Machine Learning

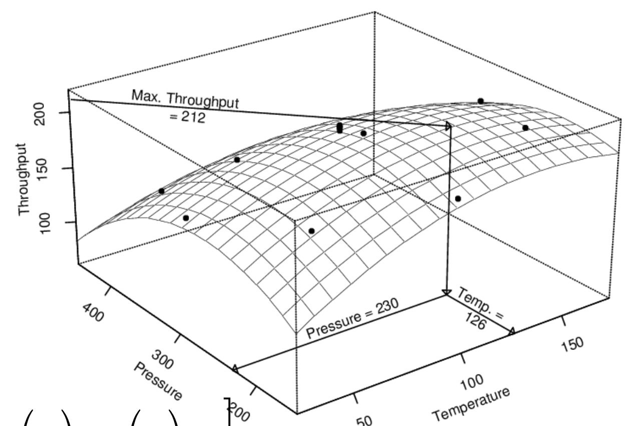
Topic 3

- Additive Models and Linear Regression
- Sinusoids and Radial Basis Functions
- Classification
- Logistic Regression
- Gradient Descent

Polynomial Basis Functions

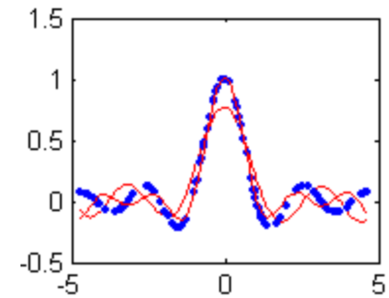
- To fit a P'th order polynomial function to multivariate data: concatenate columns of all monomials up to power P
- E.g. 2 dimensional data and 2nd order polynomial (quadratic)

$$\begin{bmatrix} x_1(1) & x_1(2) \\ \vdots & \vdots \\ x_i(1) & x_i(2) \\ \vdots & \vdots \\ x_N(1) & x_N(2) \end{bmatrix}$$



$$\begin{bmatrix} 1 & x_1(1) & x_1(2) & x_1(1)x_1(1) & x_1(1)x_1(2) & x_1(2)x_1(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_i(1) & x_i(2) & x_i(1)x_i(1) & x_i(1)x_i(2) & x_i(2)x_i(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N(1) & x_N(2) & x_N(1)x_N(1) & x_N(1)x_N(2) & x_N(2)x_N(2) \end{bmatrix}$$

Sinusoidal Basis Functions



- More generally, we don't just have to deal with polynomials, use any set of basis fn's:

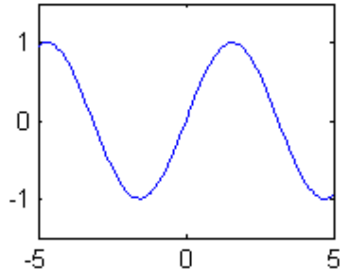
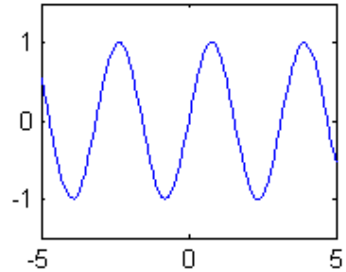
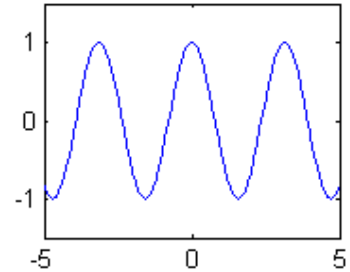
$$f\left(x;\theta\right) = \sum_{p=1}^{P} \theta_p \phi_p\left(x\right) + \theta_0$$

- These are generally called Additive Models
- Regression adds linear combinations of the basis fn's

- For example: Fourier (sinusoidal) basis

$$\phi_{2k}\left(x_i\right) = \sin\left(kx_i\right) \qquad \phi_{2k+1}\left(x_i\right) = \cos\left(kx_i\right)$$

