CONVENTIONAL (ALGORITHMIC) VIEW OF ML

features $\rightarrow \theta \rightarrow$ 5 predicted label $\hat{y}$

PROBABILITY MODELLER’S VIEW

random predicted label $Y$

5 with prob 45%
3 with prob 41%
6 with prob 5%

ground truth:
Let $y$ be the actual observed label in the dataset

evaluation metric:
loss function e.g. $L(y, \hat{y}) = 1_{\hat{y} \neq y}$

evaluation metric:
log likelihood i.e. $\log \Pr_Y(y)$
CONVENTIONAL (ALGORITHMIC) VIEW OF MODELLING

- timepoint $t$ → model parameters $\theta$ → predicted temperature $\hat{y}$ at timepoint $t$

Evaluation metric:
- loss function $e.g., L(y, \hat{y}) = (y - \hat{y})^2$

PROBABILITY MODELLER’S VIEW

- timepoint $t$ → model parameters $\theta$ → random variable for temperature $Y$ at timepoint $t$

Evaluation metric:
- log likelihood $i.e., \log \Pr_Y(y; t)$

Ground truth:
- Let $y$ be the actual observed temperature at time $t$
Our job is to invent a probability model, specifying the distribution of temperature at a given timepoint.
Example (regression)

Given a labelled dataset consisting of pairs \((x_i, y_i)\) of real numbers, fit the model \(Y_i \sim \alpha + \beta x_i + \gamma x_i^2 + N(0, \sigma^2)\)

Model for a single observation:
\[
Y \sim \alpha + \beta x + \gamma x^2 + N(0, \sigma^2)
\]

\[
\sim N(\alpha + \beta x + \gamma x^2, \sigma^2)
\]

Likelihood of a single observation:
\[
Pr_Y(y; x, \alpha, \beta, \gamma, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-(\alpha+\beta x+\gamma x^2))/2\sigma^2}
\]

Log likelihood of the dataset:
\[
\log Pr(y_1, \ldots, y_n; x_1, \ldots, x_n, \alpha, \beta, \gamma, \sigma) = -\frac{n}{2} \log(2\pi \sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2
\]

where \(\hat{y}_i = \alpha + \beta x_i + \gamma x_i^2\)

Optimize over the unknown parameters:
```python
def logpr(y, x, alpha, beta, gamma, sigma):
    pred = alpha + beta * x + gamma * (x**2)
    return -0.5 * np.log(2 * np.pi * sigma**2) - (y - pred)**2 / (2 * sigma**2)

def f(theta):
    return -np.sum(logpr(y, x, theta[0], theta[1], theta[2], theta[3]))

scipy.optimize.fmin(f, [3, 1, 0.1, 3])
```
Example (regression)
Given a labelled dataset consisting of pairs \((x_i, y_i)\) of real numbers, fit the model \(Y_i \sim \mu_\theta(x_i) + N(0, \sigma^2)\)

Log likelihood of the dataset:

\[
\log \Pr(y_1, \ldots, y_n; x_1, \ldots, x_n, \theta, \sigma) = -\frac{n}{2} \log(2\pi \sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu_\theta(x_i))^2
\]

Optimize over the unknown parameters \(\theta\) and \(\sigma\):
\[
\begin{align*}
\text{edge weights } \theta \\
\text{self.} \mu &= \text{nn.Sequential}(
\text{nn.Linear}(1,4), \text{nn.LeakyReLU}(), \\
\text{nn.Linear}(4,20), \text{nn.LeakyReLU}(), \\
\text{nn.Linear}(20,20), \text{nn.LeakyReLU}(), \\
\text{nn.Linear}(20,1) ) \\
\end{align*}
\]
This is your machine learning system?

Yup! You pour the data into this big pile of linear algebra, then collect the answers on the other side.

What if the answers are wrong?

Just stir the pile until they start looking right.
Example (regression)

Given a labelled dataset consisting of pairs \((x_i, y_i)\) of real numbers, fit the model \(Y_i \sim \mu_\theta(x_i) + N(0, \sigma^2)\).

The question implies that \(\mu_\theta(\cdot)\) is some given function with unknown parameter \(\theta\).

