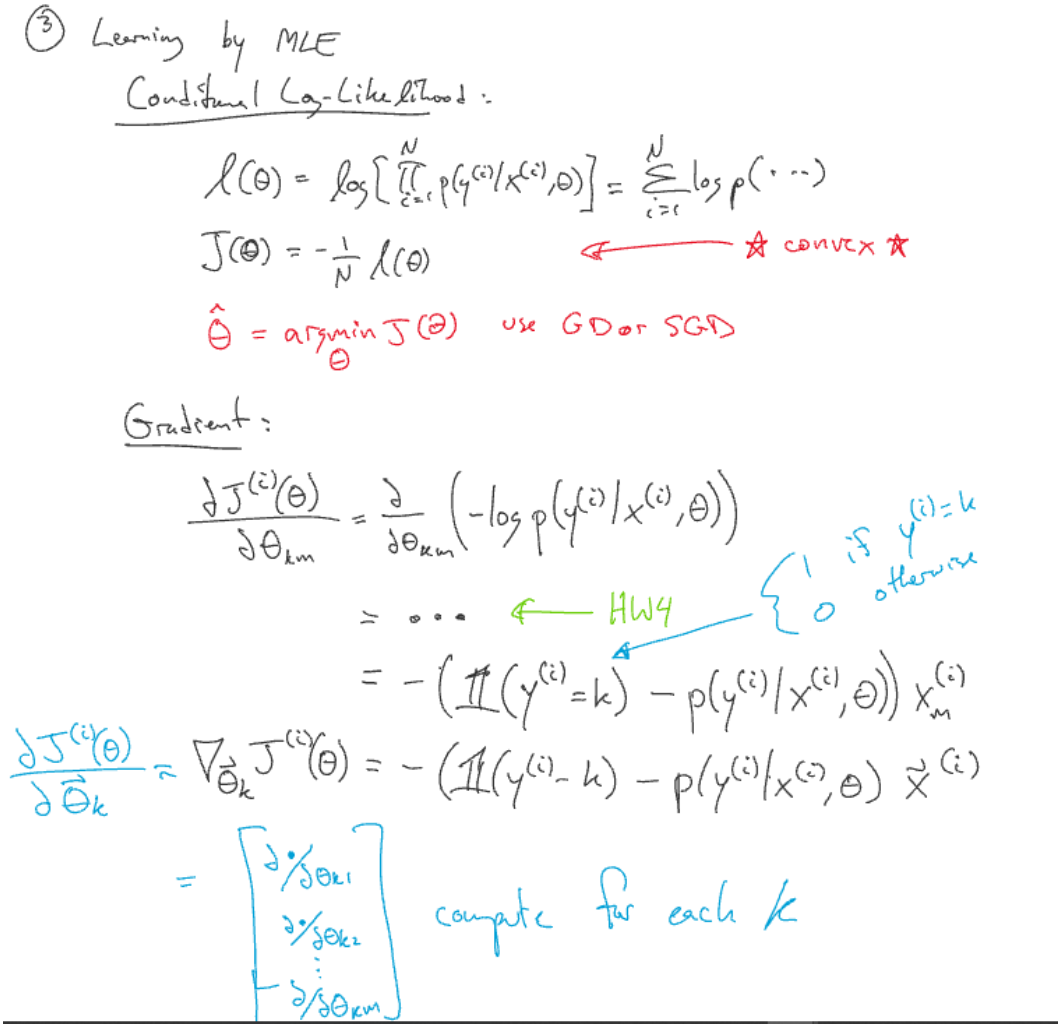
بسم الله الرحمن الرحيم

احنا كنا بنحاول ن classifiy 3 dimensional points of different animals .. الحيوانات كانو polar bear, sea lion and whales .. و في الآخر قلنا اننا عندنا موديل اللي هو ال multinomial logistic regression و هنا قلنا انو متعرف علي اساس انو probability distribution .. في حين ان ال probabilities جات من انك تبص علي ال signed distance of a point بالنسبه لواحد من ال 3 hyperplanes عندي .. و ال signed distance بمعني انها لو علي الناحية البوزيتيف يبقا هي مسافه فعلاً .. انما لو كانت علي النيجاتيف ساعد .. فانت بتضربها فسالب واحد ههه . عشان الديستانس دايماً بوزيتيف .. وبعدين قلنا نرفع الكلام ده ل exp عشان نجيب رقم موجب ما بين الصفر و المالا نهايه .. بعدها هت normalize الكلام ده .. قلنا بعد كدا ان زي ما كان بيحصل في ال binary logistic regression ... انت تقدر تفكر في الموديل علي اساس انو coin flipping distribution ... والواي جايه من Bernoulli distribution والبارمتر فاي بتاعه هو عباره عن فانكشن في البارمترز ثيتا و الفيتشرز إكس .. هنا بنقول انك تقدر تعمل weighted die roll انت معندكش غير die واحده بس .. و الواحده ديه ليها k sides .. وكل ناحيه ليها الاحتمالية بتاعتها والاحتماليه ديه بتبقا deterministic function of theta and X .... والواي في الحاله ديه هي بتبقا شبه ال weighted die roll in a multinomial case .. احنا هنكمل علي المره اللي فاتت .. دلوقت عاوزين نفكّر ال learning هنا معناه ايه وبيحصل ازاي .. فعشان تتعلم مع ال maximum likelihood estimation اللي بنحتاج نعملو اننا نكتب ال objective function و ديه بتبقا عباره عن conditional log-likelihood و هنا هنعرّفها علي اساس انها فانكشن L(theta) و هي اللوج بتاعت ال log-likelihood و ديه بتبقا ال multiplication from 1 to N ل ال p(y|x, theta) .. لما بندخّلا ال Log في النص بناخد summation مش multiplication .. ودلوقت هنعرّف ففانكشن اسمها J(theta) ... و هي ال -1/N L(theta) ... و الفانكشن اللي اسمها J(theta) ديه بتبقا convex فانت تقدر تستخدم ال GD and SGD عشان ت minimize J(theta) .. فعشان تستخدم واحده من الاتنين اللي هم ال GD or SGD انت هتحتاج تحسب ال gradient فهتحتاج تجيب ال partial derivatives .. فهتقول ان اللي انت عاوزو هنا هو ال partial derivative of Ji(theta) .. بص ع الصوره



