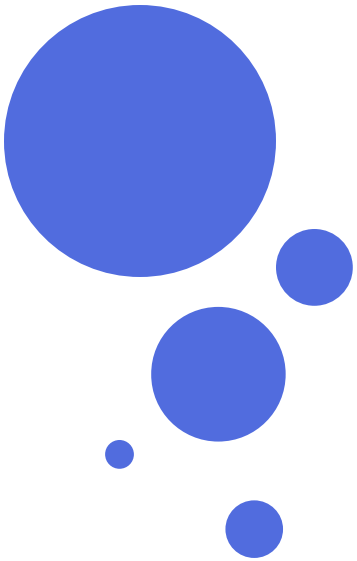
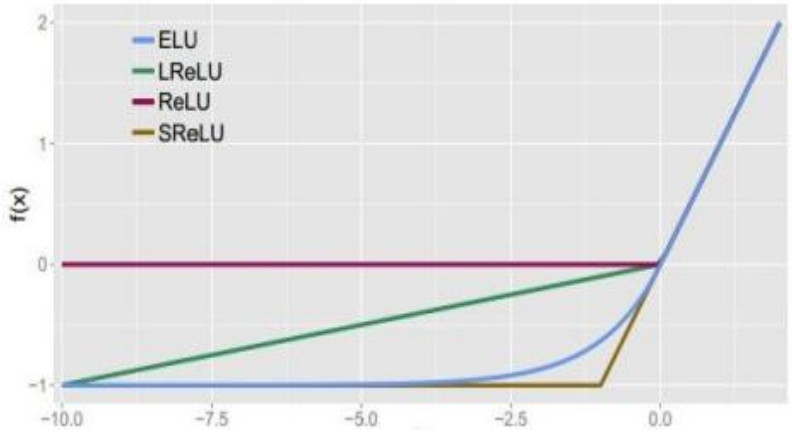
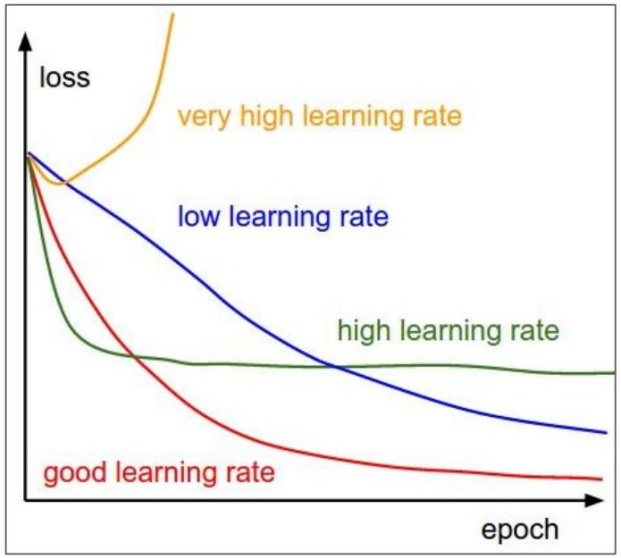


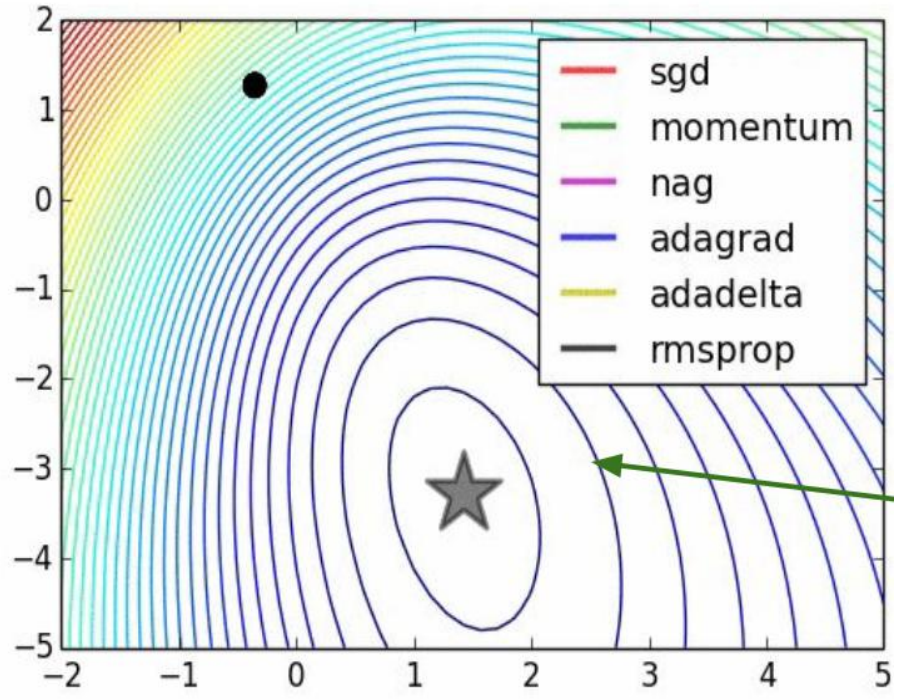
Lecture 20: Regression

Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at RPI.
Email: **longc3@rpi.edu**

Recap Previous Lecture







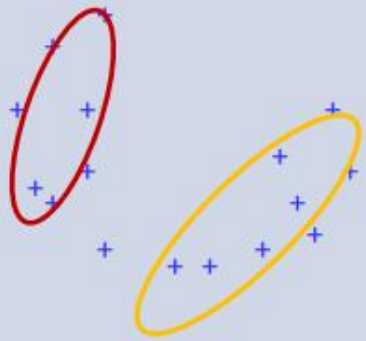
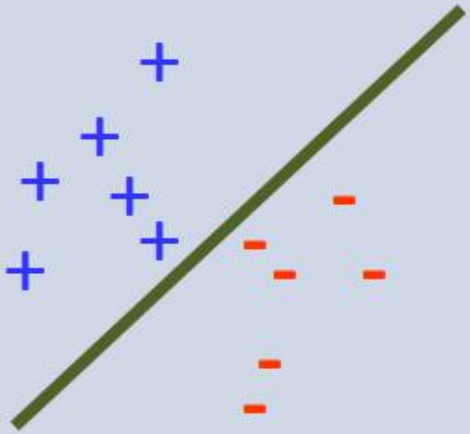
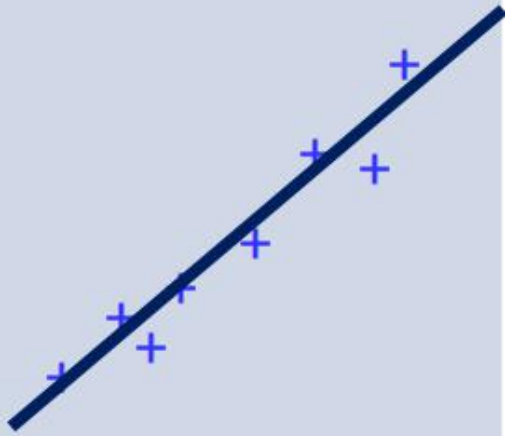
Outline

- Regression Overview
- Linear Regression
- Support Vector Regression
- Logistic Regression
- Deep Neural Network for Regression

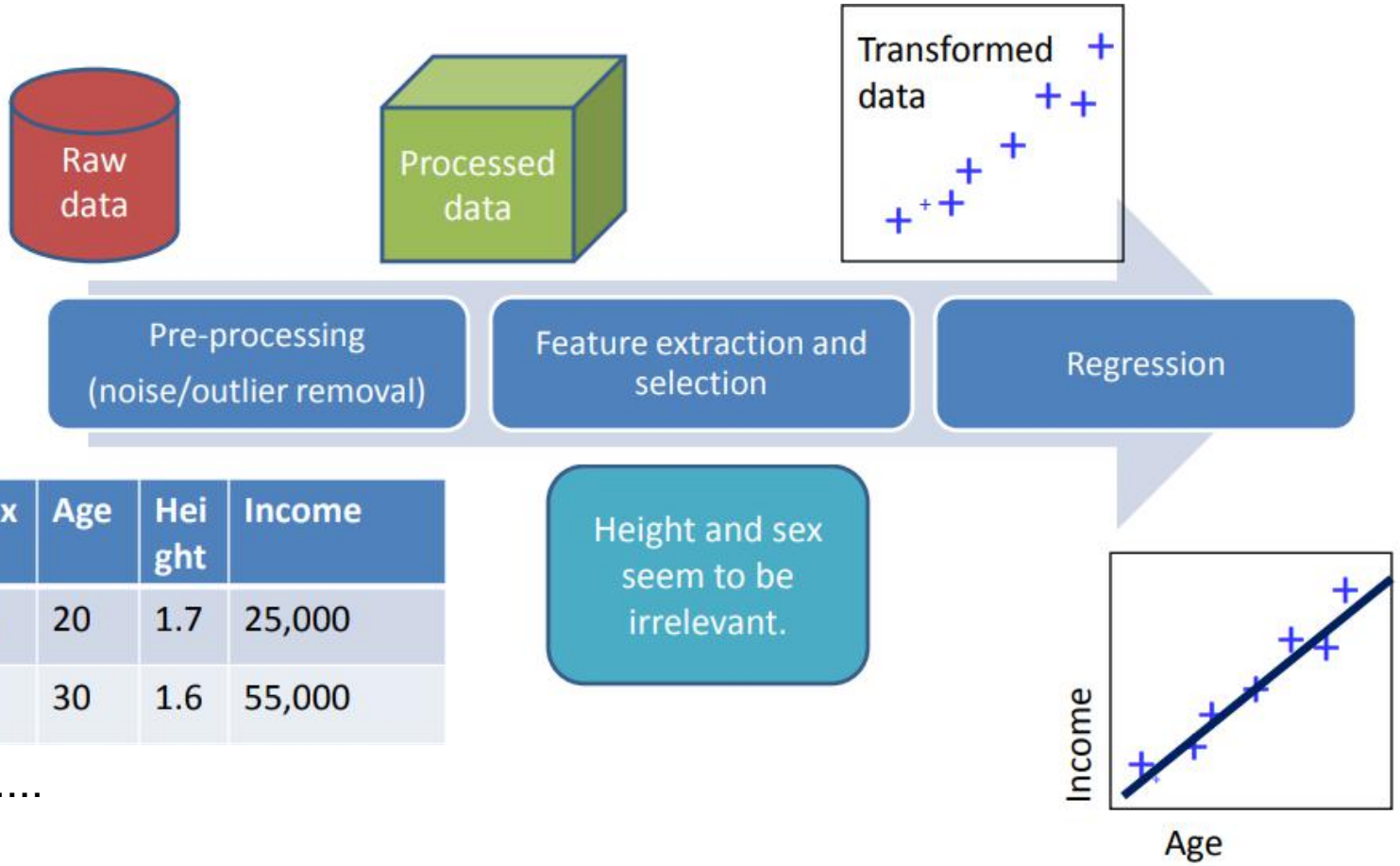
Outline

- **Regression Overview**
- Linear Regression
- Support Vector Regression
- Logistic Regression
- Deep Neural Network for Regression

Regression Overview

CLUSTERING	CLASSIFICATION	REGRESSION (THIS TALK)
		
K-means Hierarchy clustering Gaussian Mixture Model	• Decision tree • Linear Discriminant Analysis • Neural Networks • Support Vector Machines • Boosting	• Linear Regression • Support Vector Regression • Logistic Regression • Neural Networks
Group data based on their characteristics	Separate data based on their labels	Find a model that can explain the output given the input

One Example



Sex	Age	Height	Income
M	20	1.7	25,000
F	30	1.6	55,000

.....

