Classification problems. y is discreet, either 0 or 1. Example: spam classification in emails, flag if online transaction is fraudulent, tumor. $y \in 0, 1$, where 0 is negative and 1 is positive.
Look at cancer problem. Fit a linear regression straight line, and set threshold classifier output $h_\theta(x) = 0.5$
$h_\theta(x) \geq 0.5$, predict $y = 1$, i.e. yes malignant
$h_\theta(x) < 0.5$, predict $y = 0$, i.e. not malignant.

Logistic regression. $0 \leq h_\theta(x) \leq 1$. $\boxed{h_\theta(x) = g(\theta^T x)}$, where $g(z) = \frac{1}{1+e^{-z}}$. g is called the sigmoid or logistic function. Then,

$$\boxed{h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}}$$

Interpretation of hypothesis output. $h_\theta(x)$ is estimated probability that $y = 1$ on input x.

Example: if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \frac{1}{\text{tumor size}}$, we get $h_\theta(x) = 0.7$. Tell patient that 70% chance that tumor being malignant.

$\boxed{h_\theta(x) = \mathbb{P}(y = 1|x; \theta)}$, probability that $y = 1$ given x parameterized by θ.

Decision Boundary. Suppose predict $y = 1$ if $h_\theta(x) \geq 05$, predict $y = 0$ if $h_\theta(x) < 0.5$. For sigmoid function, $g(z) \geq 0.5$ when $z \geq 0$, cross 0.5 in when $z = 0$. This implies that

$$h_\theta(x) = g(\theta^T x) \geq 0.5 \implies \theta^T x \geq 0$$

and

$$h_\theta(x) = g(\theta^T x) < 0.5 \implies \theta^T x < 0$$

Example: $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$. Let $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$, predict $y = 1$ if $-3 + x_1 + x_2 \geq 0 \implies x_1 + x_2 \geq 3$. Note that we obtain this from $\theta^T x$. When graphing, everything greater than 3 corresponds to those that have $y = 1$. The line is called *Decision Boundary.*

Non-linear decision boundaries. Negatives around the origin, positive around. $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$. Set $\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$. Gives:

predict $y = 1$ if $-1 + x_1^2 + x_2^2 \geq 0 \implies x_1^2 + x_2^2 \geq 1$. This is equation of a circle.

Example. $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \ldots)$

Cost Function. Training set: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$. m examples, where feature vector $x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$. $x_0 = 1, y \in 0, 1$. $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$. How to choose θ?

Cost function
linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^m cost(h_\theta(x^i), y^{(i)})$ where $cost(h_\theta(x^i), y^{(i)}) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$.
The cost function for logistic regression is non-convex, jigsaw convex shape.

Logistic Regression Cost Function. $cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x))\text{if} y = 1 \\ -log(1 - h_\theta(x))\text{if } y = 0 \end{cases}$.

We look at the graph of $-log$ for the case that $y = 1$. $Cost = 0$ if $y = 1, h_\theta(x) = 1$, but as $h_\theta(x) \to 0, cost \to \infty$.
Captures intuition that if $h_\theta(x) = 0$, i.e. predict $\mathbb{P}(y = 1|x; \theta)$, but $y = 1$ we will penalize learning algorithm by a very large cost.

For the case that $y = 0$, flip $-log$ vertically, i.e. y approaches there is a vertical asymptote at $x = 1$, and $y = 0, x = 0$. So, penalizes largely if $y = 1$ when it is actually zero.

Simplified Cost Function and Gradient Descent.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} cost(h_\theta(x^i), y^{(i)})$$

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x))\text{if} y = 1 \\ -log(1 - h_\theta(x))\text{if } y = 0 \end{cases}$$

Note $y = 0, 1$ always. Above is equivalent to

$$cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)log(1 - h_\theta(x))$$

We have,

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^{m} cost(h_\theta(x^{(i)}), y^{(i)}) \\ &= \boxed{-\frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} log h_\theta(x^{(i)}) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)})) \right]} \end{aligned}$$

principle of maximum likelihood, efficiently find parameters data for different models, convex.

Gradient Descent. To fit parameters θ, want to minimize $J(\theta)$.

Repeat: { $\theta_j - \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \implies \boxed{\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)} x_j^{(i)})}$ } (simultaneously update all θ_j) and $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \\ \theta_3 \\ \vdots \\ \theta_n \end{bmatrix}$

Note that boxed algorithm looks identical to linear regression. For linear regression, $h_\theta(x) = \theta^T x$. For logistic regression, $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$.