- Note, don't have to be a basis per se, usually subset

$\theta_1 \times$  $+\theta_2 \times$  $+\theta_3 \times$ 

Radial Basis Functions

- Can act as prototypes of the data itself

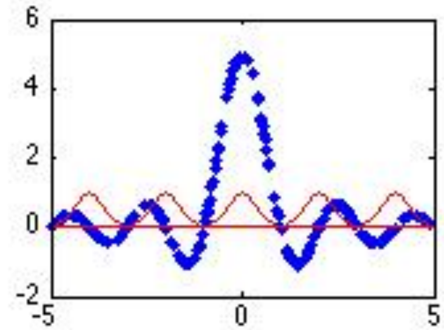
$$f\left(\mathbf{x};\theta\right) = \sum_{k=1}^{N} \theta_k \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{x}_k\right\|^2\right)$$

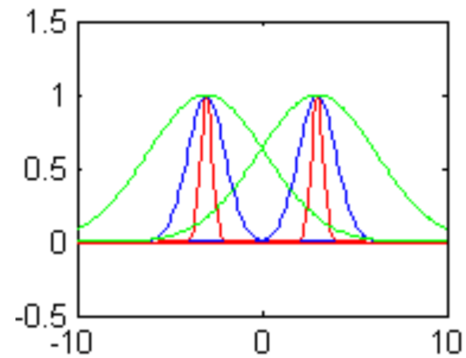
- Parameter σ = standard deviation
  σ^2 = covariance

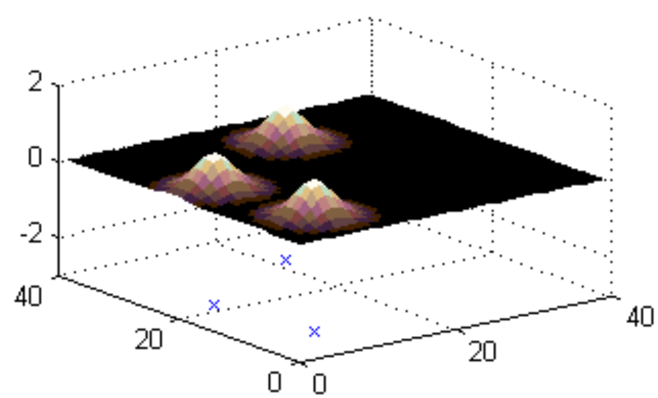
  controls how wide bumps are
  what happens if too big/small?

- Also works in multi-dimensions

- Called RBF for short







Radial Basis Functions

- Each training point leads to a bump function

$$f\left(\mathbf{x};\theta\right) = \sum_{k=1}^{N} \theta_k \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{x}_k\right\|^2\right)$$

- Reuse solution from linear regression: $\theta^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$
- Can view the data instead as X, a big matrix of size N x N

$$\mathbf{X} = \begin{bmatrix} \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_1-\mathbf{x}_1\right\|^2\right) & \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_1-\mathbf{x}_2\right\|^2\right) & \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_1-\mathbf{x}_3\right\|^2\right) \\ \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_2-\mathbf{x}_1\right\|^2\right) & \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_2-\mathbf{x}_2\right\|^2\right) & \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_2-\mathbf{x}_3\right\|^2\right) \\ \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_3-\mathbf{x}_1\right\|^2\right) & \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_3-\mathbf{x}_2\right\|^2\right) & \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_3-\mathbf{x}_3\right\|^2\right) \end{bmatrix}$$

- For RBFs, X is square and symmetric, so solution is just

$$\nabla_\theta R = 0 \rightarrow \mathbf{X}^T\mathbf{X}\theta = \mathbf{X}^T\mathbf{y} \rightarrow \mathbf{X}\theta = \mathbf{y} \rightarrow \theta^* = \mathbf{X}^{-1}\mathbf{y}$$

Evaluating Our Learned Function

- We minimized empirical risk to get θ^*
- How well does $f(x;\theta^*)$ perform on future data?
- It should *Generalize* and have low True Risk:

$$R_{true}\left(\theta\right) = \int P\left(x,y\right) L\left(y, f\left(x;\theta\right)\right) dx\, dy$$