Evaluation Metrics

- Root mean-square error (RMSE)
  - RMSE is a popular formula to measure the error rate of a regression model. However, it can only be compared between models whose errors are measured in the same units.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}}$$

a = actual target

p = predicted target

- Mean absolute error (MAE)
  - MAE has the same unit as the original data, and it can only be compared between models whose errors are measured in the same units. It is usually similar in magnitude to RMSE, but slightly smaller.

$$MAE = \frac{\sum_{i=1}^{n}|p_i - a_i|}{n}$$

Evaluation Metrics

- Relative Squared Error (RSE)
  - Unlike RMSE, the relative squared error (RSE) can be compared between models whose errors are measured in the different units.

$$RSE = \frac{\sum_{i=1}^{n}(p_i - a_i)^2}{\sum_{i=1}^{n}(\bar{a} - a_i)^2}$$

mean value

- Relative Absolute Error (RAE)
  - Like RSE , the relative absolute error (RAE) can be compared between models whose errors are measured in the different units..

$$RAE = \frac{\sum_{i=1}^{n}|p_i - a_i|}{\sum_{i=1}^{n}|\bar{a} - a_i|}$$

Outline

- Regression Overview
- **Linear Regression**
- Support Vector Regression
- Logistic Regression
- Deep Neural Network for Regression

Linear Regression

- Given data with n dimensional variables and 1 target-variable (real number)

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$$

where $\mathbf{x} \in \Re^n, y \in \Re$

- The objective: Find a function f that returns the best fit.

$$f : \Re^n \rightarrow \Re$$

- Assume that the relationship between X and y is approximately linear. The model can be represented as (w represents coefficients and b is an intercept)

$$f(w_1, \ldots, w_n, b) = y = \mathbf{w} \cdot \mathbf{x} + b + \varepsilon$$

Linear Regression

- To find the best fit, we minimize the sum of squared errors –> Least square estimation

$$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - (\mathbf{w} \cdot \mathbf{x}_i + b))^2$$

- The solution can be found by solving

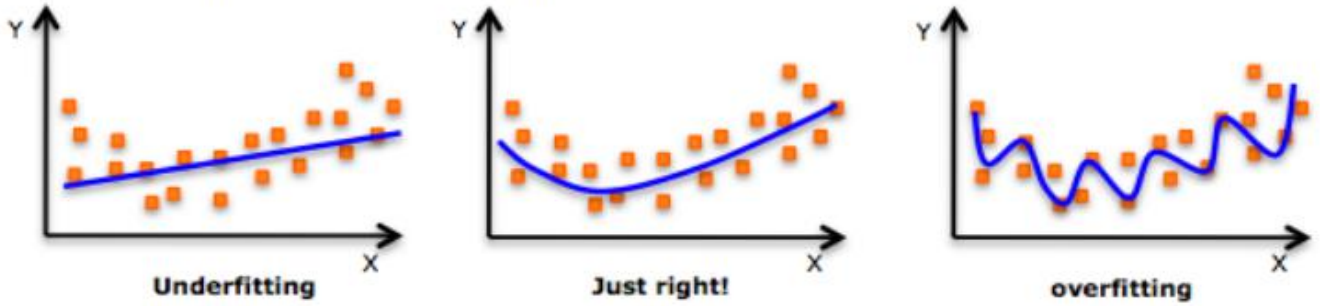
$$\hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$$

(By taking the derivative of the above objective function w.r.t.)

- In MATLAB, the back-slash operator computes a least square solution.

Linear Regression

$$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - (\hat{\mathbf{w}} \cdot \mathbf{x}_i + \hat{b}))^2$$



To avoid over-fitting, a regularization term can be introduced (minimize a magnitude of w)

– LASSO: $\min \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^{n} | w_j |$

– Ridge regression: $\min \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^{n} | \mathbf{w}_j^2 |$

Outline

- Regression Overview
- Linear Regression
- **Support Vector Regression**
- Logistic Regression
- Deep Neural Network for Regression

Support Vector Regression

- Find a function, f(x), with at most ε –deviation from the target y

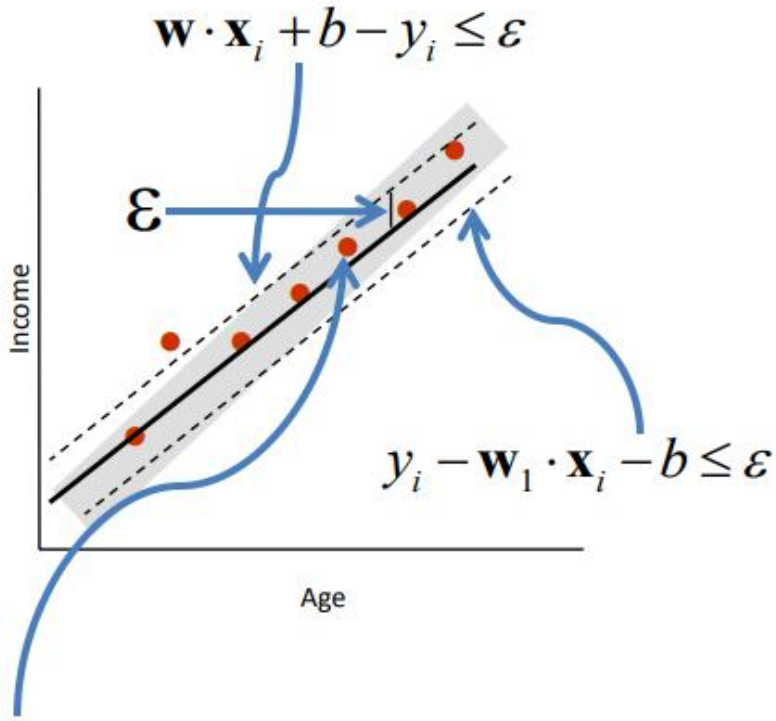
The problem can be written as a convex optimization problem

$$\min \frac{1}{2} \| \mathbf{w} \|^2$$

$$s.t.\ y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon;$$

$$\mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \le \varepsilon;$$

C: trade off the complexity



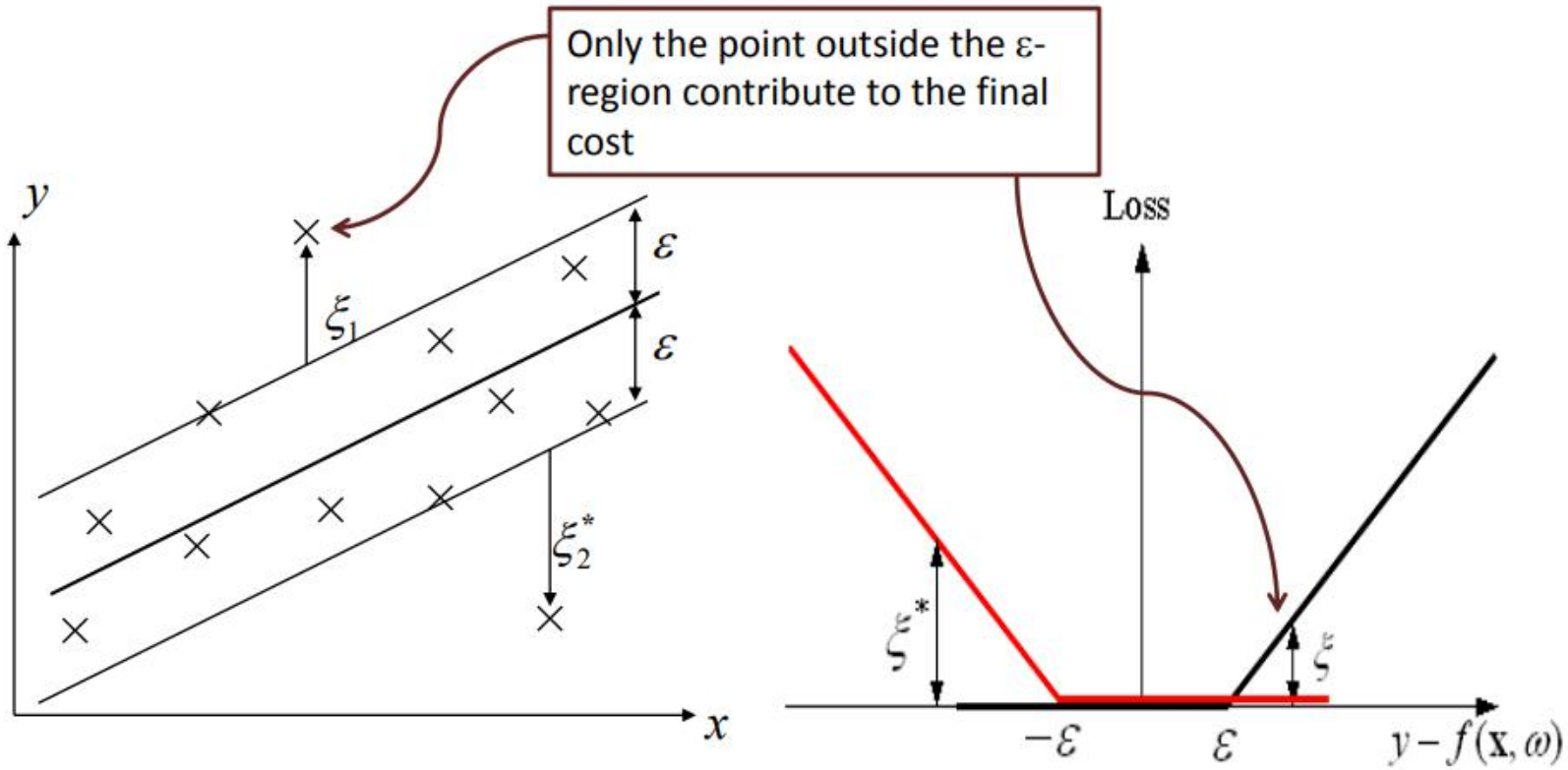
What if the problem is not feasible?

