Regression

1. Hypothesis & Cost/loss function (MSE):

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\underbrace{h(\mathbf{x}_n)}_{\mathbf{w}^T\mathbf{x}_n} - y_n)^2$$

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

2. Gradient decent: one takes steps proportional to the negative of the gradient **how to avoid local minima?**

Learning rate: too small -> slow convergence/ too large -> J(w) may increase on every iteration, may not converge.

Gradient descent:

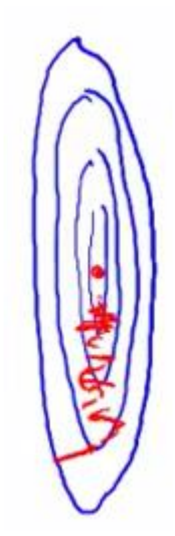
```
    Repeat {
→       $\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\ldots,\theta_n)$
    }                       (simultaneously update for every $j = 0,\ldots,n$)
```

Take a derivative of cost function:

$$\frac{\partial J(\theta)}{\partial\theta_j} = \sum_i x_j^{(i)}\left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

```
Repeat {
                        $\frac{\partial}{\partial\theta_j}J(\theta)$
→   $\theta_j := \theta_j - \alpha\boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}$
                        (simultaneously update $\theta_j$ for
                        $j = 0,\ldots,n$)
}
```

3. Feature scaling: make sure features are on the similar scale

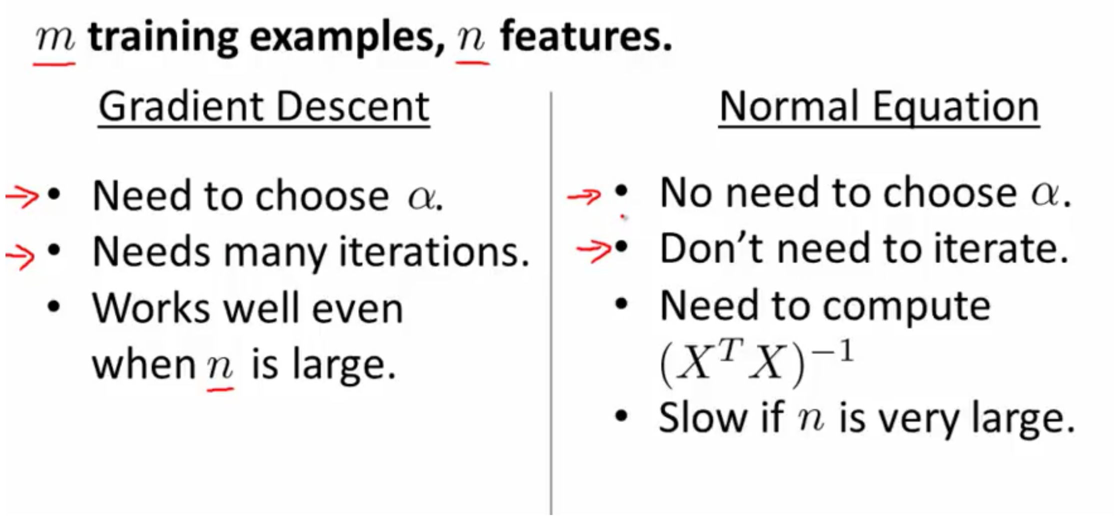


too skinny, long way to minimum.

mean normalization: x = (x-mean)/range range = maximum - minimum

4. Normal equation:

Gradient descent vs Normal equation: when n (>10000) is large, choose gradient descent



m **training examples,** n **features.**

Gradient Descent	Normal Equation
• Need to choose α.	• No need to choose α.
• Needs many iterations.	• Don't need to iterate.
• Works well even when n is large.	• Need to compute $(X^TX)^{-1}$
	• Slow if n is very large.

