Week 3: Classification

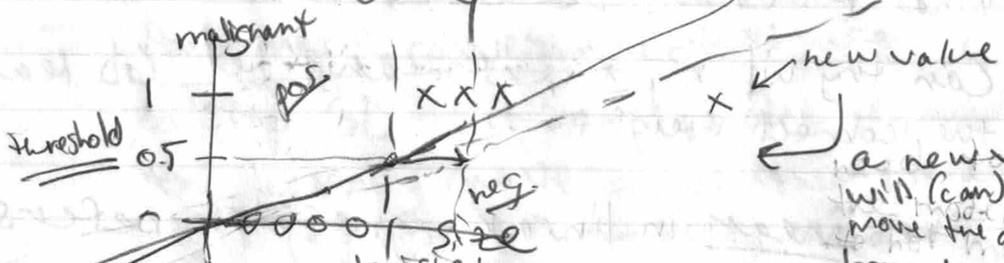
LOGISTIC REGRESSION

(Note: classes categories)

Recall classification examples:
- spam detection
- fraudulent transactions
- malignant tumors

} binary: yes (pos. class) or no (neg. class)

Linear regression?
- Predicts a continuous line of numerical values.
- Pick some threshold?
  → With one additional value you might need to drastically change the decision boundary



decision boundary: $< \to 0$
$> \to 1$

a new value will (can) significantly move the decision boundary

- Can work, but often not well!

Logistic regression
- Most used alg. for classification.
  - Fits an S-shaped curve b/w 0 and 1.
  - Predicts probabilities

Sigmoid function is a class of mathematical fn. that ~~always outputs~~ b/w 0 and 1.

Eg. $\boxed{g(z) = \frac{1}{1+e^{-z}}}$ $\lim_{z \to k} g(z) = \begin{cases} 1 & \text{if } k = \infty \\ 0 & \text{if } k \to -\infty \end{cases}$

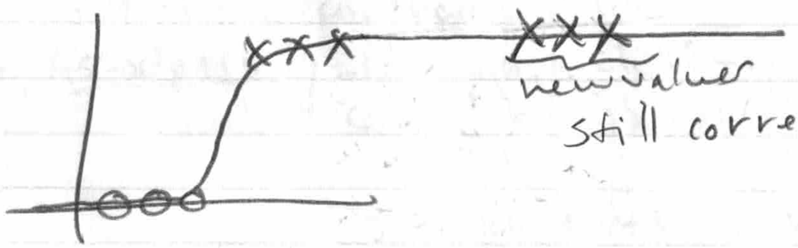
- Constructing logit:

$$z = \vec{w} \cdot \vec{x} + b \quad \text{(linear)}$$

$$g(z) = \frac{1}{1+e^{-z}} \quad \text{(sigmoid)}$$

$$\boxed{f_{\vec{w},b}(\vec{x}) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}}$$ = prob. that $\vec{x}$ belongs to class 1

- Handles additional outlier data very well: the model still makes good predictions w/out needing major changes

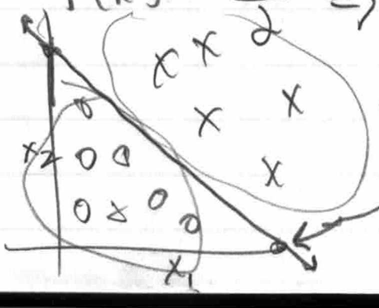


new values still correct!

Decision Boundary

Logistic function outputs probability that input belongs to class 1. We pick a threshold (0.5 e.g.) above which we assign the input to 1 and below to 0. Working back to the input gives a decision boundary.

$$f(\vec{x}) = \frac{1}{2} \Rightarrow e^{-(\vec{w}\cdot\vec{x}+b)} = 1 \Rightarrow \boxed{\vec{w}\cdot\vec{x}+b=0}$$

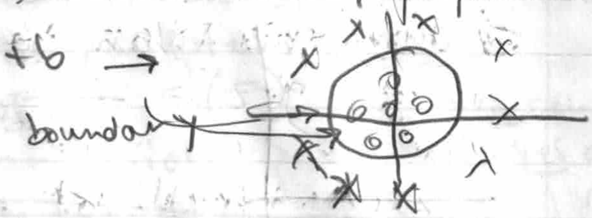


$w_1x_1 + w_2x_2 + b = 0$

this is a hyperplane (e.g. line for $\vec{x} \in \mathbb{R}^2$)