We can introduce slack variables (similar to soft margin loss function).

We do not care about errors as long as they are less than ε

Support Vector Regression

Assume linear parameterization $f(\mathbf{x}, \omega) = \mathbf{w} \cdot \mathbf{x} + b$

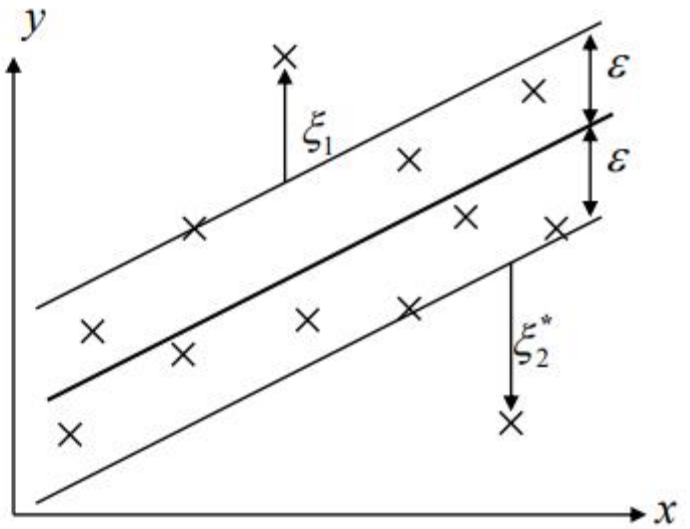


$$L_\varepsilon(y, f(\mathbf{x}, \omega)) = \max(|\, y - f(\mathbf{x}, \omega)\,| - \varepsilon, 0)$$

Soft margin

Given training data

$$(\mathbf{x}_i, y_i) \quad i = 1, \ldots, m$$



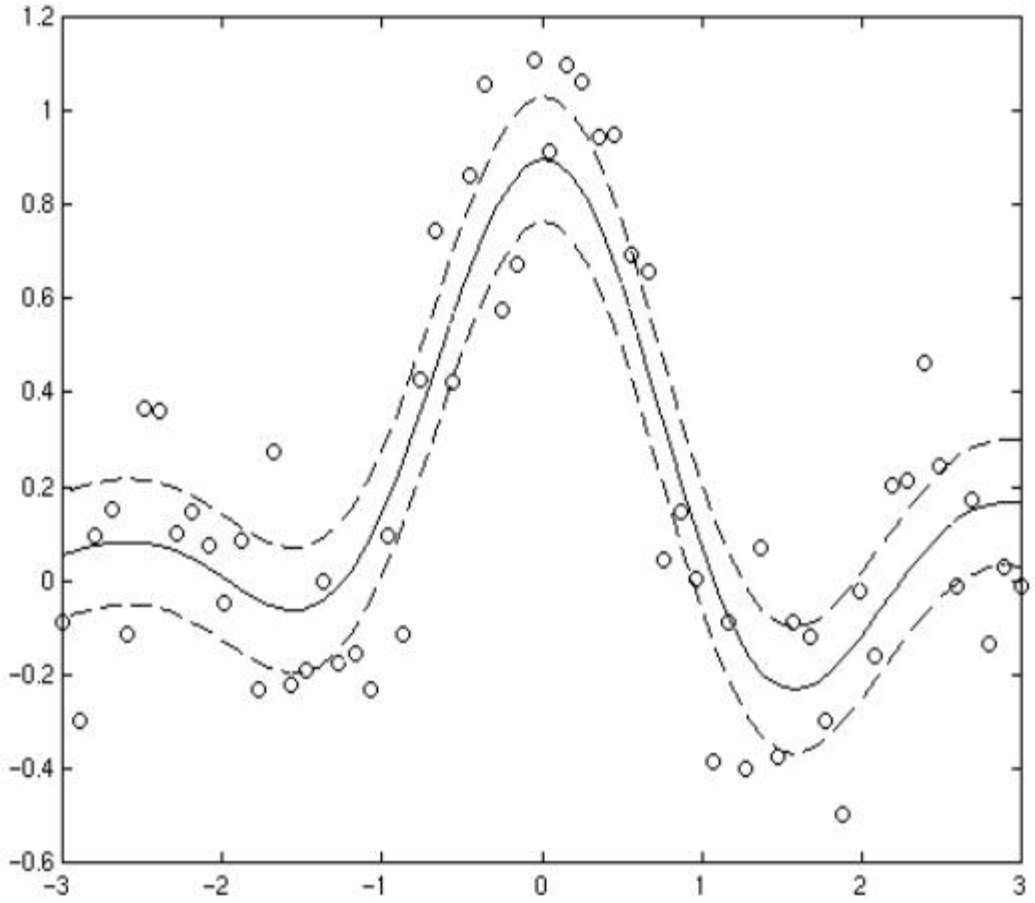
Minimize

$$\frac{1}{2}\| \mathbf{w} \|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*)$$

Under constraints

$$\begin{cases} y_i - (\mathbf{w} \cdot \mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w} \cdot \mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, i = 1, \ldots, m \end{cases}$$

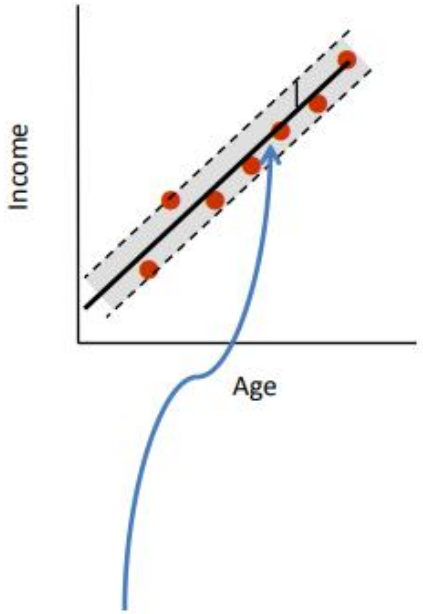
How about a non-linear case?



Linear versus Non-linear SVR

- **Linear case**

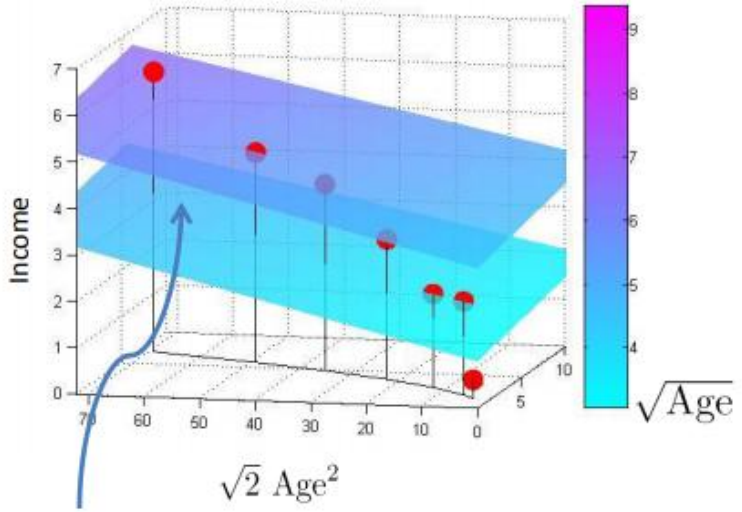
$$f : age \rightarrow income$$



$$y_i = \mathbf{w}_1 \cdot \mathbf{x}_i + b$$

- **Non-linear case**
  - Map data into a higher dimensional space, e.g.,

$$f : (\sqrt{age}, \sqrt{2}age^2) \rightarrow income$$



$$y_i = \mathbf{w}_1 \sqrt{\mathbf{x}_i} + \mathbf{w}_2 \sqrt{2}\mathbf{x}_i^2 + b$$

Dual problem

- Primal

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

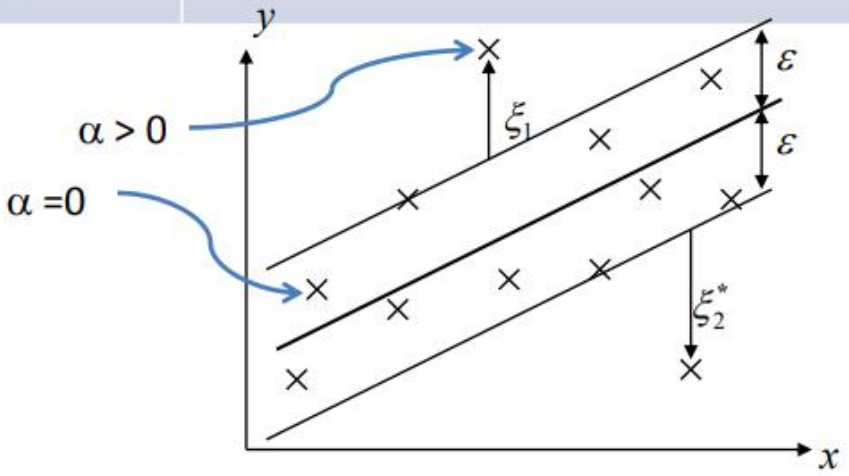
$$s.t.\begin{cases} y_i - (\mathbf{w}\cdot\mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w}\cdot\mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, i = 1,\ldots,m \end{cases}$$

- Dual

$$\max\begin{cases} \frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$s.t.\sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 0;\ 0 \le \alpha_i, \alpha_i^* \le C$$

Primal variables: w for each feature dim	Dual variables: α, α* for each data point
Complexity: the dim of the input space	Complexity: Number of support vectors



Kernel trick

- Linear: $\langle x, y \rangle$
- Non-linear: $\langle \varphi(x), \varphi(y) \rangle = K(x, y)$

Note: No need to compute the mapping function, φ(.), explicitly. Instead, we use the kernel function.