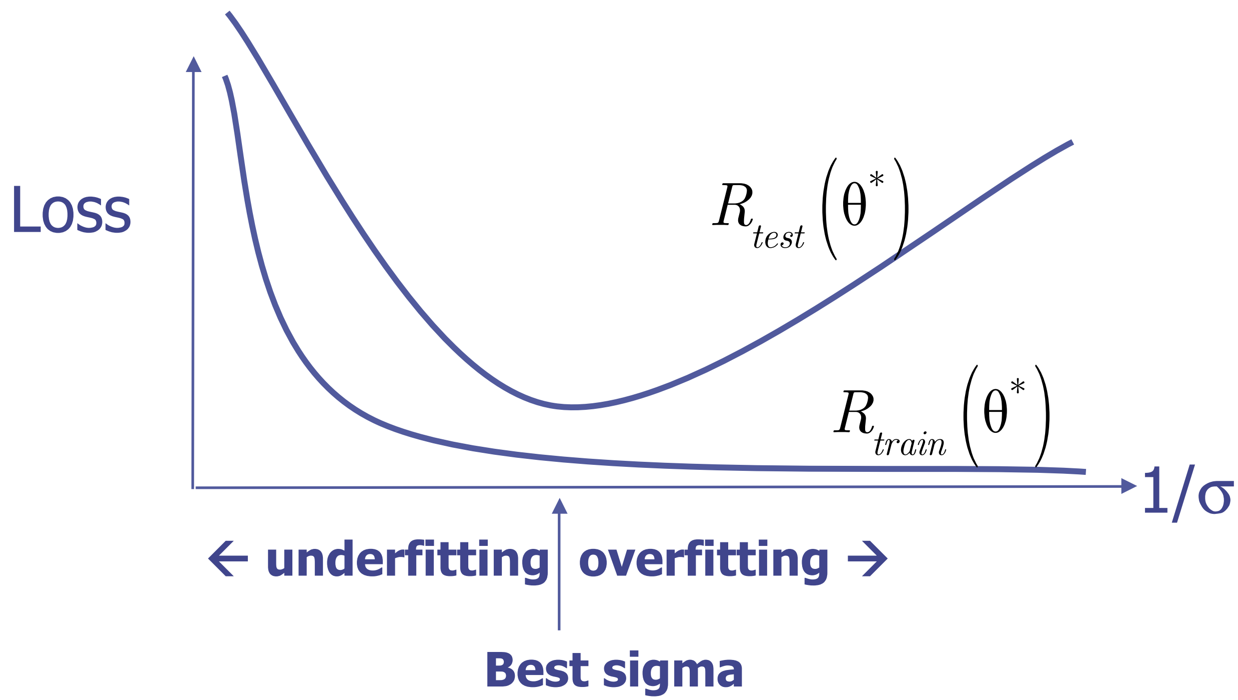
- Can't compute true risk, instead use Testing Empirical Risk
- We randomly split data into training and testing portions

$$\left\{\left(x_1,y_1\right),\ldots,\left(x_N,y_N\right)\right\} \qquad \left\{\left(x_{N+1},y_{N+1}\right),\ldots,\left(x_{N+M},y_{N+M}\right)\right\}$$

- Find θ^* with training data: $R_{train}\left(\theta\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(x_i;\theta\right)\right)$
- Evaluate it with testing data: $R_{test}\left(\theta\right) = \frac{1}{M}\sum_{i=N+1}^{N+M} L\left(y_i, f\left(x_i;\theta\right)\right)$

Crossvalidation

- Try fitting with different sigma radial basis function widths
- Select sigma which gives lowest $R_{test}(\theta^*)$



- Think of sigma as a measure of the simplicity of the model
- Thinner RBFs are more flexible and complex

Regularized Risk Minimization

- Empirical Risk Minimization gave overfitting & underfitting
- We want to add a penalty for using too many theta values
- This gives us the Regularized Risk

$$
\begin{aligned}
R_{regularized}\left(\theta\right) &= R_{empirical}\left(\theta\right) + Penalty\left(\theta\right) \\
&= \tfrac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(x_i;\theta\right)\right) + \tfrac{\lambda}{2N}\left\|\theta\right\|^2
\end{aligned}
$$