Log likelihood of the dataset:

\[
\log \Pr(y_1, \ldots, y_n; x_1, \ldots, x_n, \theta, \sigma) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu_\theta(x_i))^2
\]

Optimize over the unknown parameters \(\theta\) and \(\sigma\):

\[
\max_{\theta, \sigma} \left\{ -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu_\theta(x_i))^2 \right\}
\]

\[
= \max_{\sigma} \left\{ \max_{\theta} \left[ -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu_\theta(x_i))^2 \right] \right\}
\]

\[
= \max_{\sigma} \left\{ -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \min_{\theta} \left[ \sum_{i=1}^{n} (y_i - \mu_\theta(x_i))^2 \right] \right\}
\]

"First, find \(\theta\) to minimize prediction loss. Next, pick \(\sigma\)."
Any useful “minimize prediction loss” problem can be restated as a maximum likelihood problem.

- Supervised ML is fitting a probability model.
- Probability modelling is a more powerful way to think about ML.
Example (classification)
The MNIST dataset consists of pairs \((x_i, y_i)\), where each record consists of \(x_i \in \mathbb{R}^{28 \times 28}\) an image of a handwritten digit and \(y_i \in \{0, 1, \ldots, 9\}\) is its label.

Devise a probabilistic model to predict the label of a given input image, and fit it.

What sort of probability model might we use for the response \(Y\)?
Example (classification)
The MNIST dataset consists of pairs \((x_i, y_i)\), where each record consists of
\(x_i \in \mathbb{R}^{28 \times 28}\) an image of a handwritten digit and \(y_i \in \{0,1,...,9\}\) is its label.

Devise a probabilistic model to predict the label of a given input image, and fit it.

\[
\begin{align*}
\text{input } x & \quad \rightarrow \quad \text{some arbitrary function, parameters } \theta \\
\hat{p}(x) & = [p_0, \ldots, p_9] \\
Y & \sim \text{Cat}(\hat{p}(x))
\end{align*}
\]

How can we make sure that \(\hat{p}\) is a valid probability vector?

(We need \(p_i \in [0,1]\) for each \(i\), and \(\sum_i p_i = 1\).)
Example (classification)
The MNIST dataset consists of pairs $(x_i, y_i)$, where each record consists of $x_i \in \mathbb{R}^{28 \times 28}$ an image of a handwritten digit and $y_i \in \{0,1,\ldots,9\}$ is its label.

Devise a probabilistic model to predict the label of a given input image, and fit it.

How should we fit the function parameters $\theta$?

Softmax function:
$$p_k = \frac{e^{s_k}}{\sum_{\ell=0}^{9} e^{s_\ell}}$$
The MNIST dataset consists of pairs $(x_i, y_i)$, where each record consists of $x_i \in \mathbb{R}^{28 \times 28}$ an image of a handwritten digit and $y_i \in \{0,1,\ldots,9\}$ is its label.

Deviwe a probabilistic model to predict the label of a given input image, and fit it.

This is called softmax cross-entropy, and it’s the standard loss function for classification.
some arbitrary function, parameters $\theta$:

$$\begin{align*}
\text{input } x & \rightarrow \tilde{s} \in \mathbb{R}^{10} & \tilde{p} = \text{softmax}(\tilde{s}) & \sim \text{Cat}(\tilde{p})
\end{align*}$$

HOMEWORK

- Make sure you can run this code
- For each digit $\{0,1,\ldots,9\}$, select some images where the label is very likely, and some where it is very unlikely
This is machine learning, too!
CONVENTIONAL VIEW OF MACHINE LEARNING

**Supervised Learning**

- **Data:** \( \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \)
- **Labels:** \(y_1, y_2, \ldots, y_n\)
- **Task:** Predict the label \(y_i \approx f_\theta(x_i)\)
- **Training goal:** Invent a loss function and learn \(\theta\) to minimize the prediction loss \(\sum_i L(y_i, f_\theta(x_i))\)
- **Evaluation:** Prediction loss on holdout data

**Generative Modelling**

- **Data:** \(\{x_1, x_2, \ldots, x_n\}\)
- **Labels:** n/a
- **Task:** Learn to synthesize new values similar (but not identical) to those in the dataset, ...
- **Training goal:** ???
- **Evaluation:** ???
Section 3.4. Latent-variable generative models

random noise $Z$ \[ \longrightarrow \] edge weights $\theta$ \[ \longrightarrow \] $X = f_{\theta}(Z)$

The output $X$ is a random variable. It therefore has a likelihood function $\Pr_X(x)$.