Commonly used kernels:

- Polynomial kernels: $K(x, y) = (x^T y + 1)^d$
- Radial basis function (RBF) kernels:

$$K(x, y) = \exp\left(-\frac{1}{2\sigma^2} \| x - y \|^2\right)$$

Note: for RBF kernel, dim(φ(.)) is infinite

Dual problem for non-linear case

- Primal

$$\min \frac{1}{2}\| \mathbf{w} \|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*)$$

$$s.t. \begin{cases} y_i - (\mathbf{w} \cdot \varphi(\mathbf{x}_i)) - b \le \varepsilon + \xi_i \\ (\mathbf{w} \cdot \varphi(\mathbf{x}_i)) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i \, , \xi_i^* \ge 0, i = 1, \ldots, m \end{cases}$$

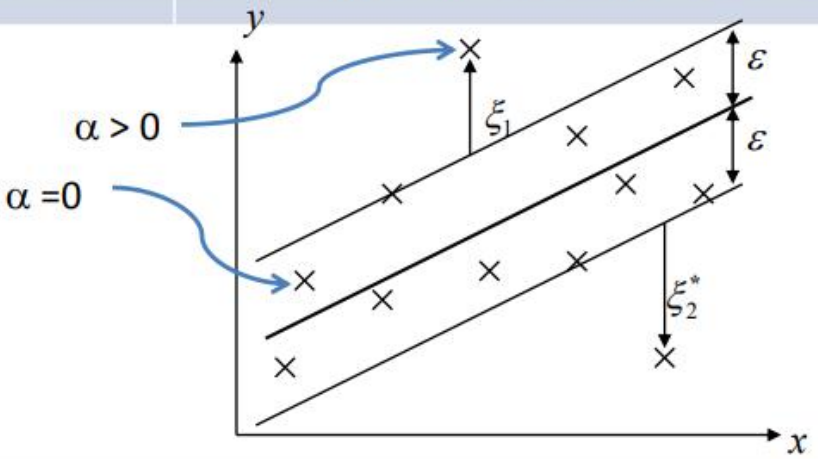
- Dual

K(xi, xj)

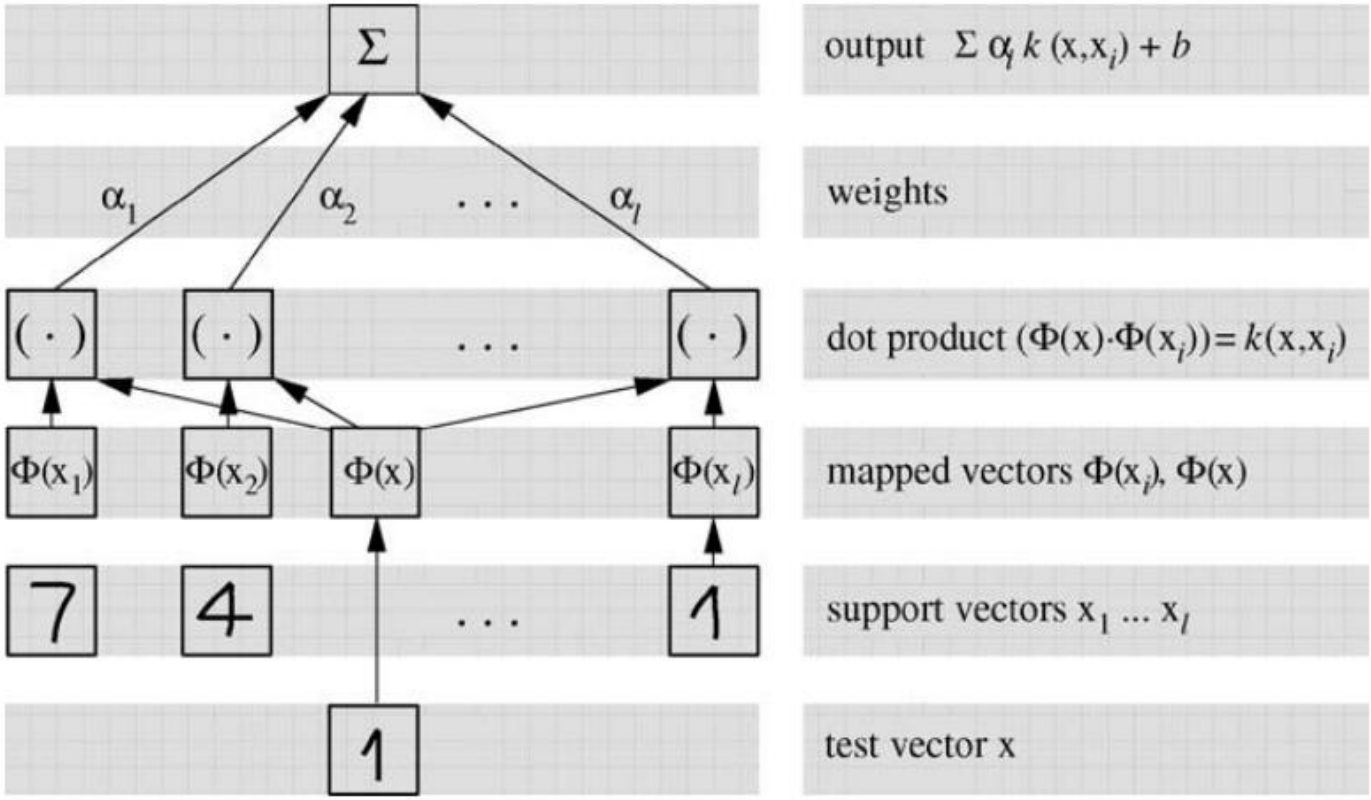
$$\max \begin{cases} \frac{1}{2} \sum_{i,j=1}^{m} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle \\ -\varepsilon \sum_{i=1}^{m} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y_i (\alpha_i - \alpha_i^*) \end{cases}$$

$$s.t. \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) = 0; \; 0 \le \alpha_i, \alpha_i^* \le C$$

Primal variables: w for each feature dim	Dual variables: α, α* for each data point
Complexity: the dim of the input space	Complexity: Number of support vectors



Architecture of a regression machine



[Alex J. Smola et al. A tutorial on support vector regression, 2004.]
URL: https://alex.smola.org/papers/2004/SmoSch04.pdf

Outline

- Regression Overview
- Linear Regression
- Support Vector Regression
- **Logistic Regression**
- Deep Neural Network for Regression

Logistic Regression

- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

– Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$

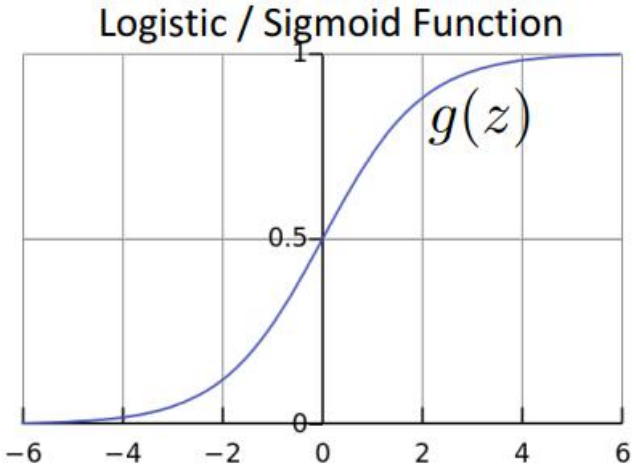
Can't just use linear regression with a threshold

- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

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

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



Interpretation of Hypothesis Output

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Example: Cancer diagnosis from tumor size

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.7$$

→ Tell patient that 70% chance of tumor being malignant

Note that: $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) + p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$

Therefore, $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1 - p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Another Interpretation

- Equivalently, logistic regression assumes that

$$\log \frac{p(y=1 \mid \boldsymbol{x}; \boldsymbol{\theta})}{p(y=0 \mid \boldsymbol{x}; \boldsymbol{\theta})} = \theta_0 + \theta_1 x_1 + \ldots + \theta_d x_d$$

odds of y = 1

Side Note: the odds in favor of an event is the quantity $p / (1 - p)$, where p is the probability of the event

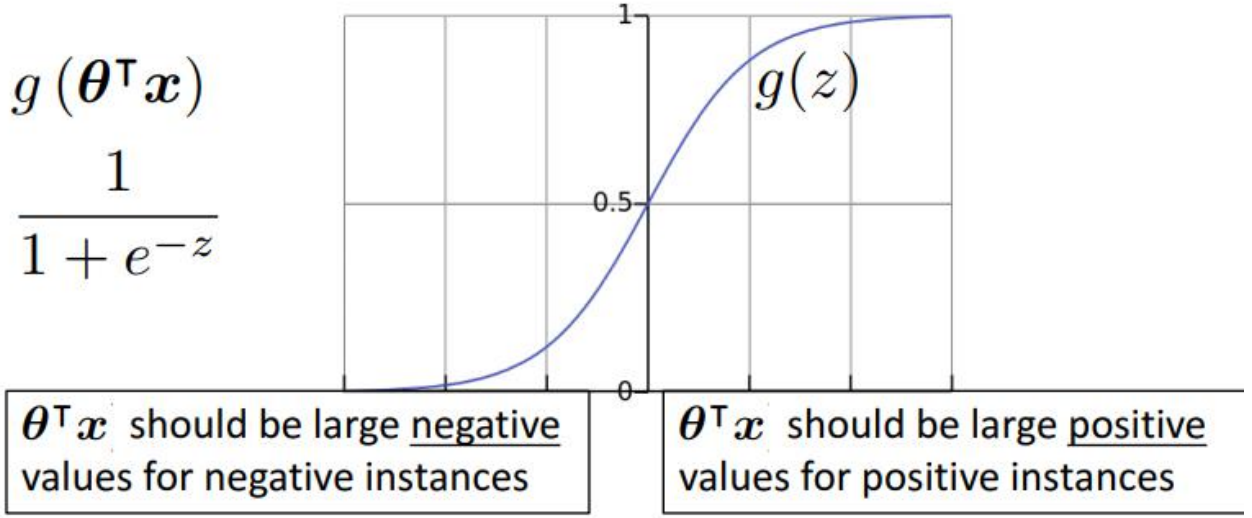
E.g., If I toss a fair dice, what are the odds that I will have a 6?

- In other words, logistic regression assumes that the log odds is a linear function of **x**

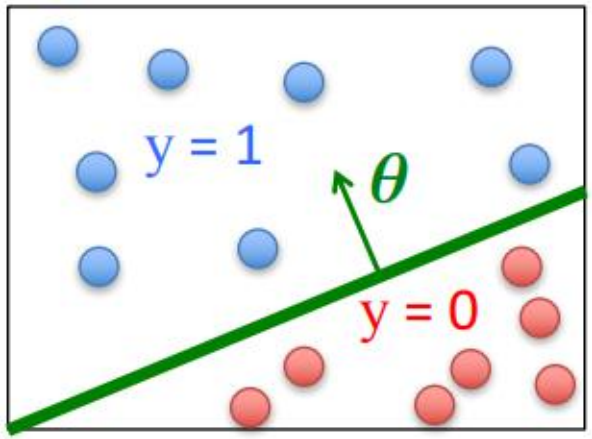
Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

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



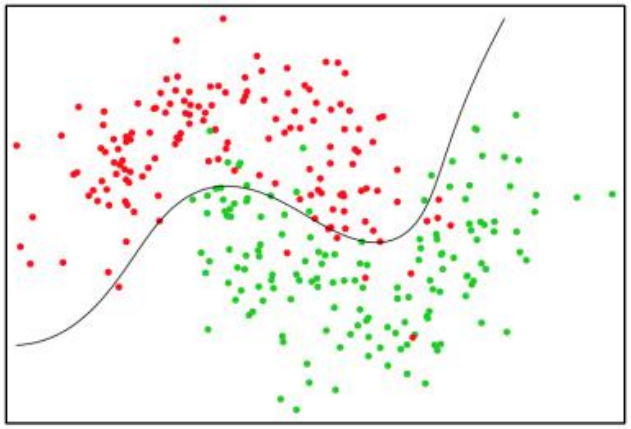
- Assume a threshold and...
  - Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Non-Linear Decision Boundary

- Can apply basis function expansion to features, same as with linear regression?