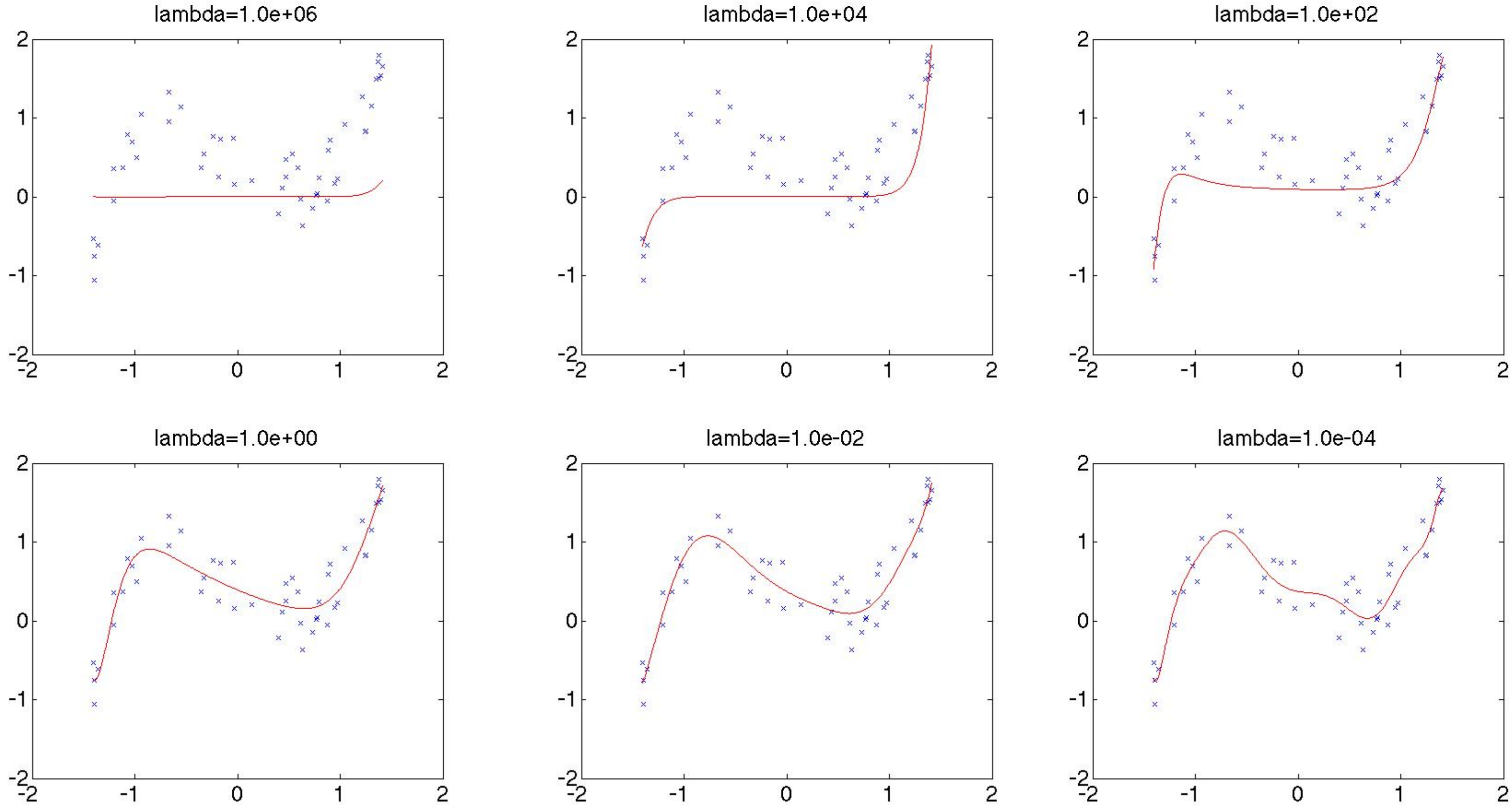
- Solution for Regularized Risk with Least Squares Loss:

$$
\nabla_\theta R_{regularized} = 0 \quad \Rightarrow \quad \nabla_\theta\left(\tfrac{1}{2N}\left\|\mathbf{y}-\mathbf{X}\theta\right\|^2 + \tfrac{\lambda}{2N}\left\|\theta\right\|^2\right) = 0
$$

$$
\theta^* = \left(\mathbf{X}^T\mathbf{X} + \lambda I\right)^{-1}\mathbf{X}^T\mathbf{y}
$$

