3 Fundamental Algorithms (part 2 of 5)

Logistic Regression

Logistic regression models the probability of 1=*success*, $P(y_i = 1)$, in a binary classification problem where the other outcome is 0=*failure*.

- If we are interested in $\hat{P}(y_i = 1) \in [0, 1]$, logistic regression does regression.
- If we add a decision threshold $T \in [0, 1]$ (like $T = \frac{1}{2}$) and say $\hat{P}(y_i = 1) > T$ indicates a label $\hat{y}_i = 1$, logistic regression does classification (where $\hat{y}_i \in \{0, 1\}$).

e.g. Applications include:

- A weather forecaster wants to classify a day, given conditions the day before, as "rain" or "no rain" and also give the probability of "rain".
- A doctor wants to classify a patient, given symptoms, as "sick with disease X" or "not disease X" and also give the probability of "sick with disease X".

Linear regression fails here because any line (or hyperplane) with nonzero slope spans $[-\infty, \infty]$:

- We want to model a probability: $p = P(y_i = 1) \in [0, 1]$ (the interval).
- Map the probability to an odds: $\frac{p}{1-p} \in \mathbb{R}_{>0}$ (draw for $p \in [0, 1)$).
- Map the odds to a log odds: $\ln \frac{p}{1-p} \in \mathbb{R}$ (draw $\ln x$ for $x \in \mathbb{R}_{>0}$).
- Require the log odds to be linear: $\ln \frac{p}{1-p} = \mathbf{wx} + b$ (draw $wx + b$ for 1D $x \in \mathbb{R}$).

  Solve for p:

$$
\begin{aligned}
\implies \frac{p}{1-p} &= e^{\mathbf{wx}+b} \\
\implies p(1 + e^{\mathbf{wx}+b}) &= e^{\mathbf{wx}+b} \\
\implies p &= \frac{e^{\mathbf{wx}+b}}{1 + e^{\mathbf{wx}+b}} \\
&= \frac{1}{1 + e^{-(\mathbf{wx}+b)}}
\end{aligned}
$$

Logistic function

The *standard logistic function* (or *sigmoid function*) is $\boxed{\sigma(t) = \dfrac{1}{1+e^{-t}}}$.

(Draw for $t \in \mathbb{R}$).

The logistic regression model is $\boxed{\hat{P}_{\mathbf{w},b}(y=1|\mathbf{x}) = f_{\mathbf{w},b}(\mathbf{x}) = \sigma(\mathbf{wx}+b) = \dfrac{1}{1+e^{-(\mathbf{wx}+b)}}}$:

- We can classify to $\{0,1\}$ by the rule $\text{label}(\mathbf{x}) = \begin{cases} 1 \text{ (success), if } f_{\mathbf{w},b}(\mathbf{x}) > T \text{ for some threshold } T \\ 0 \text{ (failure), otherwise} \end{cases}$
- For 1D $\mathbf{x}$, we have $f_{w,b}(x)$ (scalar w, b, and x):
  - The midpoint of the logistic curve is at $\left(-\frac{b}{w}, \frac{1}{2}\right)$, so for $w > 0$, increasing b shifts left.
  - Increasing w steepens the curve.

Maximum likelihood estimation

We seek $\mathbf{w}^*$ and b^* that maximize the *likelihood* (or probability) of the data.

The likelihood of a single example $(\mathbf{x}, y)$ is

$$l_{\mathbf{w},b}(\mathbf{x}, y) = \begin{cases} P(y=1|\mathbf{x}), \text{ if } y=1 \\ \quad = f_{\mathbf{w},b}(\mathbf{x}) \\ \\ P(y=0|\mathbf{x}), \text{ if } y=0 \\ \quad = [1 - f_{\mathbf{w},b}(\mathbf{x})] \end{cases}$$
$$= f_{\mathbf{w},b}(\mathbf{x})^y \left[1 - f_{\mathbf{w},b}(\mathbf{x})\right]^{1-y}$$

The likelihood of all the training examples is $L_{\mathbf{w},b} = \prod_{i=1}^{N} l_{\mathbf{w},b}(\mathbf{x_i}, y_i)$.