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ \vdots \end{bmatrix}$$



Logistic Regression

Given $\left\{ \left(\boldsymbol{x}^{(1)}, y^{(1)} \right), \left(\boldsymbol{x}^{(2)}, y^{(2)} \right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)} \right) \right\}$

where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d, \; y^{(i)} \in \{0, 1\}$

Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{x}\right)$

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

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^{\mathsf{T}} = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

Logistic Regression Objective Function

- Can't just use squared loss as in linear regression

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

  – Using the logistic regression model

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

results in a non-convex optimization

Deriving the Cost Function via Maximum Likelihood Estimation

- Likelihood of data is given by: $l(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$
- So, looking for the $\boldsymbol{\theta}$ that maximizes the likelihood

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

- Can take the log without changing the solution:

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Deriving the Cost Function via Maximum Likelihood Estimation

- Expand as follows:

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left[y^{(i)} \log p(y^{(i)} = 1 \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) + \left(1 - y^{(i)}\right) \log \left(1 - p(y^{(i)} = 1 \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right) \right]$$

- Substitute in model, and take negative to yield

Logistic regression objective:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right) \right]$$

Intuition Behind the Objective

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

- Cost of a single instance:

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

- Can re-write objective function as

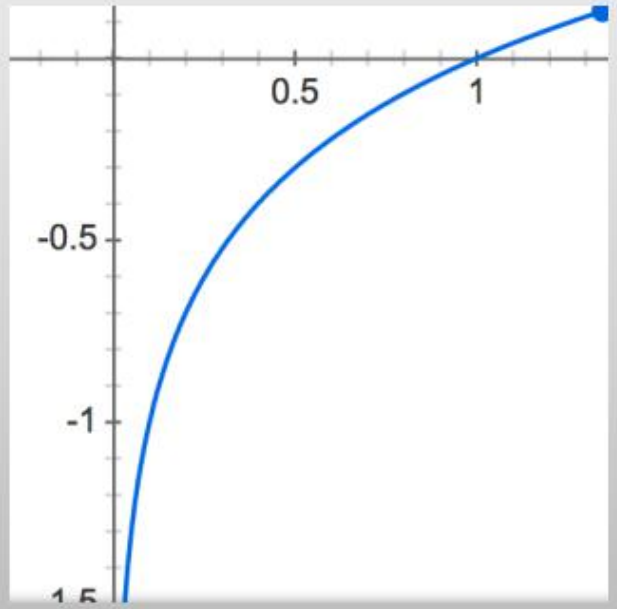
$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n}\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}\right)$$

Compare to linear regression: $J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$

Intuition Behind the Objective

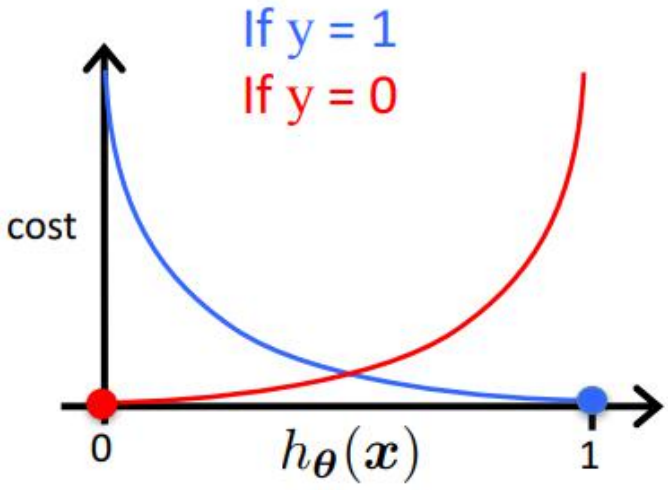
$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Aside: Recall the plot of log(z)



Intuition Behind the Objective

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



If y = 0

- Cost = 0 if prediction is correct
- As $(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \to 0, \text{cost} \to \infty$
- Captures intuition that larger mistakes should get larger penalties

Regularized Logistic Regression

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right]$$

- We can regularize logistic regression exactly as before

$$\begin{aligned} J_{\text{regularized}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda\sum_{j=1}^{d}\theta_j^2 \\ &= J(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}_{[1:d]}\|_2^2 \end{aligned}$$

Gradient Descent for Logistic Regression

$$J_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right] + \lambda\|\boldsymbol{\theta}_{[1:d]}\|_2^2$$

Want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

Use the natural logarithm (ln = $\log_e$) to cancel with the exp() in $h_{\boldsymbol{\theta}}(\boldsymbol{x})$

Gradient Descent for Logistic Regression

$$J_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right] + \lambda\|\boldsymbol{\theta}_{[1:d]}\|_2^2$$

Want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence (simultaneous update for j = 0 ... d)

$$\theta_0 \leftarrow \theta_0 - \alpha\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)$$

$$\theta_j \leftarrow \theta_j - \alpha\left[\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} - \frac{\lambda}{n}\theta_j\right]$$

Gradient Descent for Logistic Regression

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence (simultaneous update for j = 0 ... d)

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)$$

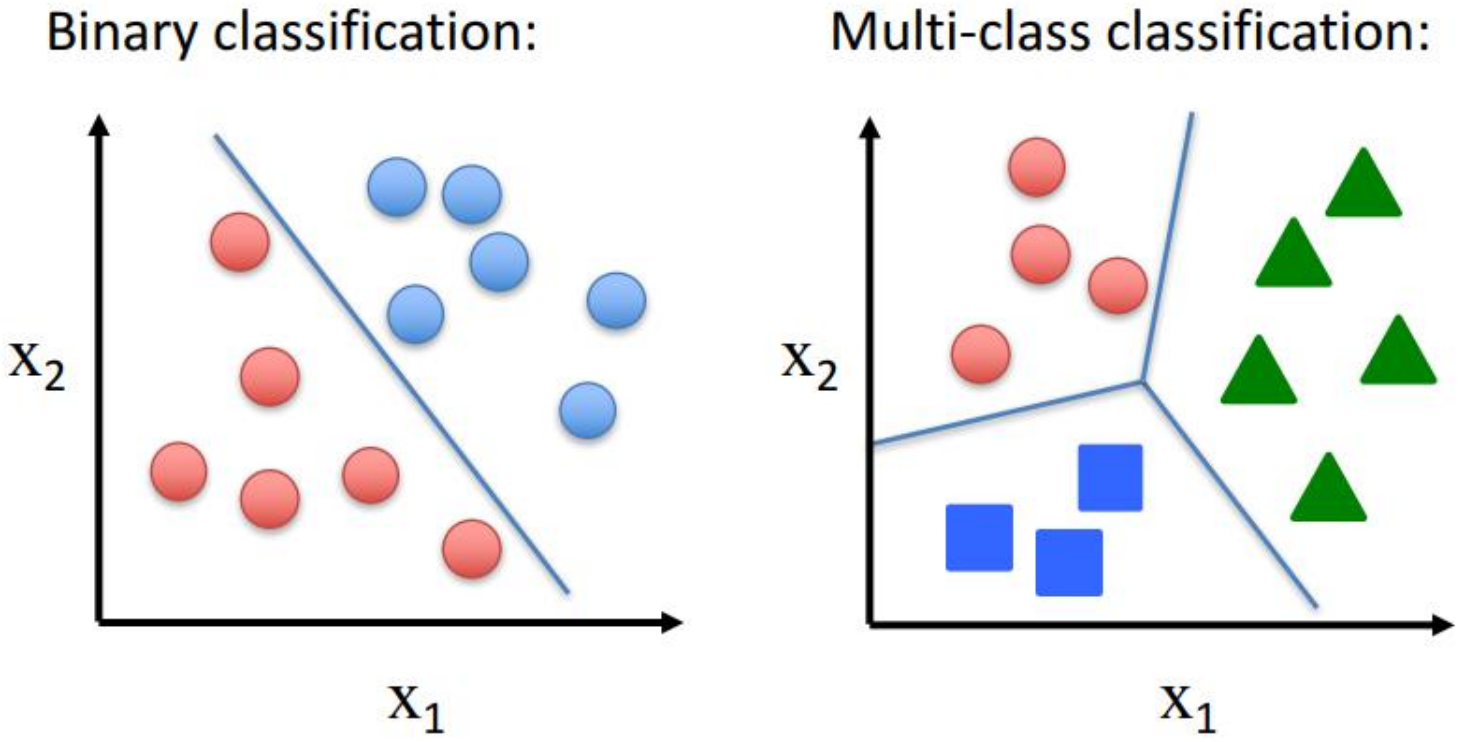
$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} - \frac{\lambda}{n} \theta_j \right]$$

This looks IDENTICAL to linear regression!!!

