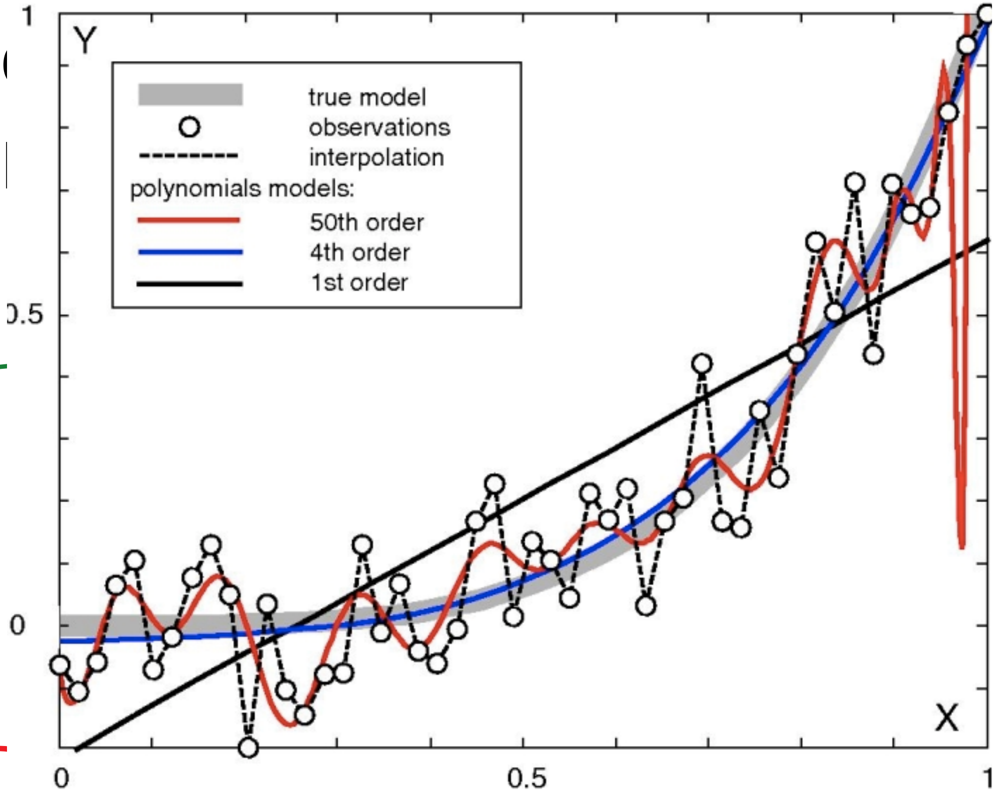
- When we perform MLE, we get just <u>one single estimate of the parameter, $\boldsymbol{\theta}$,</u> rather than a distribution over it which captures uncertainty.
- In Bayesian statistics, we obtain a (posterior) distribution over $\boldsymbol{\theta}$. We will touch more on this in a few lectures.



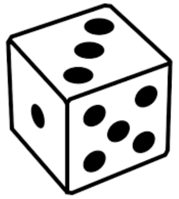
MLE yields a “point estimate” of our parameter

- When we perform MLE, we get just one single estimate of the parameter, $\boldsymbol{\theta}$, rather than a distribution over it which captures uncertainty.
- In Bayesian statistics, we obtain a (p
over $\boldsymbol{\theta}$. We will touch more on this i





e.g. MLE for the multinomial distribution



- Consider a six-sided die that we will roll: we want to know the probability of each side of the die turning up ($\theta = \theta_1 \ldots \theta_6$).
- Assume we have observed N rolls, with RV, $X \sim p_\theta(X)$.
- We write that $P(X = k|\theta) = \theta_k$ (when k^{th} side faced up).
- Lets use MLE to estimate these parameters.
- First, since one side must always face up, we know that $1 = \sum_k \theta_k$.
- Second, let us denote $P(X = x|\theta) \equiv \theta_x$ (pick off the right parameter).
- Now we write the likelihood:

$n_k \equiv |\{i | x_i = k\}|$

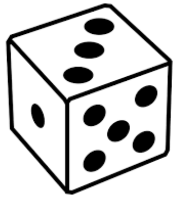
$$P(D|\theta) = p(x_1, \ldots x_N|\theta) = \prod_{i=1}^{N} p(x_i|\theta) = \prod_{i=1}^{N}\prod_{k=1}^{6} \theta_k^{I[x_i=k]} = \prod_{k=1}^{6} \theta_k^{\sum_i^N I[x_i=k]} = \prod_{k=1}^{6} \theta_k^{n_k}$$

Now our MLE problem becomes:

constrained optimization

$$\theta_{MLE} = \underset{\theta\in\Theta}{\operatorname{argmax}} \log p(D|\theta) = \underset{\theta\in\{\Theta|1=\sum_k \theta_k\}}{\operatorname{argmax}} \sum_{k=1}^{6} \log \theta_k^{n_k}$$

e.g. MLE for the multinomial distribution



Have a constrained optimization problem:

constrained optimization

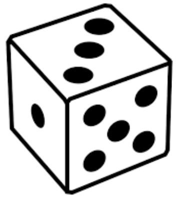
$$\theta_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}} \log p(D|\theta) = \underset{\theta \in \{\Theta | 1 = \sum_k \theta_k\}}{\operatorname{argmax}} \sum_{k=1}^{6} \log \theta_k^{n_k}$$

What is one technique you should have learned in first year calculus to solve this?

The technique of Lagrange multipliers (Appendix C of textbook):

$J(\theta, \lambda) = \log p(D|\theta) + \lambda(1 - \sum_k \theta_k)$ (look for stationary points wrt θ, λ)

e.g. MLE for the multinomial distribution



$$J(\theta,\lambda) = log\, p(D|\theta) + \lambda(1 - \sum_k \theta_k) = \sum_{k=1}^{6} \log \theta_k^{n_k} + \lambda(1 - \sum_k \theta_k)$$

1. $\frac{\partial J}{\partial \lambda} = 0 \Rightarrow 1 = \sum_k \theta_k$ (we just get the constraint back)
2. $\frac{\partial J}{\partial \theta_k} = \frac{\partial}{\partial \theta_k} \sum_{k=1}^{6} \log \theta_k^{n_k} - \frac{\partial}{\partial \theta_k} \lambda\theta_k = \frac{n_k}{\theta_k} - \lambda = 0 \Rightarrow \theta_k = \frac{n_k}{\lambda}$.
3. Lets plug this into 1), $1 = \sum_k \theta_k = \sum_k \frac{n_k}{\lambda} \Rightarrow \lambda = \sum_k n_k = N$.
4. All together then, $\theta_k = \frac{n_k}{N}$.

Doing MLE requires optimization

$$\theta_{MLE} = \underset{\theta \in \Theta}{\operatorname{argmax}} \log p(D|\theta)$$

- For Gaussian, multinomial, and more, the MLE can be obtained in closed form by setting the derivative to zero.
- What if we had a neural network model such as mentioned in the first lecture?

- Here, we need *iterative optimization* (can take entire classes on special cases of this (e.g. Convex Optimization). More later.

$$\boxed{V_j = g\left(\sum_K w_{jK} x_K\right) ; \quad O_i = g\left(\sum w_{ij} V_j\right)}$$