QUESTION. How could we even use neural networks to generate novel images? What should the input be?
Supervised Learning

Data: \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)

Labels: \(y_1, y_2, \ldots, y_n\)

Task: Fit a probability model
\[ \Pr_Y(y_i; f_\theta(x_i)) \]

Training goal: Learn \(\theta\) to maximize the log likelihood of the dataset
\[ \sum_i \log \Pr_Y(y_i; f_\theta(x_i)) \]

Evaluation: log likelihood of holdout data

Generative Modelling

Data: \(\{x_1, x_2, \ldots, x_n\}\)

Labels: n/a

Task: Fit the probability model
\[ \Pr_X(x; \theta) \]

Training goal: Learn \(\theta\) to maximize the log likelihood of the dataset
\[ \sum_i \log \Pr_X(x_i; \theta) \]

Evaluation: log likelihood of holdout dataset
Exercise (generative modelling).
Train a generative model for a collection of points \( x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^2 \). The model should have the form

\[
\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \sim f(Z) + \begin{bmatrix} N(0, \sigma^2) \\ N(0, \sigma^2) \end{bmatrix}
\]

where \( Z \sim U[0,1] \) and \( f: [0,1] \to \mathbb{R}^2 \) is a neural network to be trained.

Model for a single observation

\[
Z \sim U[0,1] \\
X_1 \sim f_1(Z) + N(0, \sigma^2) \\
X_2 \sim f_2(Z) + N(0, \sigma^2)
\]

Likelihood for a single observation

\[
\Pr(x_1, x_2) = \int_{z=0}^{1} \Pr(x_1, x_2 | Z = z) \Pr_Z(z) \, dz
\]

\[
\Pr(x_1 | Z = z) \Pr(x_2 | Z = z)
\]

\[
\Pr(x_i | Z = z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(x_i - f_i(z))^2 / 2\sigma^2}
\]

Log likelihood of the dataset

\[
\sum_{i=1}^{n} \log \Pr(x^{(i)}_1, x^{(i)}_2)
\]

Maximize over unknown parameters

(We’ll approximate the integral over \( z \) by a sum.)
Our job is to invent a probability model, specifying the **distribution** of the response at a given input.
What are we really after, when we fit a probability model?

What’s a good model?
How can we compare models?
dataset of $(x_i, y_i)$ pairs
\[ Y_i \sim 1.62 + 0.49 x_i + \text{Normal}(0, 2.39^2) \]

\[ Y_i \sim -38.5 + 95.7 x_i - 84.8 x_i^2 + 38.3 x_i^3 -9.5 x_i^4 + 1.3 x_i^5 - 0.09 x_i^6 + 0.003 x_i^7 + \text{Normal}(0, 0.31^2) \]

**Question**

Which of these two models fits the dataset better?
dataset $\{x_1, \ldots, x_n\}$
Question
Which of these two models fits the dataset better?

Model A:
IID sample from
\( X \sim N(0.9, 0.03^2) \)
This model is extraordinarily unlikely to generate the dataset, so it's a bad model.
\[ \log \text{lik} \text{ (dataset)} = -570.5 \]

Model B:
IID sample from
\( X \sim N(0.1, 0.6^2) \)
This model might possibly have generated the data (but it's still not great).
\[ \log \text{lik} \text{ (dataset)} = -28.0 \]
(dataset of \((x_i, y_i)\) pairs)

\[
Y_i \sim 1.62 + 0.49 \, x_i \\
+ \text{Normal}(0, 2.39^2)
\]

\[
Y_i \sim -38.5 + 95.7 \, x_i - 84.8 \, x_i^2 + 38.3 \, x_i^3 \\
-9.5 \, x_i^4 + 1.3 \, x_i^5 - 0.09 \, x_i^6 + 0.003 \, x_i^7 \\
+ \text{Normal}(0, 0.31^2)
\]

**Question**
Which of these two models fits the dataset better?
\begin{align*}
Y_i & \sim 1.62 + 0.49 \, x_i \\
& \quad + \text{Normal}(0, 2.39^2) \\
\log \text{lik} (\text{dataset}) & = -64.6 \\
\text{This is the better model.}
\end{align*}

\begin{align*}
Y_i & \sim -38.5 + 95.7 \, x_i - 84.8 \, x_i^2 + 38.3 \, x_i^3 \\
& \quad - 9.5 \, x_i^4 + 1.3 \, x_i^5 - 0.09 \, x_i^6 + 0.003 \, x_i^7 \\
& \quad + \text{Normal}(0, 0.31^2) \\
\log \text{lik} (\text{dataset}) & = -379.3
\end{align*}
The goal of modelling is to find models that fit the dataset well.