ال || ديه اسمها binary indicator .. ديه كل الي بتعملو انها تقول 1 لو ال yi = k .. غير كدا هتبقا بصفر .. المهم لم كل دول وحطهم في ال gradient فال gradient هنا هيبقا زي الصوره اللي فوق كدا بالظبط ... خد بالك بس من حاجه .. الجريدينت اللي بتحسبو ده اسمو vector derivative .. وهنتكلم عنو في ال NNs .. المهم انت مش لازم تعرف ازاي تتعامل معاهم يعني .. بس ديه مشتقه بالنسبه لفيكتور كامل ... فانت لما بتعمل كدا بيطلعلك الجريدينت .. وده الكولوم فيكتور اللي انت شايفو فوق في الصوره ديه ... النقطه اللي الدكتور حاططها ديه اللي هي ال Ji(theta) بس الدكتور مكسّل يكتب كل شويه

يعني ... خد بالك ان الفيكتور ده مش ال full gradient اللي انت مهتم بيها .. اللي احنا فعلاً مهتمين بيه هو ال full gradient for the entire matrix of parameters .. انت هتحتاج تحسب الفيكتور اللي انت شايفو بالازرق فوق ده لكل k class عندك .. وده هيديلك ال k rows of the gradient of the matrix .. أول ما هتعمل الكلام ده .. تعال نستخدم ال SGD او GD عشان نتعلم ال optimization وبعدها نعمل ال prediction لأحسن كلاس في الموديل بتاعنا .. احنا شفنا الكلام ده في حاجات كتير زي مثلاً majority vote او binary logistic regression و دلوقت ل multinomial logistic regression .. و دلوقت عاوزين نتكلم علي حته practical كدا .. في غلطه شائعه .. الكود اللي في الصوره الجايه ديه فيه غلطه ..

Debug that Program!

In-Class Exercise: *Think-Pair-Share*

Debug the following program which is (incorrectly) attempting to run SGD for multinomial logistic regression

Buggy Program:

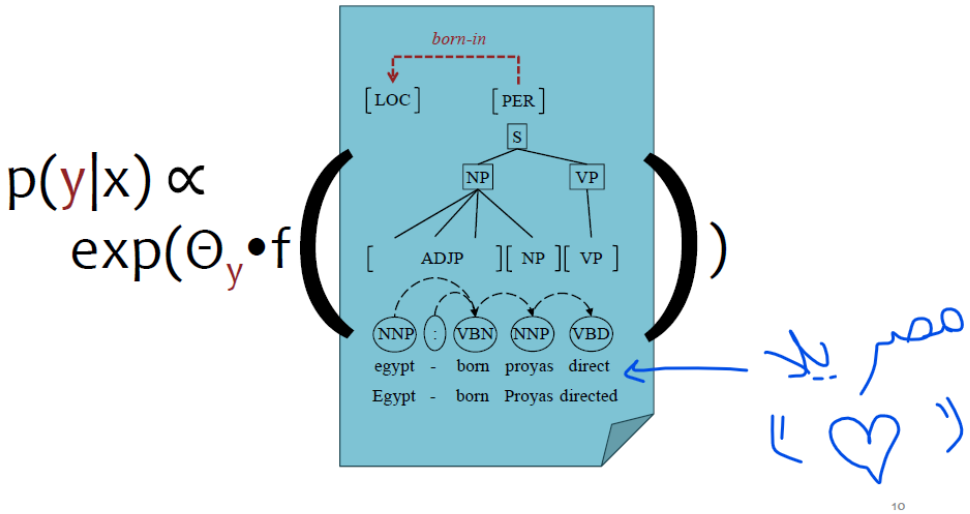
```
while not converged:
  for i in shuffle([1,…,N]):
    for k in [1,…,K]:
      theta[k] = theta[k] - gamma * grad(x[i], y[i], theta, k)
```

Assume: grad(x[i], y[i], theta, k) returns the gradient of the negative log-likelihood of the training example (x[i],y[i]) with respect to vector theta[k]. gamma is the learning rate. N = # of examples. K = # of output classes. M = # of features. theta is a K by M matrix.