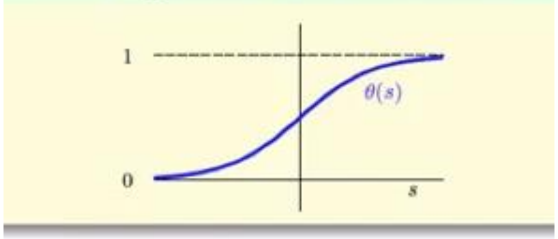
Logistic regression

5. Hypothesis & logistic function

H(x) = 0.8, P(1|x) = 0.8, P(0|x) = 0.2.

logistic hypothesis: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

Logistic Function



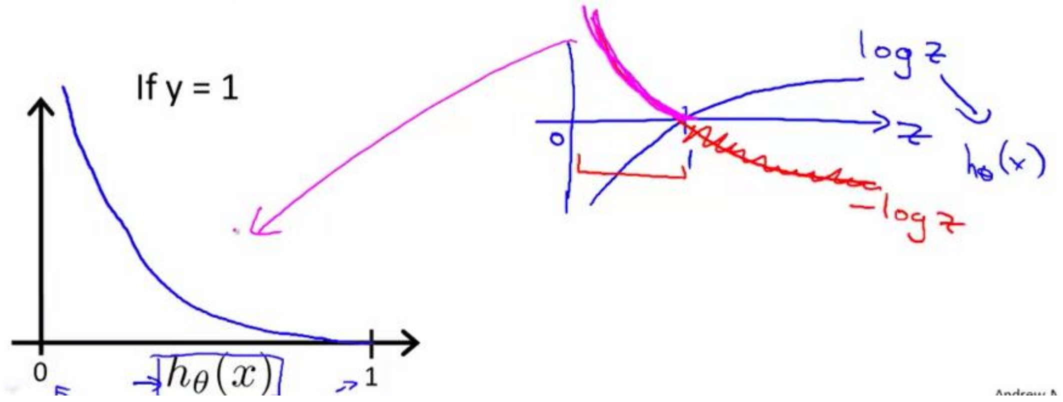
$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

6. Cost function & Gradient decent for logistic regression:

Choose cost function: If use the same cost function as linear regression, it will lead to non-convex problem. We need a different cost function.

Logistic regression cost function

$$\mathrm{Cost}(\underline{h_\theta(x)}, \overset{}{\underset{\uparrow}{y}}) = \begin{cases} \boxed{-\log(h_\theta(x))} & \text{if } y=1 \\ \underline{-\log(1-h_\theta(x))} & \text{if } y=0 \end{cases}$$



Logistic regression cost function

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

$$= -\frac{1}{m}\Big[\sum_{i=1}^{m} y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log\big(1-h_\theta(x^{(i)})\big)\Big]$$

This is a Maximum likelihood estimation.
Maximum likelihood = minimum error.
求最大的 training set 出现可能性，之后取反，变求最大值为求最小值。相当于最大化可能性，同时也最小化 error

probability that f generates $\mathcal{D}$	likelihood that h generates $\mathcal{D}$
$P(\mathbf{x}_1)P(\circ\vert\mathbf{x}_1)\ \times$	$P(\mathbf{x}_1)h(\mathbf{x}_1)\ \times$
$P(\mathbf{x}_2)P(\times\vert\mathbf{x}_2)\ \times$	$P(\mathbf{x}_2)(1-h(\mathbf{x}_2))\ \times$
$\ldots$	$\ldots$
$P(\mathbf{x}_N)P(\times\vert\mathbf{x}_N)$	$P(\mathbf{x}_N)(1-h(\mathbf{x}_N))$

$$\text{likelihood}(h) = P(\mathbf{x}_1)h(\mathbf{x}_1) \times P(\mathbf{x}_2)(1-h(\mathbf{x}_2)) \times \ldots P(\mathbf{x}_N)(1-h(\mathbf{x}_N))$$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$

$$\text{likelihood}(\text{logistic } h) \propto \prod_{n=1}^{N} h(y_n\mathbf{x}_n)$$