A good metric for model fit is: likelihood of the dataset, according to the model.

This applies equally to both supervised and generative modelling.

In NLP, log likelihood is called “perplexity.”

In sports betting, log likelihood is called “ignorance score.”
Do You Understand the Variance In Your Data?

by Thomas C. Redman

August 16, 2019
Number of defective pairs of shoes each day
Monthly average temperatures in Cambridge, UK

What's a good model for this dataset?

Climate is stable?

\[ \text{Temp}(t) \sim a + b \sin(2\pi(t + \phi)) + N(0, \sigma^2) \]

Temperatures are increasing?

Temperatures are increasing, and the increase is accelerating?

The extremes are getting worse?
You’ve got to have models in your head. And you’ve got to array your experience – both vicarious and direct – on this latticework of models.

You may have noticed students who just try to remember and pound back what is remembered. Well, they fail in school and in life. You’ve got to hang experience on a latticework of models in your head.

Charlie Munger, *A lesson on elementary, worldly wisdom as it relates to investment management & business.*
Example 2.1.1

The Iris dataset has 50 records of iris measurements, from three species.

<table>
<thead>
<tr>
<th>Petal. Length</th>
<th>Petal. Width</th>
<th>Sepal. Length</th>
<th>Sepal. Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.2</td>
<td>4.6</td>
<td>3.6</td>
<td>setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>1.9</td>
<td>6.3</td>
<td>2.5</td>
<td>virginica</td>
</tr>
<tr>
<td>5.8</td>
<td>1.6</td>
<td>7.2</td>
<td>3.0</td>
<td>virginica</td>
</tr>
<tr>
<td>4.2</td>
<td>1.2</td>
<td>5.7</td>
<td>3.0</td>
<td>versicolor</td>
</tr>
</tbody>
</table>

... 

How does Petal.Length depend on Sepal.Length?

Let’s guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),

$$\text{Petal.Length} \sim \alpha + \beta \text{Sepal.Length} + \gamma (\text{Sepal.Length})^2 + N(0, \sigma^2)$$

Dataset collected by Edgar Anderson and popularized by Ronald Fisher in 1936
Example 2.1.1
The Iris dataset has 50 records of iris measurements, from three species.

<table>
<thead>
<tr>
<th>Petal. Length</th>
<th>Petal. Width</th>
<th>Sepal. Length</th>
<th>Sepal. Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.2</td>
<td>4.6</td>
<td>3.6</td>
<td>setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>1.9</td>
<td>6.3</td>
<td>2.5</td>
<td>virginica</td>
</tr>
<tr>
<td>5.8</td>
<td>1.6</td>
<td>7.2</td>
<td>3.0</td>
<td>virginica</td>
</tr>
<tr>
<td>4.2</td>
<td>1.2</td>
<td>5.7</td>
<td>3.0</td>
<td>versicolor</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How does Petal. Length depend on Sepal. Length?

Let’s guess that for parameters $\alpha, \beta, \gamma, \sigma$ (to be estimated),

$$
Petal.\text{Length} \sim \alpha + \beta \text{Sepal. Length} + \gamma \text{(Sepal. Length)}^2 + N(0, \sigma^2)$$
2.1. LINEAR MODELS

Models of this form are called linear models (because they’re based on linear algebra).

They are flexible, and very fast to optimize.

We’ll assume Gaussian errors. Thus, maximum likelihood estimation is the same as minimizing squared prediction loss. Linear modelling is also called “least squares model-fitting”.
iris = pandas.read_csv(...)  

\[ \begin{bmatrix} \text{PL}_1 \\ \text{PL}_2 \\ \vdots \\ \text{PL}_n \end{bmatrix} \approx \alpha \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \beta \begin{bmatrix} \text{SL}_1 \\ \text{SL}_2 \\ \vdots \\ \text{SL}_n \end{bmatrix} + \gamma \begin{bmatrix} (\text{SL}_1)^2 \\ (\text{SL}_2)^2 \\ \vdots \\ (\text{SL}_n)^2 \end{bmatrix}. \]

Fitting the model

```python
one, SL, PL = np.ones(len(iris)), iris['Sepal.Length'], iris['Petal.Length']
model = sklearn.linear_model.LinearRegression(fit_intercept=False)
model.fit(np.column_stack([one, SL, SL**2]), PL)
(α, β, γ) = model.coef_
```