افترض انك بتحاول ترن ال SGD وعندك 2 بارمترز theta 1 and theta 2 .. هل ال SGD or GD بتاخد خطوه في ال direction بتاع ال steepest descent ولا الاول بتاخد ال steepest descent لبارمتر و بعدين البارمتر التاني .. الدكتور قال ان المفروض اننا نحسب الجريدينت كلو بتاع الثيتا وبعدين نأبديت الثيتا بالجريدينت ده .. انت فعلاً ممكن تفكّر فيها كإنك عندك مشكله فيها ال k = 2 and M = 1 .. كدا الثيتا ماتركس فيها 2 entries .. انت هنا المفروض تحسب ال full gradient matrix اللي هي دلوقت بقت فيكتور ليه 2 entries .. وبعدين تاخد خطوه في الاتجاه ده .. اللي هو الخط الأحمر ... انما اللي انت بتعملو في الصوره الي فوق .. انك بتحسب ال gradient لواحد من الاتنين بارمترز دول .. اللي هو قيمه واحده من القيمتين بتوع ال K .. وبعدين تقوم واخد خطوه في الاتجاه ده و تعمل update للثيتا .. وبعدين ت call the gradient تاني .. للقيمه التانيه بتاعت ال k وتقوم واخد خطوه تانيه ... وده ممكن يودينا لمكان تاني .. فالامبلمنتيشن عشان يتصلح ... حط variable tmp في النص ما بين ال 2 loops .. الفور لووب اللي في النص ديه هتحسبلها كل قيم ال gradient .. وتقوم مستخدمها عشان تعمل update to theta ..

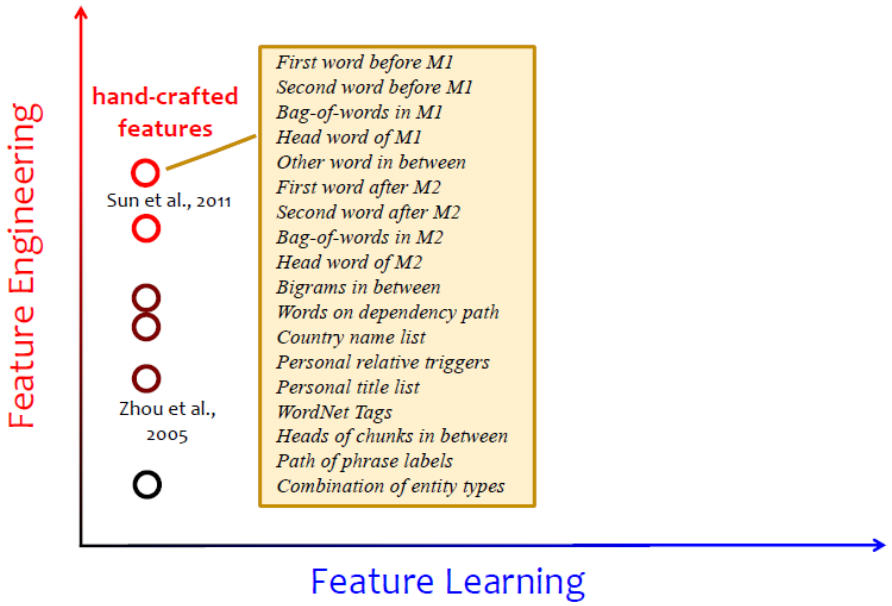
تعال دلوقت نفكّر في مشكله مهمه .. احنا اتكلمنا علي الفيتشرز كتير ... بس الفكره بنجيبهم منين ... افترض ان عندك جمله ... الجمله نفسها مملة .. انما ايه رأيك لو عدلنا علي الجمله مثلاً خليناها كلها lower-case او انك شلت شوية حاجات منها .. بعدها تقدر تضيف مثلا جزء من speech tags للجمله .. او تضيفلهم حاجات كتيره .. بص عالصوره ... اول ما يكون عندك كل ده .. دخلو في الفانكشن f و ده هيرجعلك فيتشر فيكتور ريبرزنتيشن لكل الستراكتشر ده .. الفيكتور ده هيبقا عامل ازاي .. تعال نبص ع الجمله

Handcrafted Features



هو مين برورياس ده .. هل ده متجوز يعني .. هل هوه موولود في مصر .. اشتري الكلب بتاعه من مصر .. عمل ايه يعني .. هنا بقا الموديل هيقدر انو ي pick out the correct relationship ما بين الحاجات ديه باستخدام الفيبيتشر فيكتور بتاعك .. ازاي الناس بتعمل كدا .. طريقه انهم بيعملو ريسيرش .. ازاي ن handcraft احسن فانكشن f ... بص ع الصوره .. انت هتكتب كود بيرن deterministic function وبيشد اهم الحاجات و يجيلك فيكتور طويل عريض كدا فيه كل الحاجات المهمه اللي انت جبتها .. زي مثلاً الحاجات اللي بالاصفر في الصوره اللي تحتينا ..