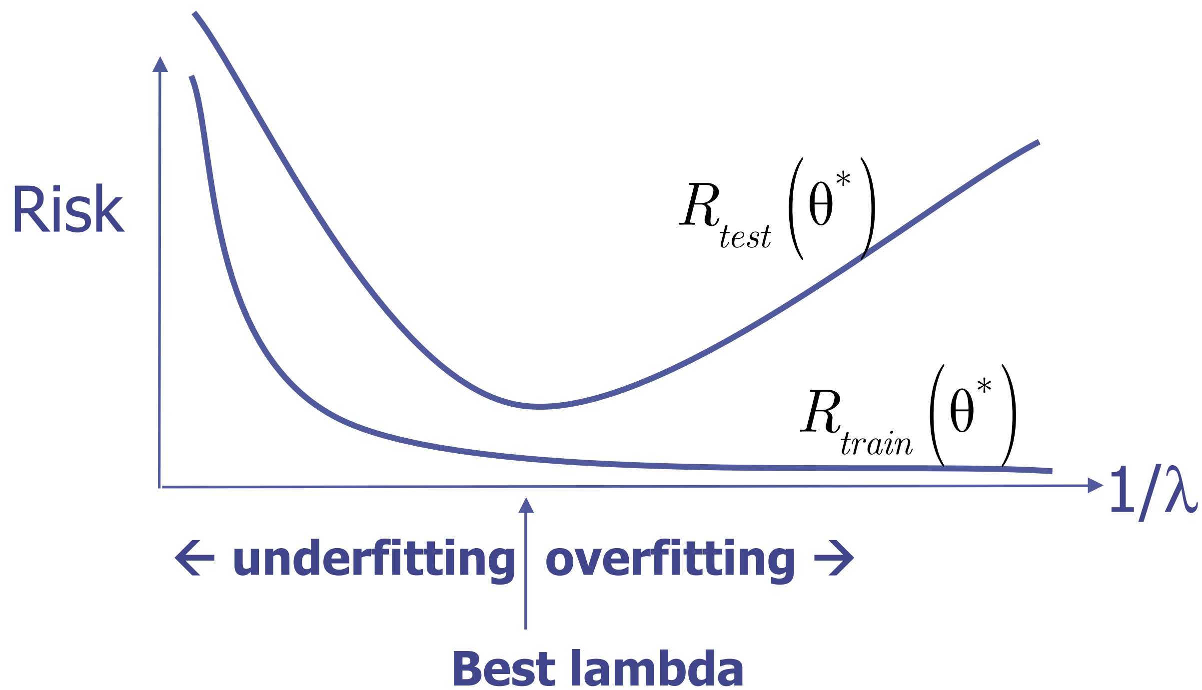
Regularized Risk Minimization

- Have D=16 features (or P=15 throughout)
- Try minimizing $R_{regularized}(\theta)$ to get θ^* with different λ
- Note that $\lambda=0$ give back Empirical Risk Minimization



Crossvalidation

- Try fitting with different lambda regularization levels
- Select lambda which gives lowest $R_{test}(\theta^*)$



- Lambda measures simplicity of the model
- Models with low lambda are more flexible

From Regression To Classification

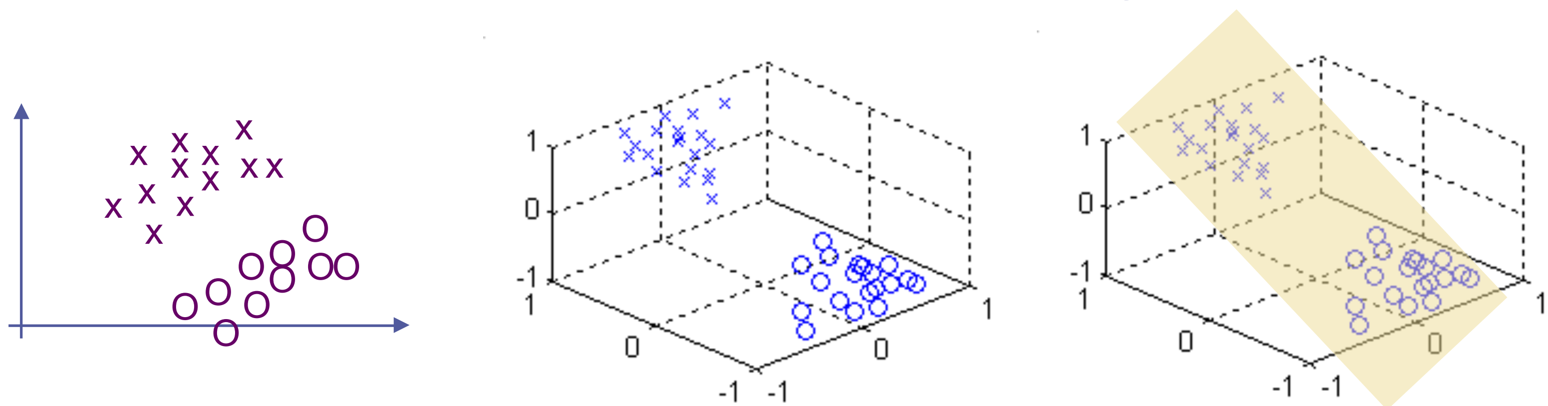
- Classification is another important learning problem