Making predictions / getting fitted values from the model

```python
newSL = np.linspace(4.2, 8.2, 20)
predPL = model.predict(np.column_stack([one, newSL, newSL**2]))
```
Feature design

How do we design features, so that linear models answer the questions we want answered?
ONE-HOT CODING

Want to fit three straight lines

$PL \approx \alpha_{\text{species}} + \beta_{\text{species}} \cdot SL$

This has six parameters to fit:

$\alpha_{\text{seto}}, \alpha_{\text{virg}}, \alpha_{\text{vers}}$

$\beta_{\text{seto}}, \beta_{\text{vers}}, \beta_{\text{virg}}$

e.g. first row is setosa, and linear model says

$PL_1 \approx \alpha_{\text{seto}} + \beta_{\text{seto}} \cdot SL_1$

Linear model form

(linear combination of features, weighted by parameters)

$\begin{bmatrix}
PL_1 \\
PL_2 \\
PL_3 \\
PL_4 \\
\vdots
\end{bmatrix}
\approx
\begin{bmatrix}
\alpha_{\text{seto}} \\
\alpha_{\text{virg}} \\
\alpha_{\text{vers}} \\
\beta_{\text{seto}} \\
\beta_{\text{vers}} \\
\beta_{\text{virg}} \\
\end{bmatrix}
\cdot
\begin{bmatrix}
SL_1 \\
SL_2 \\
SL_3 \\
SL_4 \\
\vdots
\end{bmatrix}$

1 $\text{species, SL} = \text{iris}['\text{Species}'], \text{iris}['\text{Sepal.Length}']$
2 $PL = \text{iris}['\text{Petal.Length}']$
3 $\text{species} \_\text{levels} = [\text{setosa}, \text{versicolor}, \text{virginica}]$
4 $i1, i2, i3 = [\text{np.where}('\text{species}==k, 1, 0) \text{ for k in species_levels}]$
5 $X = \text{np.column_stack}([i1, i2, i3, i1*SL, i2*SL, i3*SL])$
6 $\text{model} = \text{sklearn.linear_model.LinearRegression}(\text{fit_intercept}=\text{False})$
7 $\text{model.fit}(X, PL)$
NON-LINEAR RESPONSE

\[ \text{Petal.Length} \approx \alpha + \beta \text{Sepal.Length} + \gamma (\text{Sepal.Length})^2 \]

\[ \text{Petal.Length} \approx \beta_0 + \sum_{k=1}^{K} \beta_k (\text{Sepal.Length})^k \]
NON-LINEAR RESPONSE via one-hot coding

\[ PL \approx \alpha_4 1_{SL<4} + \alpha_5 1_{|SL|=5} + \alpha_6 1_{|SL|=6} + \alpha_7 1_{SL\geq7} \]

e.g. for an observation with SL=5.3, we predict \( PL \approx \alpha_5 \)
e.g. for an observation with SL=3.1, we predict \( PL \approx \alpha_4 \)
COMPARING GROUPS

Measurements for condition $A$: $a = [a_1, a_2, ..., a_m]$
Measurements for condition $B$: $b = [b_1, b_2, ..., b_n]$

Can we use a linear model to compare $A$ and $B$?

$x \sim \alpha_A 1_{\text{cond}=A} + \alpha_B 1_{\text{cond}=B}$

$x \sim \alpha + \beta 1_{\text{cond}=B}$

for an indiv. of type $A$: $x \sim \alpha$
for an indiv. of type $B$: $x = \alpha + \beta$
MODEL DIAGNOSIS

After we fit a model, how do we learn if it’s a good fit?
1. Evaluate its log likelihood
2. Hypothesis testing [next week]
3. Eyeball it!

Find the prediction error for each datapoint, and plot it every way we can think of

Find the log likelihood of each datapoint, and showcase some datapoints with very low or very high likelihood
If we hadn’t thought to include climate change in our temperature model ...

\[ \text{temp} \approx \alpha + \beta \sin(2\pi(t + \phi)) \]

\[ \text{temp} = \alpha + \beta \sin(2\pi(t + \phi)) + \varepsilon \]

This suggests a revised model ...

\[ \text{temp} = \alpha' + \beta' \sin(2\pi(t + \phi)) + \gamma t + \varepsilon \]
Q. Should we just keep adding more and more features to our model?

A. No. If we did, we’d overfit.