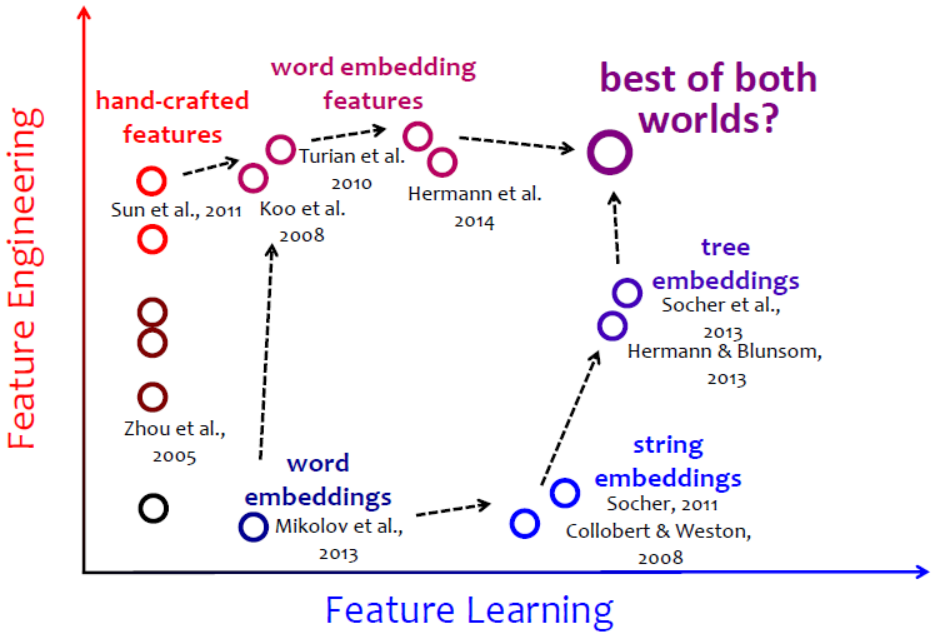
Where do features come from?



في حد فكّر في انو يستخدم ال word embeddings .. ايه يعني لو استخدمنا الماشين ليرننج انو يتعلم vector representations of the words .. الفكره ديه كانت جامده جداً وبقت تطير .. الناس بقت بتقول ان القطه ممكن تبقا فيكتور .. و الكلب ممكن يبقا فيكتور .. وعشان دول كلمتين similar to each other .. الفيكتور بتاعهم هيبقا شبه قريب من بعض ..المهم حد بعدها الدنيا شافها بتجري بسرعه اوي قال وليه مجريش انا كمان .. يلا بينا ندعك في اللي كلهم بيدعكو فيه .. وطلع وقال لما احنا هنعمل word embedding ليه منعملش embedding to the whole sentence .. و من هنا نقدر نجيب vector representation للجمله كلها بال CNNs او بال autoencoders .. بس اصبر .. في حد تالت قال .. ايه ده احنا اصلا عندنا ال syntax trees مرميين في كل حته من الناس اللي عملو ال feature handcraft .. ما يمكن نقدر نستخدمهم ونعمل embedding لشجره كامله في فيكتور ..

وبعدين الناس اللي كانت بتعمل handcrafting قالو .. الناس ديه شكلهم هيوصلو لحاجه ولا ايه .. تعال بسرعه ناخد منهم الافكار ديه و نلفها ونحطها في ال handcrafted features بتاعتنا نخليها أحسن وأحسن .. وفي الآخر الافكار بتتلاقي ..

Where do features come from?



16

تعال بقا نشوف ازاي نقدر نعمل feature handcrafting .. نفترض اننا هنبني logistic regression model .. وانت عاوز ت predict جزء من ال speech tag لكل كلمه في الجمله .. هتستهخدم انهي فيتشرز ..

Feature Engineering for NLP

Suppose you build a logistic regression model to predict a part-of-speech (POS) tag for each word in a sentence.

What features should you use?

17

تعال نسأل شوية أسئله عن الكلمات اللي عندنا .. وناخدها ترو وفولس مثلاً يعني

Feature Engineering for NLP

Per-word Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
is-capital(w_i)	1	0	1	0	0	0
endswith(w_i, "e")	1	1	0	0	0	1
endswith(w_i, "d")	0	0	0	1	1	0
endswith(w_i, "ed")	0	0	0	1	1	0
w_i == "aardvark"	0	0	0	0	0	0
w_i == "hope"	0	0	0	0	0	1
...	...	...	...	...	...	...

Feature Engineering for NLP

Context Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
...	...	...	...	...	...	...
w_i == "watched"	0	0	0	1	0	0
w_{i+1} == "watched"	0	0	1	0	0	0
w_{i-1} == "watched"	0	0	0	0	1	0
w_{i+2} == "watched"	0	1	0	0	0	0
w_{i-2} == "watched"	0	0	0	0	0	1
...	...	...	...	...	...	...