Regression $\mathcal{X} = \left\{ \left(\mathbf{x}_1, y_1\right), \left(\mathbf{x}_2, y_2\right), \ldots, \left(\mathbf{x}_N, y_N\right) \right\} \quad \mathbf{x} \in \mathbb{R}^D \quad y \in \mathbb{R}^1$

Classification $\mathcal{X} = \left\{ \left(\mathbf{x}_1, y_1\right), \left(\mathbf{x}_2, y_2\right), \ldots, \left(\mathbf{x}_N, y_N\right) \right\} \quad \mathbf{x} \in \mathbb{R}^D \quad y \in \left\{0, 1\right\}$

- E.g. Given x = [tumor size, tumor density]
  Predict y in {benign,malignant}

- Should we solve this as a least squares regression problem?



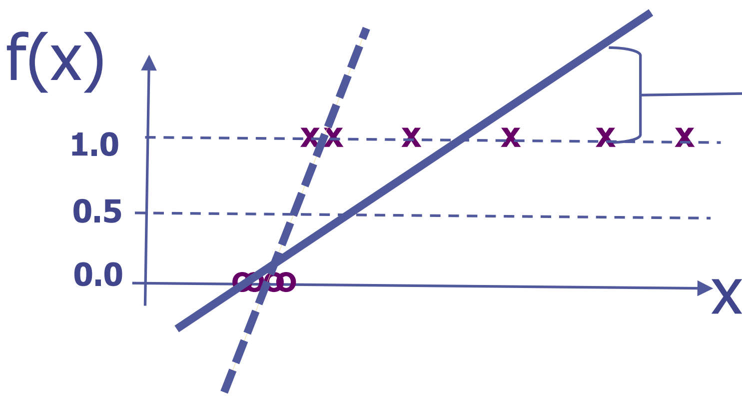
Classification vs. Regression

a) Classification needs binary answers like {0,1}
b) Least squares is an unfair measure of risk here
   e.g. Why penalize a correct but large positive y answer?
   e.g. Why penalize a correct but large negative y answer?

•Example: not good to use regression output for a decision

$f(x)>0.5 \rightarrow$ Class 1 $f(x)<0.5 \rightarrow$ Class 0

if $f(x)=-3.8$ & correct class=0, squared error penalizes it…



We pay a hefty squared error loss here even if we got the correct classification result. The thick solid line model makes two mistakes while the dashed model is perfect

Classification vs. Regression

We will consider the following four steps to improve from naïve regression to get better classification learning:

1) Fix functions f(x) to give binary output (logistic neuron)
2) Fix our definition of the Risk we will minimize so that we get good classification accuracy (logistic loss)

…and later on…

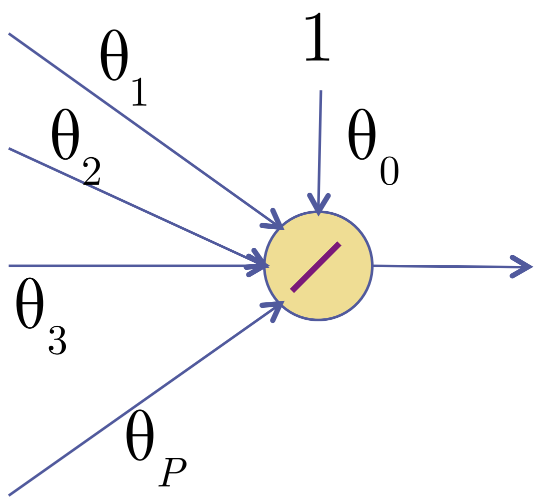
3) Make an even better fix on f(x) to binarize (perceptron)
4) Make an even better risk (perceptron loss)