A product of probabilities can cause underflow, differentiating is easier with a sum than a product, and ln() is strictly increasing, so instead we maximize the *log likelihood.* Or, equivalently, we minimize the negative log likelihood:

$$\begin{aligned}
-\ln L_{\mathbf{w},b} &= -\ln \prod_{i=1}^{N} l_{\mathbf{w},b}(\mathbf{x_i}, y_i) \\
&= -\sum_{i=1}^{N} \ln l_{\mathbf{w},b}(\mathbf{x}_i, y_i) \\
&= -\sum \ln \left(f_{\mathbf{w},b}(\mathbf{x}_i)^{y_i} \left[1 - f_{\mathbf{w},b}(\mathbf{x}_i)\right]^{1-y_i} \right) \\
&= -\sum \left(y_i \ln f_{\mathbf{w},b}(\mathbf{x}_i) + (1 - y_i) \ln \left[1 - f_{\mathbf{w},b}(\mathbf{x}_i)\right] \right)
\end{aligned}$$

Recall: $f_{\mathbf{w},b}(\mathbf{x}) = \dfrac{1}{1 + e^{-(\mathbf{wx}+b)}}$.

There are not closed-form expressions for the minimizing $\mathbf{w}^*$ and b^*, so we use numerical methods.

e.g. Find partial derivatives and use gradient descent, coming soon.

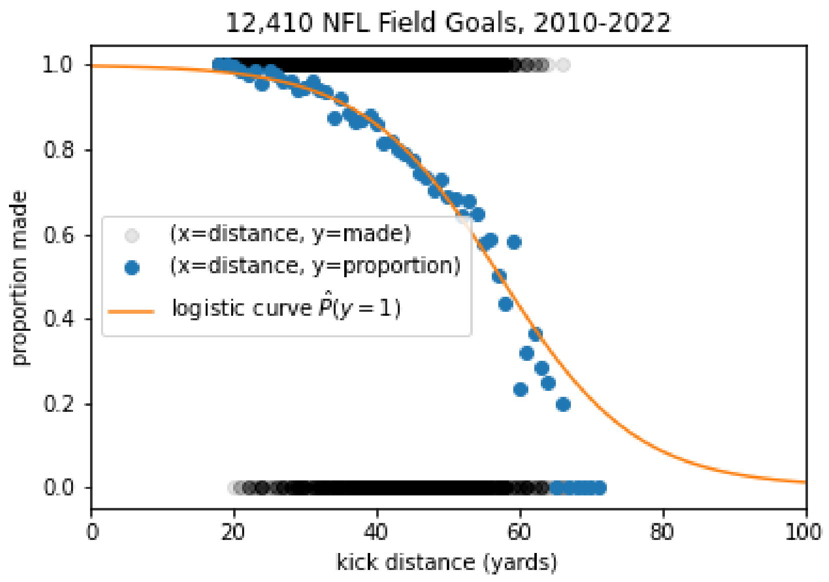
Regularization

To avoid overfitting the data, we can include a (L2) *regularization* term $\frac{1}{2}||\mathbf{w}||^2$ that penalizes any large coefficient w_i. Then, for some constant $C > 0$, we minimize the cost function $\frac{1}{2}||\mathbf{w}||^2 + C\,(-\ln L_{\mathbf{w},b}) =$

$$\boxed{\frac{1}{2}||\mathbf{w}||^2 + C \left[-\sum_{i=1}^{N} \left(y_i \ln f_{\mathbf{w},b}(\mathbf{x}_i) + (1 - y_i) \ln \left[1 - f_{\mathbf{w},b}(\mathbf{x}_i)\right] \right) \right]}.$$

Increasing C emphasizes fitting the data. Decreasing C prevents overfitting. (We should standardize features, coming in §8, to use regularization best.)

A helpful figure



Python

- `from sklearn import linear_model` loads the `linear_model` module
- `model = linear_model.LogisticRegression(C=1)` gives the model.
- `model.fit(X, y)` fits the model to array $\texttt{X}_{N\times D}$ and $\texttt{y}_{N\times 1}$. We can use $y_i \in \{0,1\}$ or $y_i \in \{-1,1\}$ (or other $\{y_i\}$; `model.fit()` models the probability of the largest value).
- `model.coef_` gives $\mathbf{w}^*$ and `model.intercept_` gives b^*
- `model.predict_proba(X)[:, 1]` gives probabilities $\{\hat{P}(y_i=1)\}$ (`[:, 0]` gives $\{\hat{P}(y_i=0)\}$)
- `model.predict(X)` uses decision threshold $T=\frac{1}{2}$ to give predictions $\{\hat{y}_i\}$ for examples in `X`
- `model.score(X, y)` gives the average accuracy of the model on `X` with respect to `y`

To learn more:

- User guide:

  `https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`
- Reference manual:

  `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`
- Example: See 7 lines of code after "Example" on the manual page.