Feature Engineering for NLP

Context Features:

	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	$x^{(4)}$	$x^{(5)}$	$x^{(6)}$
...	...	...	...	...	...	...
w_i == "I"	0	0	1	0	0	0
w_{i+1} == "I"	0	1	0	0	0	0
w_{i-1} == "I"	0	0	0	1	0	0
w_{i+2} == "I"	1	0	0	0	0	0
w_{i-2} == "I"	0	0	0	0	1	0
...	...	...	...	...	...	...

Table from Manning (2011)

Feature Engineering for NLP

Table 3. Tagging accuracies with different feature templates and other changes on the WSJ 19-21 development set.

Model	Feature Templates	# Feats	Sent. Acc.	Token Acc.	Unk. Acc.
3gramMemm	See text	248,798	52.07%	96.92%	88.99%
naacl 2003	See text and [1]	460,552	55.31%	97.15%	88.61%
Replication	See text and [1]	460,551	55.62%	97.18%	88.92%
Replication'	+ rareFeatureThresh = 5	482,364	55.67%	97.19%	88.96%
5w	+ t_0, w_{-2}, t_0, w_2	730,178	56.23%	97.20%	89.03%
5wShapes	+ $t_0, s_{-1}, t_0, s_0, t_0, s_{+1}$	731,661	56.52%	97.25%	89.81%
5wShapesDS	+ distributional similarity	737,955	56.79%	97.28%	90.46%

@test set

adding more features

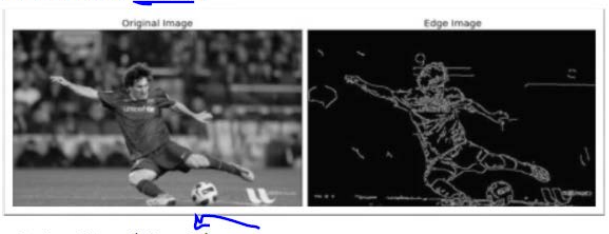
1% improvement only

21

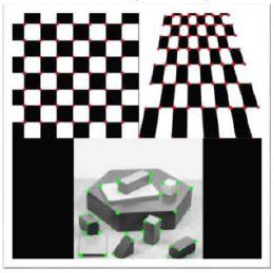
تعال نشوف شوية الكمبيوتر فيجن ..

Feature Engineering for CV

Edge detection (Canny)



Corner Detection (Harris)



ures from http://opencv.org

26

اللي عاوزين نتكلم عنو هو فكره عامه بحت .. اللي هو اننا نعمل some non-linear representation of your original representation .. عشان تفهم ده تعال نشوف أمثله ..

Nonlinear Features

- aka. "nonlinear basis functions"
- So far, input was always $\mathbf{x} = [x_1, \ldots, x_M]$
- **Key Idea:** let input be some function of $\mathbf{x}$
  - original input: $\mathbf{x} \in \mathbb{R}^M$
  - new input: $\mathbf{x}' \in \mathbb{R}^{M'}$ where $M' > M$ (usually)
  - define $\mathbf{x}' = b(\mathbf{x}) = [b_1(\mathbf{x}), b_2(\mathbf{x}), \ldots, b_{M'}(\mathbf{x})]$
    where $b_i : \mathbb{R}^M \rightarrow \mathbb{R}$ is any function
- **Examples:** (M = 1)

polynomial	$b_j(x) = x^j \quad \forall j \in \{1, \ldots, J\}$
radial basis function	$b_j(x) = \exp\left(\frac{-(x-\mu_j)^2}{2\sigma_j^2}\right)$
sigmoid	$b_j(x) = \frac{1}{1+\exp(-\omega_j x)}$
log	$b_j(x) = \log(x)$

For a linear model: still a linear function of b(**x**) even though a nonlinear function of **x**

Examples:
- Perceptron
- Linear regression
- Logistic regression

30

الفكره انك لو بتبني linear model .. وانت الموديل اللي انت بتتعلمو لما هتديلو b(x) هو لسه linear function of b(x) .. احنا مش بنغير الألجورذمز ولا الموديلز .. احنا بس عندنا non-linear function من الأنبوت إكس بتاعنا .. فالكلاسيفير بتاعنا بقا هو فعلياً non-linear in x حتي لو كان لسه linear in b(x) .. تعال نشوف الكلام ده .. حتي لو هنروح لل high dimensional feature space انا اقدر لسه اوريك الفانكشن في ال low-dimensional space ..