- Ignoring the 1/n constant
- However, the form of the model is very different:

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

Multi-Class Classification



Disease diagnosis: healthy / cold / flu / pneumonia

Object classification: desk / chair / monitor / bookcase

Multi-Class Logistic Regression

- For 2 classes:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})} = \frac{\exp(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})}{1+\exp(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})}$$

weight assigned to y = 0

weight assigned to y = 1

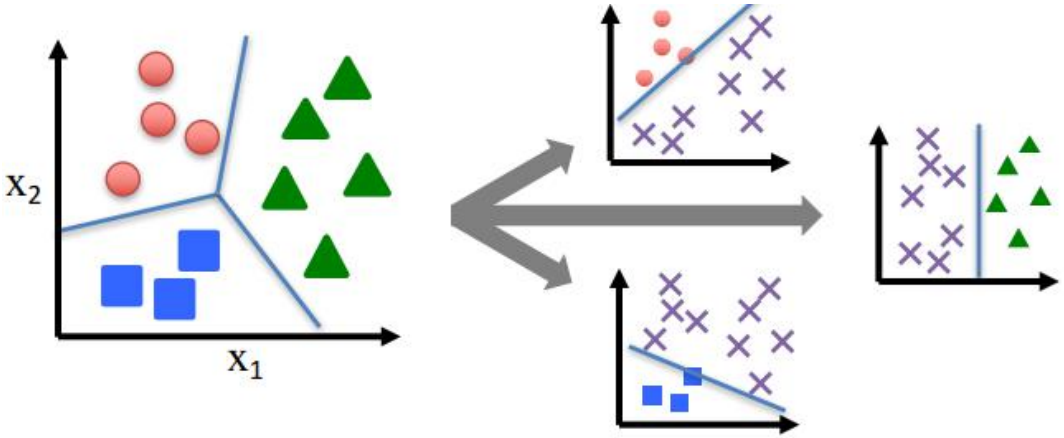
- For C classes {1, ..., C}:

$$p(y=c \mid \boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C) = \frac{\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}$$

  - Called the **softmax** function

Multi-Class Logistic Regression

- Split into One vs Rest:



- Train a logistic regression classifier for each class i to predict the probability that y = i with

$$h_c(\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}$$

Implementing Multi-Class Logistic Regression

- Use $h_c(\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}$ as the model for class c

- Gradient descent simultaneously updates all parameters for all models
  - Same derivative as before, just with the above $\mathrm{h_c(x)}$

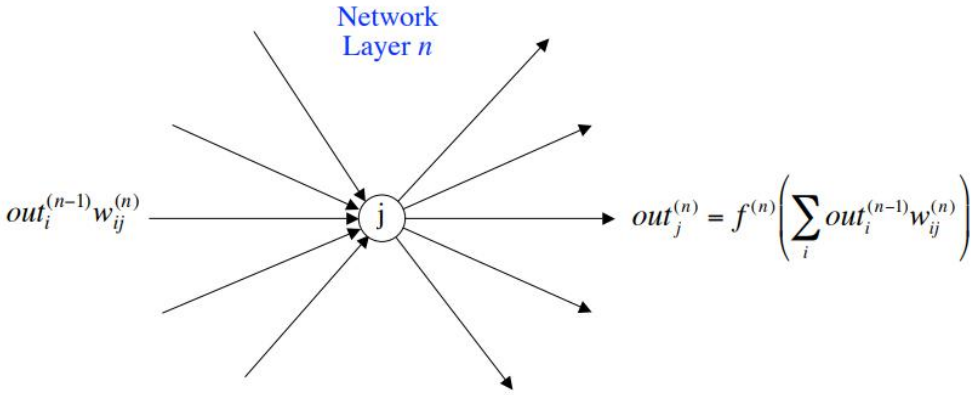
- Predict class label as the most probable label

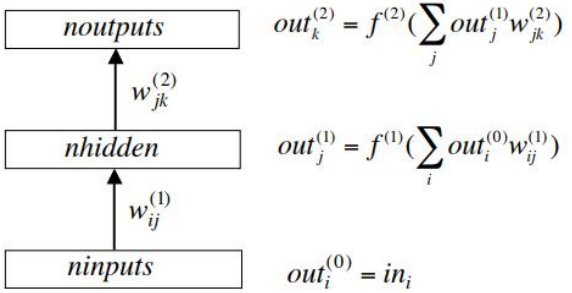
$$\max_c h_c(\boldsymbol{x})$$

Outline

- Regression Overview
- Linear Regression
- Support Vector Regression
- Logistic Regression
- **Deep Neural Network for Regression**

DNN Regression





- For a two-layer MLP:

$$out_k^{(2)} = f^{(2)}\left(\sum_j out_j^{(1)}.w_{jk}^{(2)}\right) = f^{(2)}\left(\sum_j f^{(1)}\left(\sum_i in_i w_{ij}^{(1)}\right).w_{jk}^{(2)}\right)$$

- The network weights are adjusted to minimize an output cost function

$$E_{SSE} = \frac{1}{2}\sum_p \sum_j \left(targ_j^p - out_j^{(N)p}\right)^2$$

Computing the Partial Derivatives for Regression

- We use SSE and for a two layer network the linear final outputs can be written:

$$out_k^{(2)} = \sum_j out_j^{(1)}.w_{jk}^{(2)} = \sum_j f\left(\sum_i in_i w_{ij}^{(1)}\right).w_{jk}^{(2)}$$

- We can then use the chain rules for derivatives, as for the Single Layer Perceptron, to give the derivatives with respect to the two sets of weights

$$\frac{\partial E_{SSE}(\{w_{ij}^{(n)}\})}{\partial w_{hl}^{(m)}} = -\sum_p \sum_k \left(targ_k - out_k^{(2)}\right).\frac{\partial out_k^{(2)}}{\partial w_{hl}^{(m)}}$$

$$\frac{\partial out_k^{(2)}}{\partial w_{hl}^{(2)}} = \sum_j out_j^{(1)} \frac{\partial w_{jk}^{(2)}}{\partial w_{hl}^{(2)}} = \sum_j out_j^{(1)}.\delta_{jh}.\delta_{kl} = out_h^{(1)}.\delta_{kl}$$

$$\frac{\partial out_k^{(2)}}{\partial w_{hl}^{(1)}} = \sum_j f'\left(\sum_i in_i w_{ij}^{(1)}\right)\left(\sum_m in_m \frac{\partial w_{mj}^{(1)}}{\partial w_{hl}^{(1)}}\right).w_{jk}^{(2)} = f'\left(\sum_i in_i w_{il}^{(1)}\right).in_h.w_{lk}^{(2)}$$

Deriving the Back Propagation Algorithm for Regression

- All we now have to do is substitute our derivatives into the weight update equations

$$\Delta w_{hl}^{(2)} = \eta \sum_p \sum_k \left(targ_k - out_k^{(2)} \right) . out_h^{(1)} . \delta_{kl} = \eta \sum_p \left(targ_l - out_l^{(2)} \right) out_h^{(1)}$$

$$\Delta w_{hl}^{(1)} = \eta \sum_p \sum_k \left(targ_k - out_k^{(2)} \right) . f' \left(\sum_i in_i w_{il}^{(1)} \right) . w_{lk}^{(2)} . in_h$$

- Then if the transfer function f(x) is a Sigmoid we can use f′(x) = f(x).(1 – f(x)) to give

$$\Delta w_{hl}^{(2)} = \eta \sum_p \left(targ_l - out_l^{(2)} \right) out_h^{(1)}$$

$$\Delta w_{hl}^{(1)} = \eta \sum_p \sum_k \left(targ_k - out_k^{(2)} \right) w_{lk}^{(2)} . out_l^{(1)} . \left(1 - out_l^{(1)} \right) . in_h$$

- These equations constitute the Back-Propagation Learning Algorithm for Regression.

Classification + Localization: Task

Classification: C classes
Input: Image
Output: Class label
Evaluation metric: Accuracy

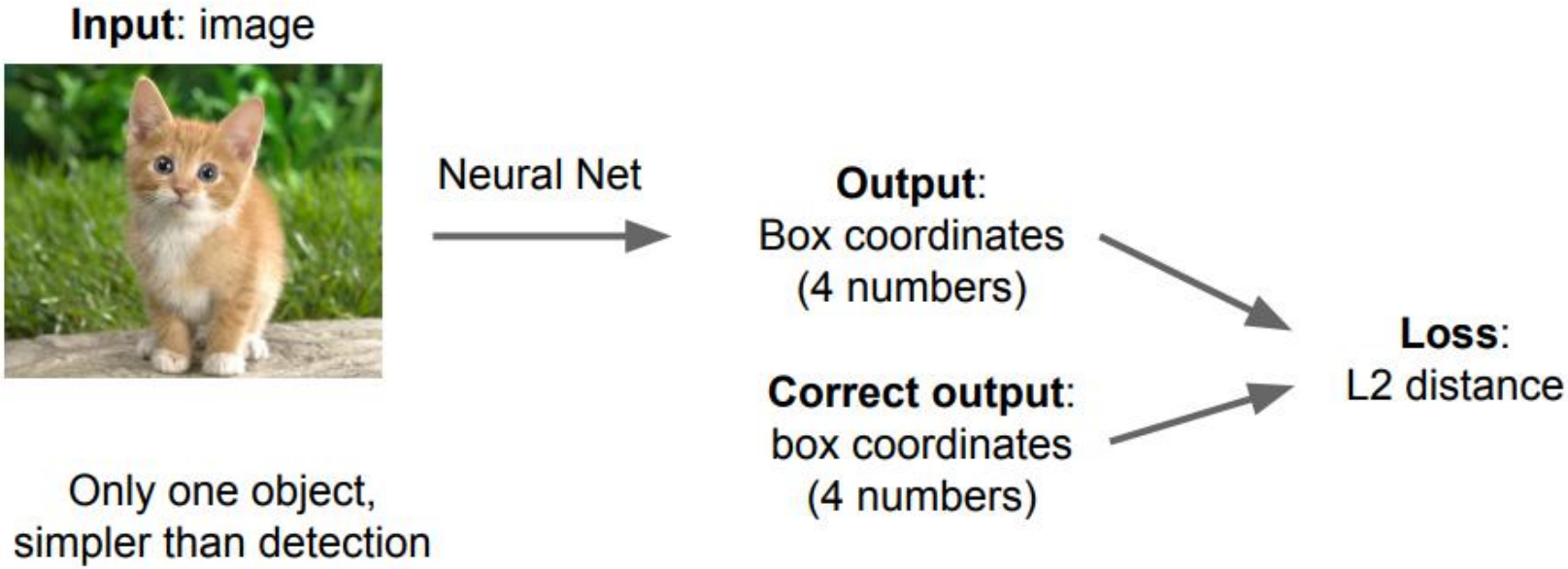
→ CAT

Localization:
Input: Image
Output: Box in the image (x, y, w, h)
Evaluation metric: Intersection over Union

→ (x, y, w, h)