Logistic Neuron (McCullough-Pitts)

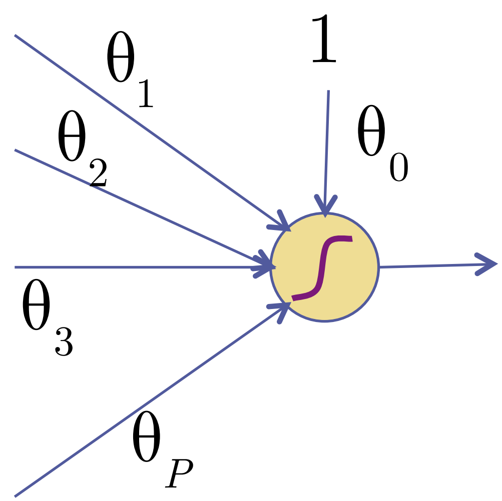
- To output binary, use squashing function g().

$$f\left(\mathbf{x};\theta\right) = \theta^T\mathbf{x}$$

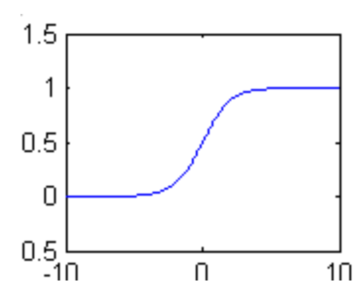
$$f\left(\mathbf{x};\theta\right) = g\left(\theta^T\mathbf{x}\right)$$
$$g\left(z\right) = \left(1 + \exp\left(-z\right)\right)^{-1}$$



Linear neuron



Logistic Neuron



- This squashing is called sigmoid or logistic function

Logistic Regression

- Given a classification problem with binary outputs

$$\mathcal{X} = \left\{ \left(\mathbf{x}_1, y_1\right), \left(\mathbf{x}_2, y_2\right), \ldots, \left(\mathbf{x}_N, y_N\right) \right\} \quad \mathbf{x} \in \mathbb{R}^D \quad y \in \left\{0,1\right\}$$

- Use this function and output 1 if f(x)>0.5 and 0 otherwise

$$f\left(\mathbf{x};\theta\right) = \left(1 + \exp\left(-\theta^T \mathbf{x}\right)\right)^{-1}$$

Short hand for Linear Functions

- What happened to adding the intercept?

$$f\left(\mathbf{x};\theta\right) = \theta^T\mathbf{x} + \theta_0$$

$$= \begin{bmatrix} \theta(1) \\ \theta(2) \\ \vdots \\ \theta(D) \end{bmatrix}^T \begin{bmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(D) \end{bmatrix} + \theta_0 = \begin{bmatrix} \theta_0 \\ \theta(1) \\ \theta(2) \\ \vdots \\ \theta(D) \end{bmatrix}^T \begin{bmatrix} 1 \\ \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(D) \end{bmatrix} = \vec{\theta}^T\vec{\mathbf{x}}$$

Logistic Regression

- Given a classification problem with binary outputs

$$\mathcal{X} = \left\{\left(\mathbf{x}_1, y_1\right), \left(\mathbf{x}_2, y_2\right), \ldots, \left(\mathbf{x}_N, y_N\right)\right\} \quad \mathbf{x} \in \mathbb{R}^D \quad y \in \left\{0,1\right\}$$

- Fix#1: use f(x) below, output 1 if f(x)>0.5 and 0 otherwise

$$f\left(\mathbf{x};\theta\right) = \left(1 + \exp\left(-\theta^T \mathbf{x}\right)\right)^{-1}$$