Example: Linear Regression

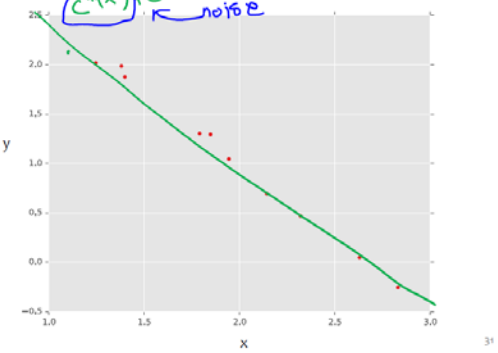
Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function

true function try to Model

$c^*(x) + \epsilon$ noise

y	x
2.0	1.2
1.3	1.7
0.1	2.7
1.1	1.9

true "unknown" target function is linear with negative slope and gaussian noise



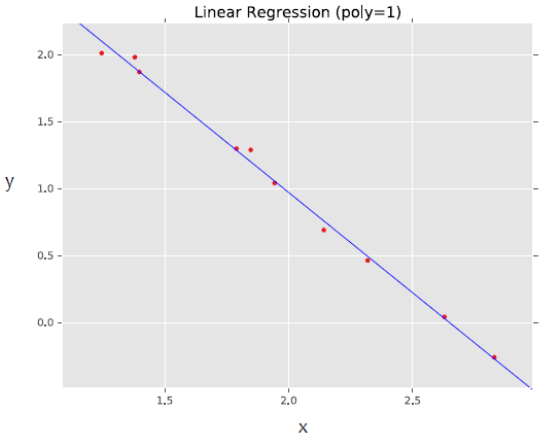
31

Example: Linear Regression

Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where f(.) is a polynomial basis function

y	x
2.0	1.2
1.3	1.7
0.1	2.7
1.1	1.9

true "unknown" target function is linear with negative slope and gaussian noise



32

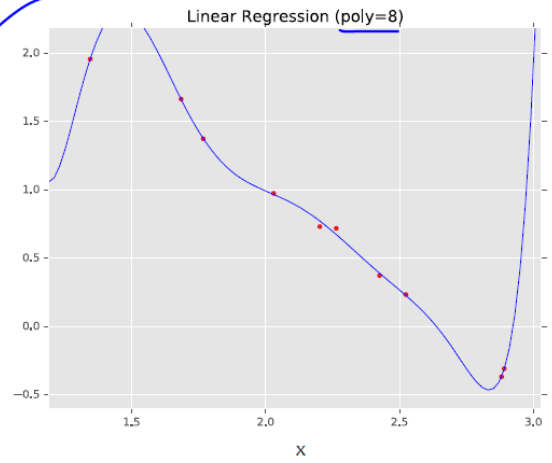
Example: Linear Regression

Goal: Learn $y = \mathbf{w}^T f(\mathbf{x}) + b$ where $f(.)$ is a polynomial basis function

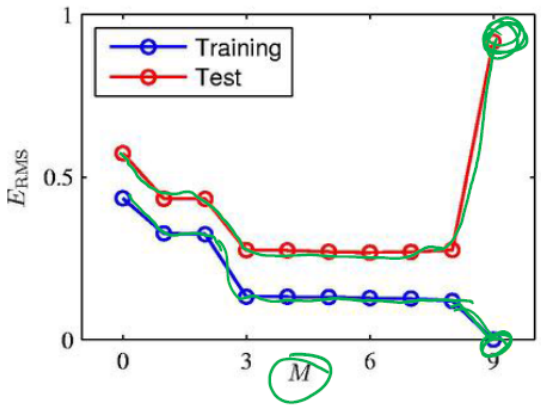
even more features

y	x	x^2	...	x^8
2.0	1.2	$(1.2)^2$	...	$(1.2)^8$
1.3	1.7	$(1.7)^2$	...	$(1.7)^8$
0.1	2.7	$(2.7)^2$	...	$(2.7)^8$
1.1	1.9	$(1.9)^2$	...	$(1.9)^8$

true "unknown" target function is linear with negative slope and gaussian noise



Over-fitting



Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Polynomial Coefficients

	$M=0$	$M=1$	$M=3$	$M=9$
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

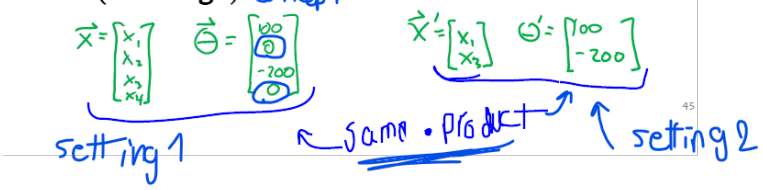
explosion

عند ال M=9 .. البامترز ظارت فوق جداً و الموديل شغال علي آخرو عشان يعمل أوفر فيتنج .. ازاي نحل المشكله ديه .. الدكتور كان مدينا 10 نقط بس .. طب لو خدنا نفس الفيتشر سيت اللي عندنا بس قمت مديلك 100 نقطه .. المشكله راحت خلاص .. ومعظم الأوفر فيتنج اتشال .. فبالتالي كل ما تقدر تضيف داتا انت هتمنع الأوفر فيتنج .. المشكله ان ايه اللي هيحصل لو معندكش داتا ... خش بالريجيولاريزيشن ...