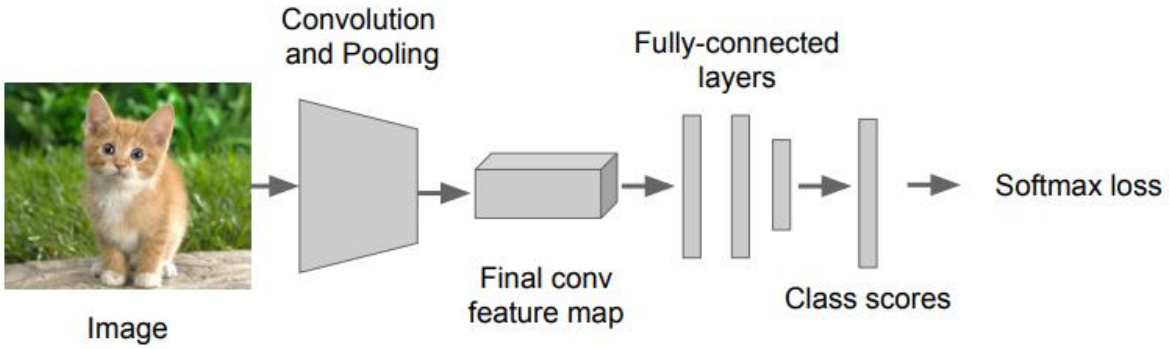
Classification + Localization: Do both

Idea #1: Localization as Regression



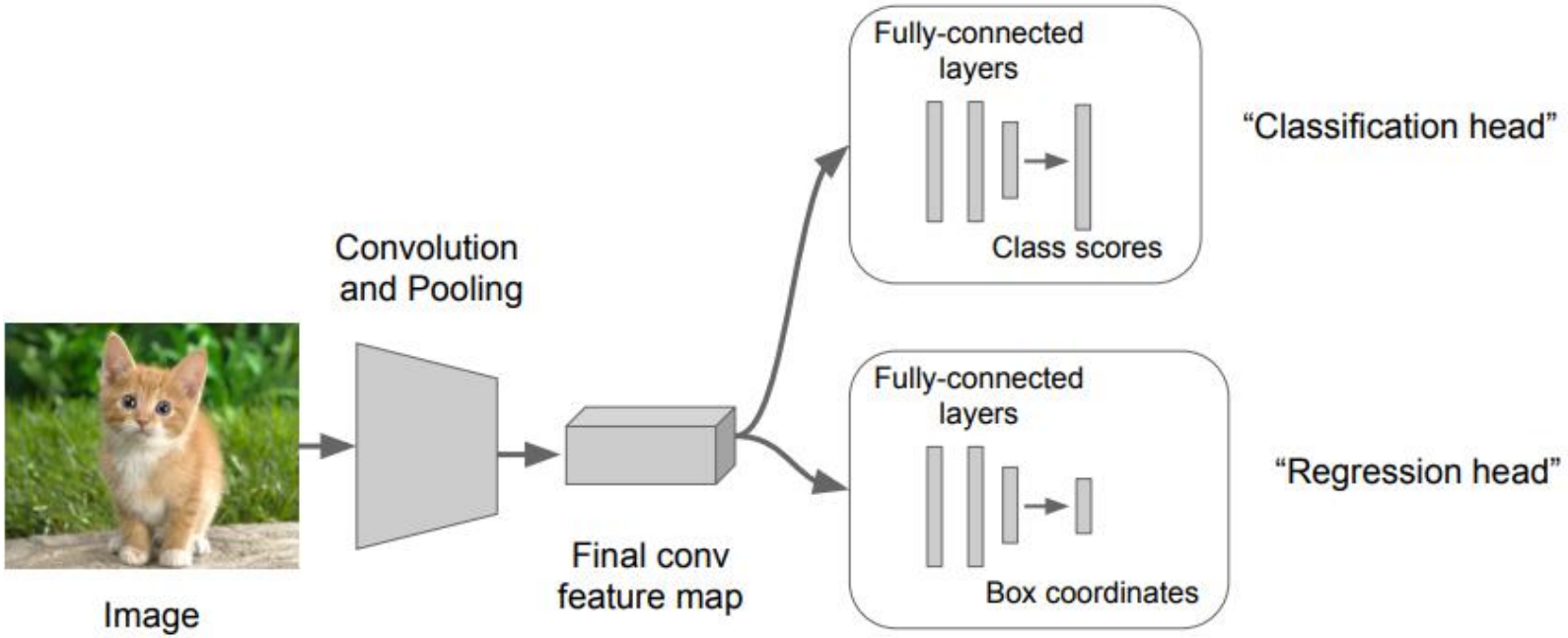
Simple Recipe for Classification + Localization

- Step 1: Train (or download) a classification model (AlexNet, VGG, GoogLeNet)



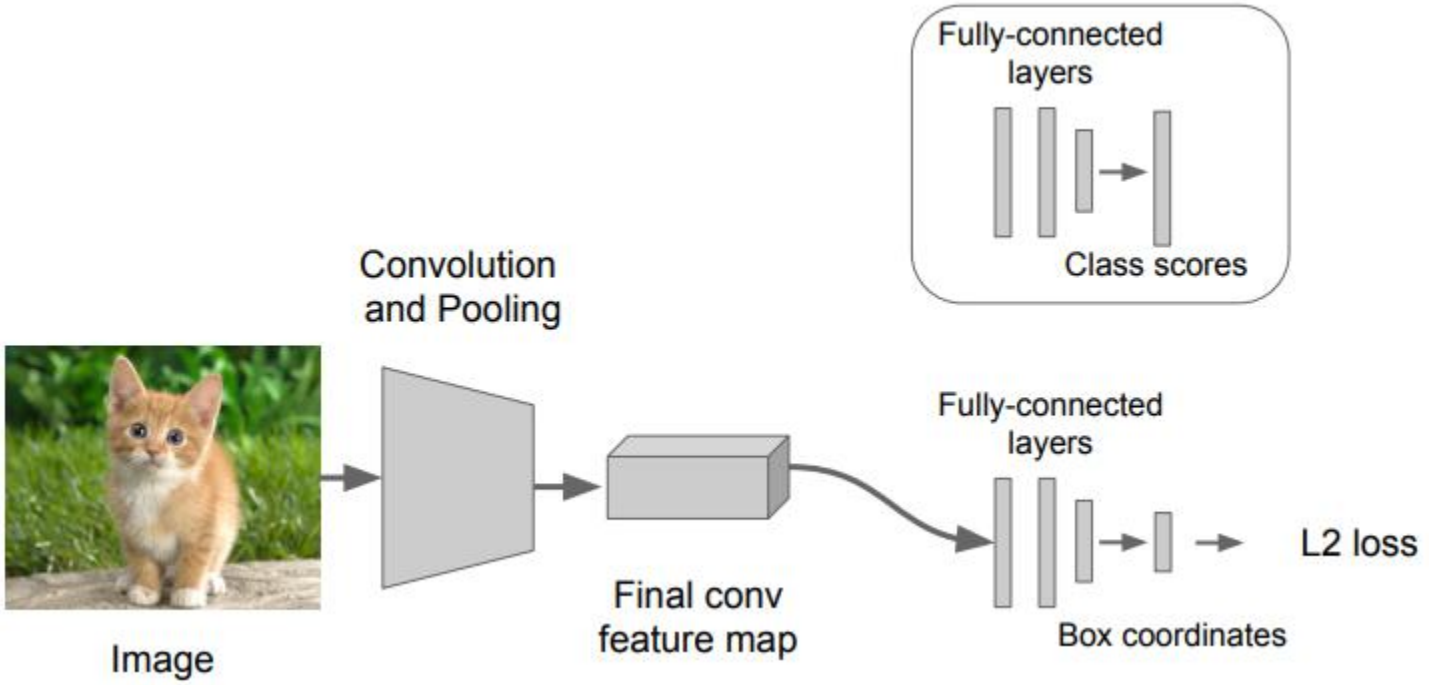
Simple Recipe for Classification + Localization

- Step 2: Attach new fully-connected "regression head" to the network



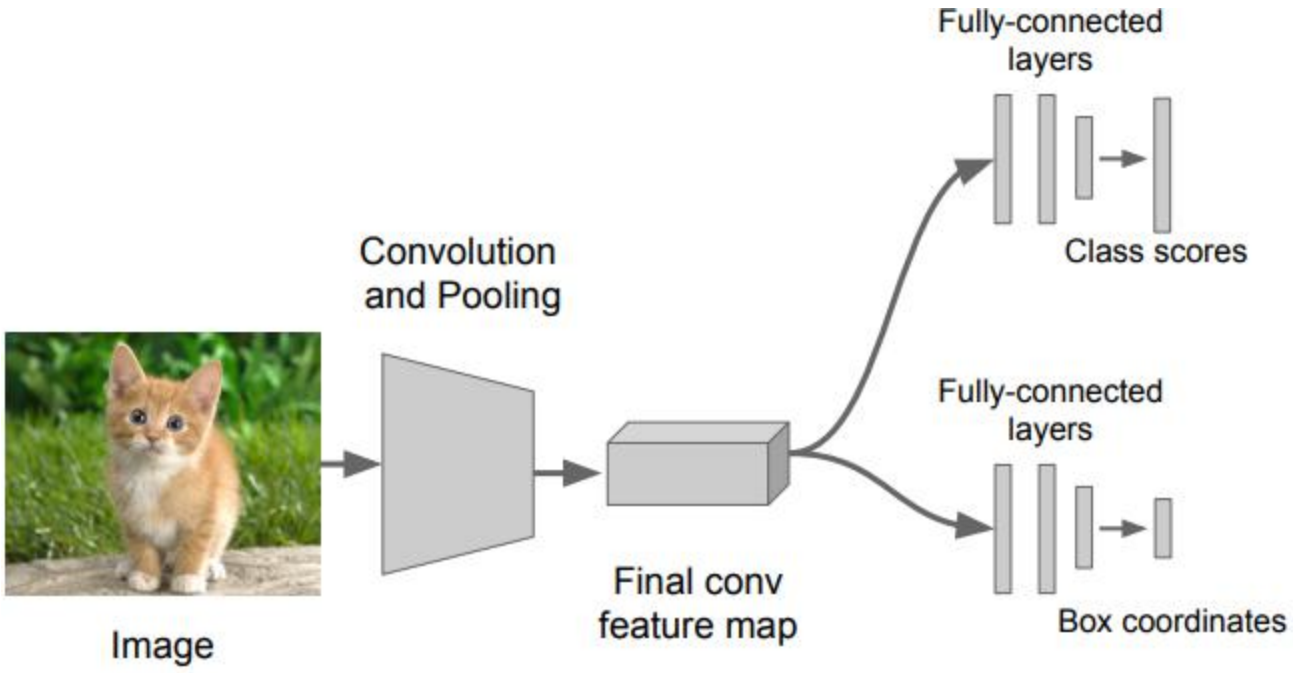
Simple Recipe for Classification + Localization