But using synthetic features we can add nonlinearity (i.e. feature crosses, polynomial reg.)

$$z = w_1 x_1^2 + w_2 x_2^2 + b \rightarrow$$

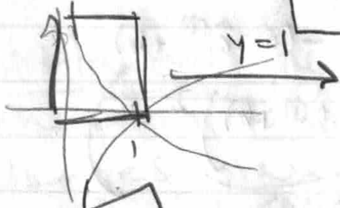


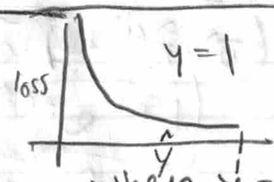
So logistic regression can learn very complex decision boundaries.

LOGISTIC LOSS

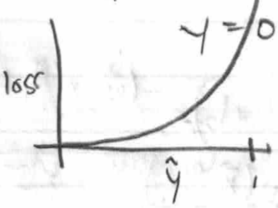
Using the squared-error cost function $\frac{1}{m}\sum(\hat{y}-y)^2$ produces a non-convex loss fn. when used w/ logistic fn $\frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}}$

Logistic loss $\boxed{L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y_i\right) = \begin{cases} -\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right), & y_i = 1 \\ -\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right), & y_i = 0 \end{cases}}$





when $y=1$, loss is high when $\hat{y}$ is close to 0 and low when close to 1



loss $\to 0$ as $\hat{y} \to 0$
$\to \infty$ as $\hat{y} \to 1$

where did this come from? Maximum likelihood optimization.

This is convex and so we can use grad. descent

$$J(\vec{w},b) = \frac{1}{m}\sum_{i=1}^{m} L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y_i\right)$$

Note: Can collapse L into a single equation using factors y and $1-y$:

$$L\left(f_{\vec{w},b}(\vec{x}^{(i)}), y_i\right) = -y\log\left[f_{\vec{w},b}(\vec{x}^{(i)})\right] - (1-y)\log\left[1-f_{\vec{w},b}(\vec{x}^{(i)})\right]$$

and this is more convenient for G.D.

Cost Fn: $$\boxed{J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y_i)\log\left(1-f_{\vec{w},b}(\vec{x}^{(i)})\right)\right]}$$

Recall $\frac{d}{dx}\log(x) = \frac{1}{x}$

$\frac{d}{dx} f(g(x)) = f'(g(x))\, g'(x)$

Gradient Descent for logistic regression

$$\frac{\partial}{\partial w_j} J(\vec{w},b) = \frac{-1}{m}\sum_{i=1}^{m}\left(\frac{y_i}{f_{\vec{w},b}(\vec{x}^{(i)})} f'_{\vec{w},b}(\vec{x}^{(i)}) - \frac{(1-y_i)}{(1-f_{\vec{w},b}(\vec{x}^{(i)}))} f'\right)$$

$$\left(\frac{\partial}{\partial w_j} f_{\vec{w},b}(\vec{x}^{(i)}) = \frac{\partial}{\partial w_j} \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}^{(i)}+b)}}\right.$$

$$= \frac{-1}{(1+e^{-(\vec{w}\cdot\vec{x}^{(i)}+b)})^2}\left(-(\vec{w}\cdot\vec{x}^{(i)}+b)\, e^{-(\vec{w}\cdot\vec{x}^{(i)}+b)}\right)$$

$$= \left.\frac{(\vec{w}\cdot\vec{x}^{(i)}+b)\, e^{-(\vec{w}\cdot\vec{x}^{(i)}+b)}}{\underbrace{(1+e^{-(\vec{w}\cdot\vec{x}^{(i)}+b)})^2}_{f_{\vec{w},b}(\vec{x}^{(i)})}}\right)$$

$$= \frac{1}{m}\sum\left(\frac{y_i\left((\vec{w}\cdot\vec{x}^{(i)}+b)e^{-(\vec{w}\cdot\vec{x}^{(i)}+b)}\right)}{f_{\vec{w},b}(\vec{x}^{(i)})} - \frac{(1-y_i)(\vec{w}\cdot\vec{x}^{(i)}+b)e^{-(\vec{w}\cdot\vec{x}+b)}}{1-f_{\vec{w},b}(\vec{x}^{(i)})}\right)$$

yuck

$\downarrow$

$$\boxed{\begin{aligned}\frac{\partial}{\partial w_j} J(\vec{w},b) &= \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y_i\right) x_j^{(i)} \\ \frac{\partial}{\partial b} J(\vec{w},b) &= \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y_i\right)\end{aligned}}$$

(Note: This gradient equation form is the same as for lin. reg, note that $f_{\vec{w},b}$ is very different.)