تعال نفتكر ايه هو الأوفر فيتنج ... الأوفر فيتنج هو انو ي capture the noise in the data بدل ما ي capture the underlying structure .. ده بيحصل في كل الموديلز حرفياً .. عادي ... تعال نفكّر ليه بيحصل .. ناخد مثال stock market prediction ... ايه هي الفيتشرز اللي نقدر نستخدمها .. انت تقدر تاخد ال stock prices بتاعت كل الستوكس عند كل الاوقات .. وبعدين تبص علي كل newspaper ... و السوشال ميديا بيقولو ايه .. هل كل ده مهم؟ .. الستوكس بتاعت شركات صغيره .. ممكن كونو فعلاً بيزيدو بس هو حرفياً مفيش علاقه ما بين الي هم بيعملوه واللي جوجل بيعملوه ... فانت بتفضّل دايماً ال simple hypothesis .. لما كنا بنتكلم علي ال decision trees .. احنا قلنا ان شجره صغيره هي اسهل حاجه ... تعال نقول حاجه مختلفه سيكه .. شجره صغيره ليها فيتشرز صغيرين .. فبرضو ال linear models الي بنشتغل بيهم هيبقو احسن لو ليهم small number of features زي ما شفنا كدا من كام صوره فاتو ... زودت كتير اوي ع الفاضي ... في حاجه تانيه مهمه .. وهو انك يبقا عندك important features ..

Motivation: Regularization

- **Occam's Razor:** prefer the simplest hypothesis
- What does it mean for a hypothesis (or model) to be **simple**?
  1. small number of features (**model selection**)
  2. small number of "important" features (**shrinkage**)

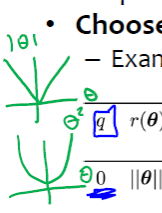


45

لو جيت تبص ع الصوره اللي فوق ديه علي طول .. هتلاقي ان ليهم نفس الدوت بروضكت ... هيبقا كويس لو فكّرت ايه اللي هيحصل لو كان الصفرين اللي عندك دول هيبقوا اقرب للصفر مش صفر علي طول .. ساعتها هيبقا الدوت برضكت علي الشمال هيبقا مختلف سيكه عن اليمين .. فلما تبني موديل ليه فيتشرز كتيره .. بس معظمهم قريبين من الصفر ... انت فعلياً اتعلمت موديل سهل .. وديه الفكره ورا ال regularization .. عندك objective J(theta) .. والهدف تلاقي theta hat اللي بتقلل فانكشن جديده .. الهدف من ال J(theta) هنا انك ت fit the data .. لو انت عملت مينيمايزينج لل J(theta) بيقا انت اتعلمت موديل كويس ... الهدف من ال r(theta) هو بيحاول ي comabat overfitting .. ازاي نختار الفورم بتاعت ال r(theta) ..

Regularization

- **Given** objective function: J(θ)
- **Goal** is to find: $\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$
  fit the data — combat overfitting
- **Key idea:** Define regularizer r(θ) s.t. we tradeoff between fitting the data and keeping the model simple
- **Choose form of r(θ):**
  - Example: q-norm (usually p-norm) $r(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_q = \left[\sum_{m=1}^{M} ||\theta_m||^q\right]^{(\frac{1}{q})}$

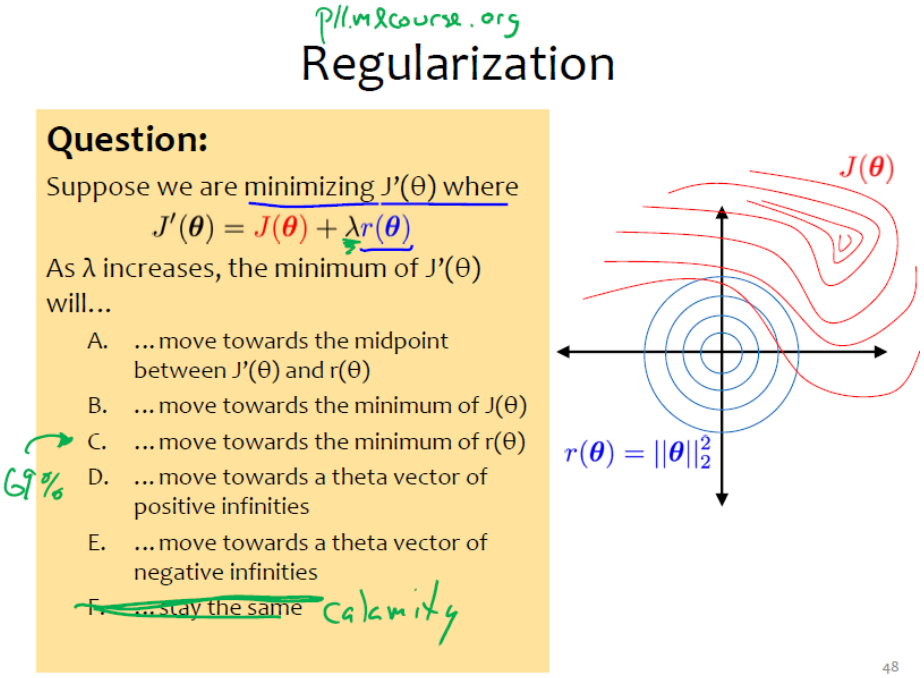


q	$r(\boldsymbol{\theta})$	yields parameters that are...	name	optimization notes
0	$\|\|\boldsymbol{\theta}\|\|_0 = \sum \mathbb{1}(\theta_m \neq 0)$	zero values	L0 reg.	no good computational solutions
1	$\|\|\boldsymbol{\theta}\|\|_1 = \sum \|\theta_m\|$	zero values	L1 reg.	subdifferentiable
2	$(\|\|\boldsymbol{\theta}\|\|_2)^2 = \sum \theta_m^2$	small values	L2 reg.	differentiable