- Step 3: Train the regression head only with SGD and L2 loss

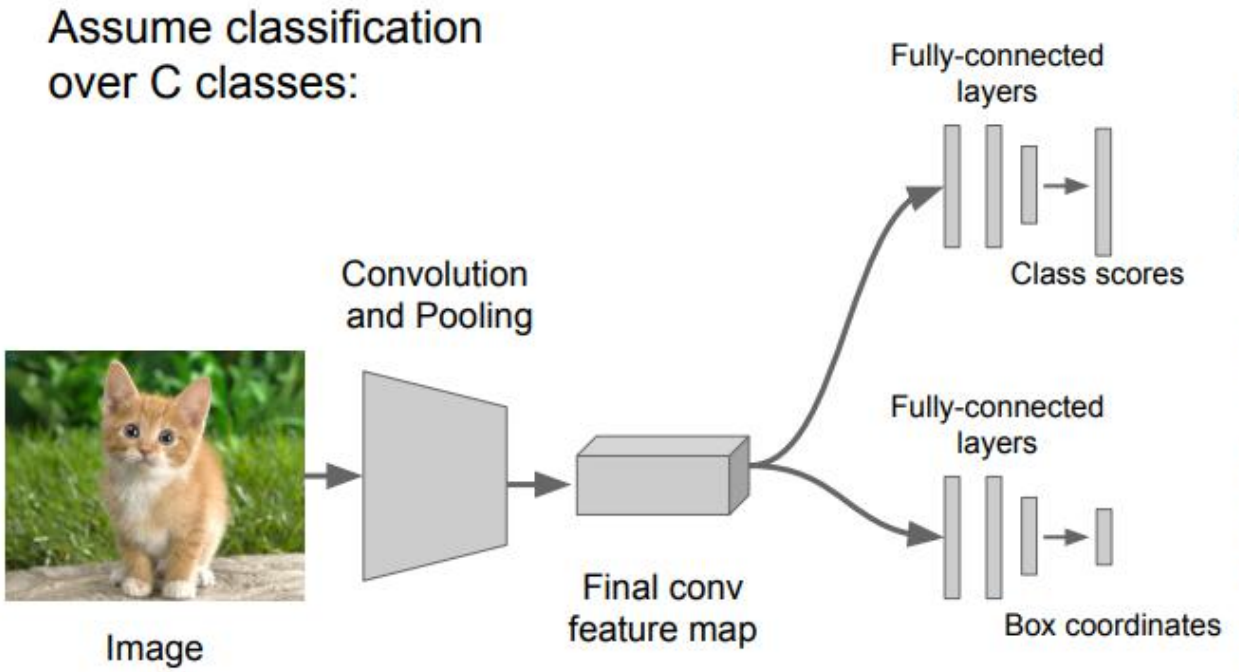


Simple Recipe for Classification + Localization

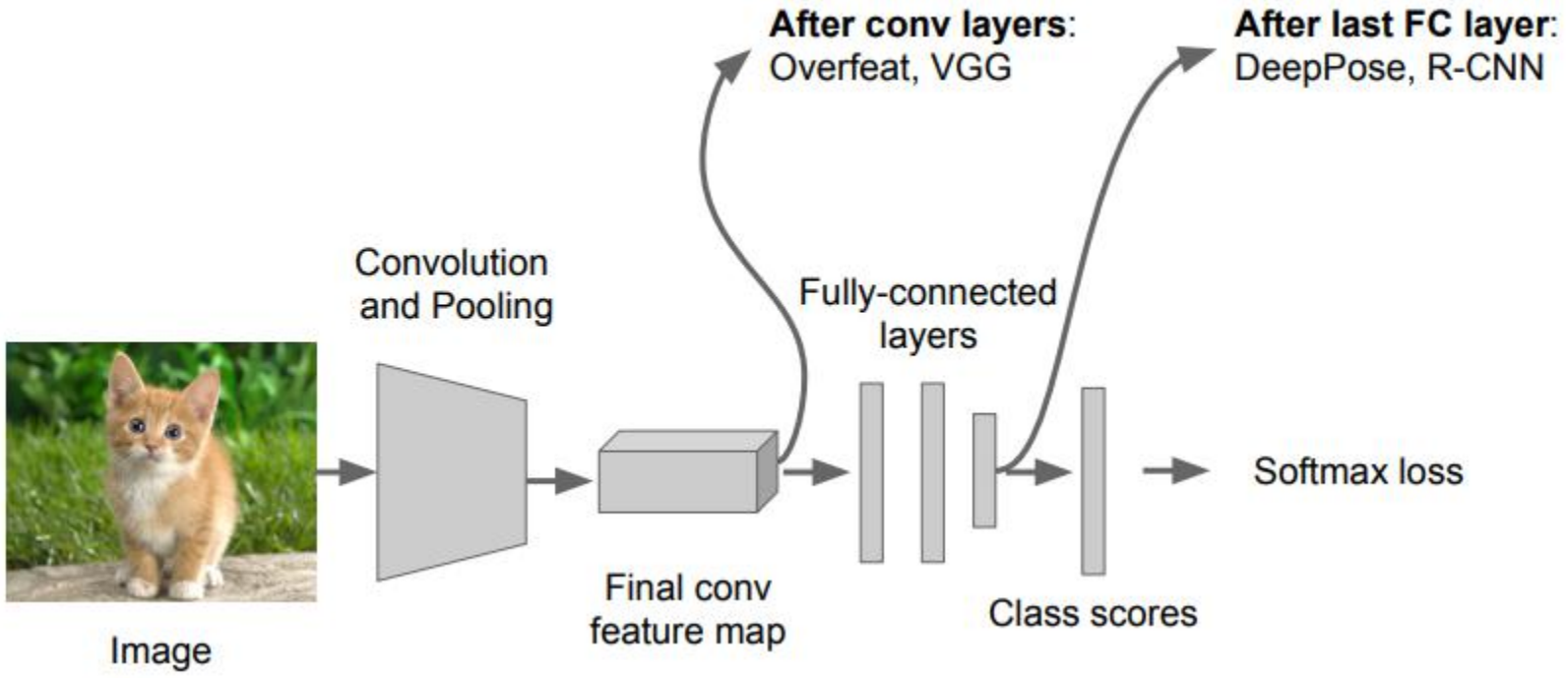
- Step 4: At test time use both heads



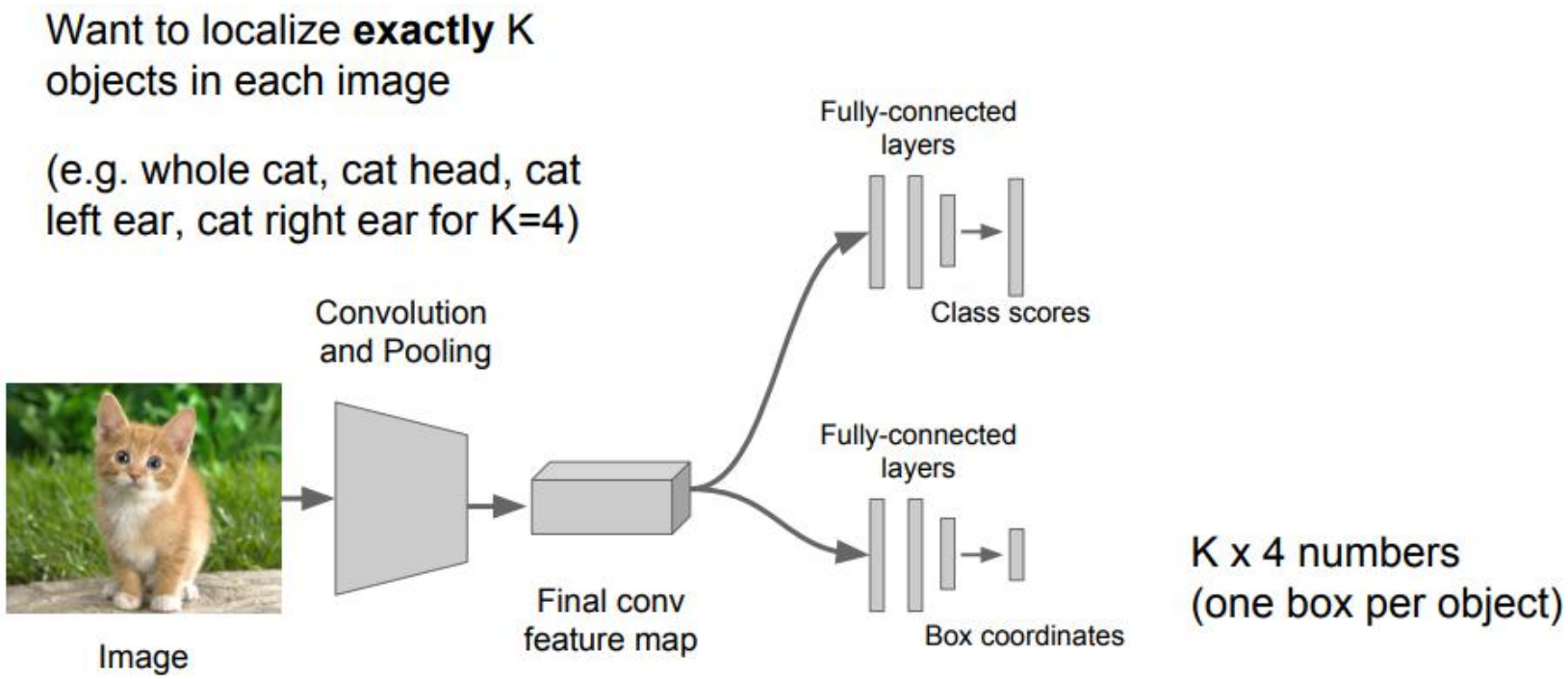
Per-class vs class agnostic regression



Where to attach the regression head?



Aside: Localizing multiple objects

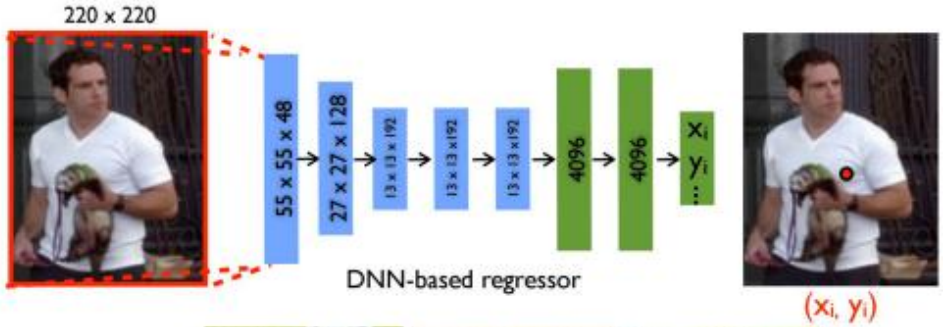


Aside: Human Pose Estimation

Represent a person by K joints

Regress (x, y) for each joint from last fully-connected layer of AlexNet

(Details: Normalized coordinates, iterative refinement)



Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

DNN Regression Applications

- Great results in:
  - Computer Vision
    - Object Localization / Detection as DNN Regression
    - Self-driving Steering Command Prediction
    - Human Pose Regression
  - Finance
    - Currency Exchange Rate
    - Stock Price Prediction
    - Forecasting Financial Time Series
    - Crude Oil Price Prediction

DNN Regression Applications

- Great results in:
  - Atmospheric Sciences
    - Air Quality Prediction
    - Carbon Dioxide Pollution Prediction
    - Ozone Concentration Modeling
    - Sulphur Dioxide Concentration Prediction
  - Infrastructure
    - Road Tunnel Cost Estimation
    - Highway Engineering Cost Estimation
  - Geology / Physics
    - Meteorology and Oceanography Application
    - Pacific Sea Surface Temperature Prediction
    - Hydrological Modeling

Q & A