OVERFITTING

(Bias/Variance tradeoff)

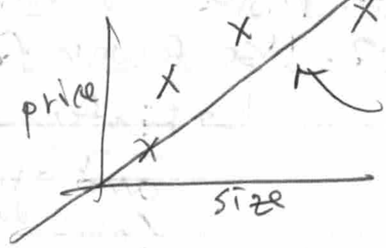
Regression example:

Bias: Algorithm is not capturing the patterns in the data well.

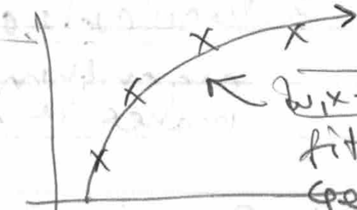
Variance: The amount the predicted function changes when the training data is slightly different.

Goal: Minimize both bias AND variance

w_1x+b

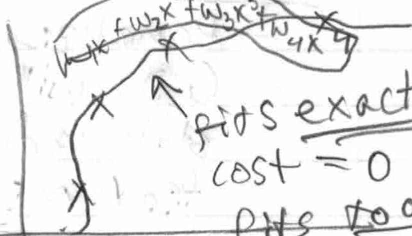


underfitting the training set = high bias



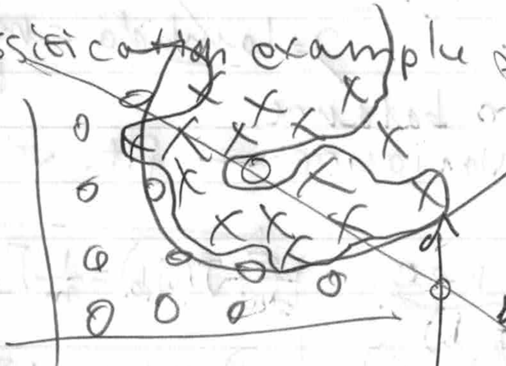
$w_1x + w_2x^2 + b$ fits well! generalizable

$w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$



fits exactly, cost = 0
~~fits too~~ well
⇒ overfit
⇒ high variance

Classification example



pretty good even if not cost=0

decision boundary too simple → underfit

overfit: won't predict well on real data

Generalizable = will make good predictions on unseen data.

Overfitting = capturing noise patterns in training set

Underfitting = failing to capture real patterns

REGULARIZATION

How to address overfitting?

1. More training data
2. Use fewer features
   - ↳ Feature Selection: use analysis and intuition to use fewer.
     Con: throwing away possible patterns.
3. Regularization: Encourages training algorithm to use smaller weights — makes it easier to keep all features.

Cost fn. w/ regularization

- Penalize all weights in the cost fn.
- G.D. will learn smaller weights while still reducing cost → weights for unhelpful features will be small.

$$J(\vec{w},b) = \underbrace{\boxed{\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}}_{\text{MSE term}} + \underbrace{\boxed{\frac{\lambda}{2m}\sum_{i=1}^{n} w_j^2}}_{\text{reg. term}}$$

λ = lambda = regularization parameter

- λ chosen to balance bias and variance — fit. vs. overfitting

Regularized lin. reg.
(recall: don't reg. b)

$$\frac{\partial}{\partial w} J(\vec{w},b) = \frac{1}{m}\sum\left(\hat{y}^{(i)} - y^{(i)}\right)^2 x_j^{(i)} + \frac{\lambda}{m} w_j$$

$$\frac{\partial}{\partial b} J(\vec{w},b) = \frac{1}{m}\sum\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

Intuition: the added $\left(\frac{\lambda}{m} w_j\right)$ term results in slightly decreasing w_j further on each step by a fixed amount.

Logistic. Also adds an additional $\left(\frac{\lambda}{m} w_j\right)$ term in $\frac{\partial J}{\partial w}$ in the summation.