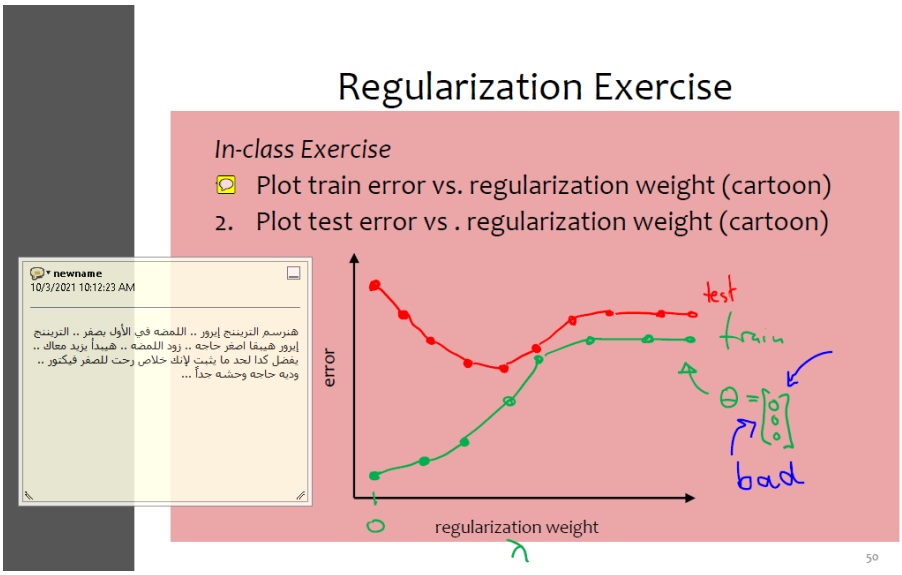
SGD

46

عند ال q = 0 بتعد عدد ال non-zero entries in my theta vector. . لو عملت مينيمايزيشن لده .. فده هيفضّل الارقام الكبيره .... خد بالك ان ده binary indicator مش سهل انك تشتقه .... لو انت خدت ال q = 1 ... تقدر تستخدم SGD عشان ت optimize الحوارده .. اللي مشهور أوي هو ال L2 regularizer وهو انك تاخد ال sum of squares of the entries of the theta vector .... هو بيرشّح ارقام صغيره في ال feature vector....



لو اللمضه كانت بصفر .. المنيمم هيبقا المنيمم بتاع ال J(theta) لو اللمضه كانت أكبر من الصفر سيكه ... كإنك بتجمع الاتنين مع بعض .. انت هتروح ناحية المنيمم بتاع ال r(theta) الي هو الصفر.. لما اللمضه تزيد شوية هتشد ناحية الصفر ... اللمضه تكبر أوي أوي .. ال J(theta) مقارنةً بال r(theta) هتبقا صغيره جداً ... كل ما اللمضه تكبر انت رايح ناحية فيكتور من الاصفار ..



بالنسبه للتست ايرور هيبدأ عالي لإنك جاي من الأوفر فيتتنج .... وينزل معاك لإن البارمترز اللي كانت في رينج المليون عماله تقل ... فانت بتبقا احسن و احسن واحسن لحد ما تلاقي المنيمم وبعدين شوية وتثبت ...

أول حاجه don't regularize the bias parameter .. خد بالك ان ال regularization لازم يفكس لل bias parameter .. يعني اه انت بتضيف لل x بتاعتك 1 علي اساس انك هتضيف ال bias جوا الثيتا .. بس انت لازم متضفش عليه ترم ال regularization ... ليه ... فكّر في الريجريشن موديل .. انت هيبقا عندك learning algorithm هيبقا invariant to a shift in y values ... فلو عندك شوية نقط لل regression... فهتقدر تشيفتهم لتحت .. غير كدا هيبقو زي بعض .. السلوب نفسو انت مش عاوز يتغير انما انت بتعوز تتحرك فوق و تحت .. ده برضو بيتطبق علي الريجريشن .. انت عاوز تشيفت الديسجن باوندري

Regularization

Don't Regularize the Bias (Intercept) Parameter!

- In our models so far, the bias / intercept parameter is usually denoted by θ_0 -- that is, the parameter for which we fixed $x_0 = 1$
- Regularizers always avoid penalizing this bias / intercept parameter
- Why? Because otherwise the learning algorithms wouldn't be invariant to a shift in the y-values for regression x-values for classification

Whitening Data

- It's common to *whiten* each feature by subtracting its mean and dividing by its variance
- For regularization, this helps all the features be penalized in the same units
  (e.g. convert both centimeters and kilometers to z-scores)

انا فصلت في آخر 10 دقايق