Advanced Optimization Algorithms. Cost function $J(\theta)$, want to minimize this. Given θ, we have code that can compute $J(\theta)$, $\frac{\partial}{\partial \theta_j} J(\theta)$.

1. Gradient descent: repeat $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j J(\theta)}$
2. Conjugate gradient
3. BFGS
4. L-BFGS

2-4 algorithms have many advantages: no need to manually pick α (has inner loop called line to pick different α for each iteration), often faster than gradient descent. disadvantages: more complex.

Example: $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$, $J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$. We have,

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$
$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

The code is as follows,

```
function[jVal, gradient] = costFunction(theta)
jVal = (theta(1) - 5)^2 + \ldots + (theta(2) - 5)^2;
gradient(1) = 2*(theta(1)-5);
gradient(2) = 2*(theta(2) - 5);
```

Call advanced optimization function called fminunc,

```
options = optimist{'GradObj', 'on', 'MaxIter', '100'};
initialTheta = zeros(2,1)
[optTheta, functionVal, exitFlag] ...
        = fminunc(@costFunction, initialTheta, options);
```

Essentially, we will have,

```
function[jVal, gradient] = costFunction(theta)
```

$$\begin{aligned} jVal &= [J(\theta)] \\ gradient(1) &= \frac{\partial}{\partial \theta_0} J(\theta) \\ gradient(2) &= \frac{\partial}{\partial \theta_1} J(\theta) \\ &\vdots \\ gradient(n+1) &= \frac{\partial}{\partial \theta_n} J(\theta) \end{aligned}$$

Multiclass Classification. Email foldering/tagging: work, friends, family, hobby, where each is y. Medical diagram: not ill, cold, flu. Weather: Sunny, cloudy, rain, snow.

One-vs-all, three class case: $h_\theta^{(1)}(x), h^{(2)}(x), h^{(3)}(x)$. Train a logistic regression classifier $h_\theta^i(x)$ for each class i to predict the probability that $y = i$. On a new input x, to make a prediction, pick the class i that maximizes $h_\theta^{(i)} x$

Overfitting. Underfit, high bias: fitting straight line to data. Just right. Overfit, high variance: fit high order polynomial to data, $J(\theta) \approx 0$, but fails to generalize.

Address overfitting.

1. Reduce number of features. a) Manually select which features to keep b) model selection algorithm

2. Regularization. a) keep all the features, but reduce magnitude/values of parameters θ_j. b) works well when we have a lot of features, each of which contributes a bit to predicting y.

Regularization. Small rvalues for parameters $\theta_0, \theta_1, \ldots, \theta_n$. Have simpler hypothesis, less prone to overfitting.
Example (housing). Features: $x_1, x_2, \ldots, x_100$, parameters: $\theta_0, \theta_1, \theta_2 \ldots, \theta_100$.

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

Want to minimize $J(\theta)$. λ called regularization parameter, controls the goal of fitting well and overfitting.

What if λ is big? Penalize all variables, that is fitting $h_\theta(x) = \theta_0$ to data, thus under fit.

Gradient Descent for Regularized Linear Regression.

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$
$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$$

This is equivalent to

$$\theta_j = \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Normal Equation.

$$X = \begin{bmatrix}(x^{(1)})^T \\ \vdots \\ (x^{(m)})^T\end{bmatrix} \qquad y = \begin{bmatrix}y^{(1)} \\ \vdots \\ y^{(m)}\end{bmatrix}$$

θ that minimize $J(\theta)$:

$$\theta = (X^T X_\theta \begin{bmatrix}0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1\end{bmatrix})^{-1} X^T y$$

The matrix is $(n+1) \times (n+1)$.

Gradient Descent for Regularized Logistic Regression.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$
$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

But $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$.

Advanced optimization.

```
function[jVal, gradient] = costFunction(theta)
```

where $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$

$$JVal = J(\theta)$$
$$J(\theta) = \left[-\frac{1}{m} \sum_{i=1}^{m} y^{(i)} log h_\theta(x^{(i)}) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$
$$gradient(1) = \frac{\partial}{\partial \theta_0} J(\theta)$$
$$\frac{\partial}{\partial \theta_0} J(\theta = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$
$$gradient(2) = \frac{\partial}{\partial \theta_1} J(\theta)$$
$$\frac{\partial}{\partial \theta_1} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(1)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_1 \vdots$$
$$gradient(n+1) = \frac{\partial}{\partial \theta_n} J(\theta)$$