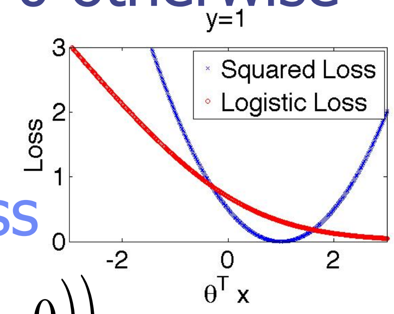
Logistic Regression

- Given a classification problem with binary outputs

$$\mathcal{X} = \left\{\left(\mathbf{x}_1, y_1\right), \left(\mathbf{x}_2, y_2\right), \ldots, \left(\mathbf{x}_N, y_N\right)\right\} \quad \mathbf{x} \in \mathbb{R}^D \quad y \in \left\{0,1\right\}$$

- Fix#1: use f(x) below, output 1 if f(x)>0.5 and 0 otherwise

$$f\left(\mathbf{x};\theta\right) = \left(1 + \exp\left(-\theta^T \mathbf{x}\right)\right)^{-1}$$



- Fix#2: instead of squared loss, use Logistic Loss

$$L_{log}\left(y, f\left(\mathbf{x};\theta\right)\right) = \left(y_i - 1\right)\log\left(1 - f\left(\mathbf{x};\theta\right)\right) - y_i \log\left(f\left(\mathbf{x};\theta\right)\right)$$

- This method is called Logistic Regression.
- But Empirical Risk Minimization has no closed-form sol'n:

$$R_{emp}\left(\theta\right) = \tfrac{1}{N}\sum_{i=1}^{N}\left(y_i - 1\right)\log\left(1 - f\left(\mathbf{x}_i;\theta\right)\right) - y_i \log\left(f\left(\mathbf{x}_i;\theta\right)\right)$$

Logistic Regression

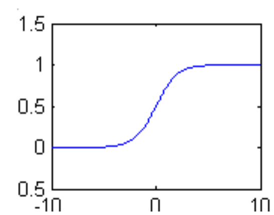
- With logistic squashing function, minimizing R(θ) is harder

$$R_{emp}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - 1)\log\left(1 - f(\mathbf{x}_i;\theta)\right) - y_i \log\left(f(\mathbf{x}_i;\theta)\right)$$

$$\nabla_\theta R = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1-y_i}{1-f(\mathbf{x}_i;\theta)} - \frac{y_i}{f(\mathbf{x}_i;\theta)}\right) f'(\mathbf{x}_i;\theta) = 0 \quad ???$$

- Can’t minimize risk and find best theta analytically!
- Let’s try finding best theta numerically.
- Use the following to compute gradient

$$f(\mathbf{x};\theta) = \left(1 + \exp\left(-\theta^T \mathbf{x}\right)\right)^{-1} = g\left(\theta^T \mathbf{x}\right)$$

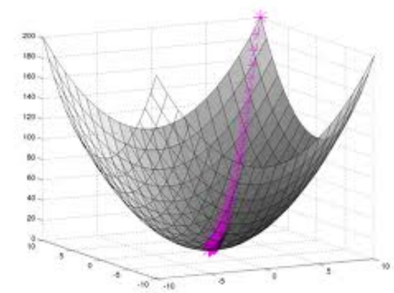


- Here, g() is the logistic squashing function

$$g(z) = \left(1 + \exp(-z)\right)^{-1} \qquad g'(z) = g(z)\left(1 - g(z)\right)$$

Gradient Descent

- Useful when we can't get minimum solution in closed form
- Gradient points in direction of fastest increase
- Take step in the opposite direction!



- Gradient Descent Algorithm

choose scalar step size η, & tolerance ε
initialize θ^0 = small random vector

$$\theta^1 = \theta^0 - \eta \nabla_\theta R_{emp}\Big|_{\theta^0}, \quad t = 1$$

while $\left\|\theta^t - \theta^{t-1}\right\| \geq \in$ $\{$

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta R_{emp}\Big|_{\theta^t}, \quad t = t + 1$$

$\}$

- For appropriate η, this will converge to local minimum

Logistic Regression

- Logistic regression gives better classification performance
- Its empirical risk is

$$R_{emp}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - 1)\log\left(1 - f(\mathbf{x}_i;\theta)\right) - y_i \log\left(f(\mathbf{x}_i;\theta)\right)$$

- This R(θ) is convex so gradient descent always converges to the same solution

- Make predictions using

$$f(\mathbf{x};\theta) = \left(1 + \exp\left(-\theta^T \mathbf{x}\right)\right)^{-1}$$

- Output 1 if f > 0.5
- Output 0 otherwise