$$h(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$$

Cross-Entropy Error

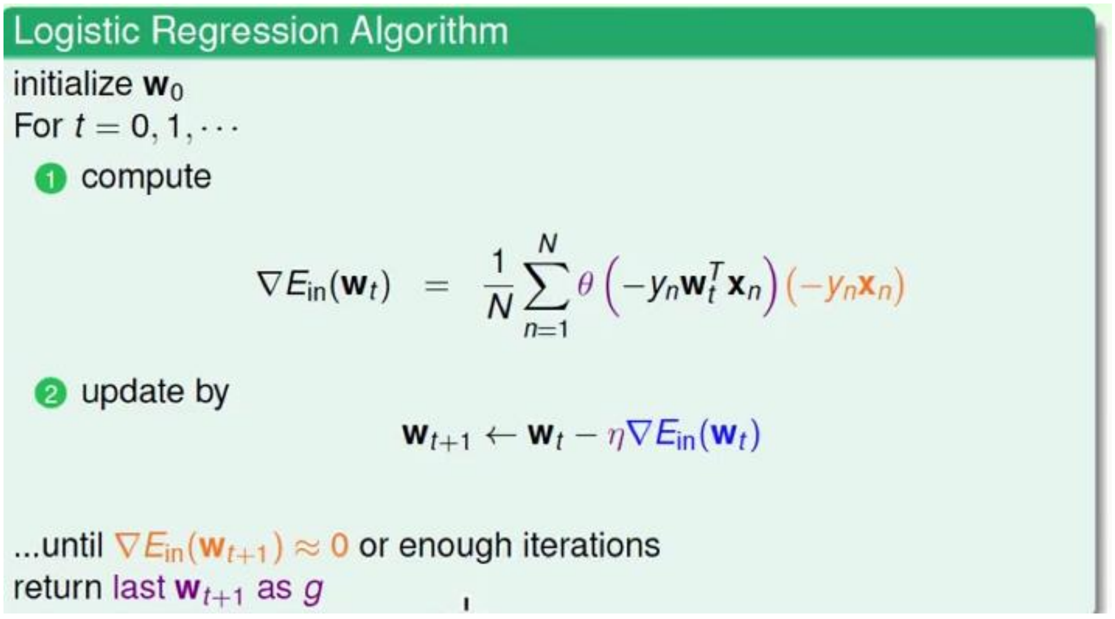
$$\min_{\mathbf{w}} \quad \frac{1}{N}\sum_{n=1}^{N} - \ln\theta\left(y_n\mathbf{w}^T\mathbf{x}_n\right)$$

$$\theta(s) = \frac{1}{1+\exp(-s)} \quad : \quad \min_{\mathbf{w}} \quad \frac{1}{N}\sum_{n=1}^{N} \ln\left(1+\exp(-y_n\mathbf{w}^T\mathbf{x}_n)\right)$$

$$\Longrightarrow \quad \min_{\mathbf{w}} \quad \underbrace{\frac{1}{N}\sum_{n=1}^{N} \operatorname{err}(\mathbf{w}, \mathbf{x}_n, y_n)}_{E_{\text{in}}(\mathbf{w})}$$

Pay attention to the new error function!
In gbdt we use the similar error function?

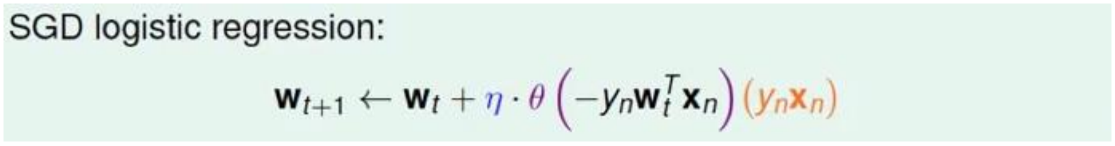
$$L(y, f(x)) = \exp(-y\, f(x)). \qquad (10.8)$$



Logistic Regression Algorithm

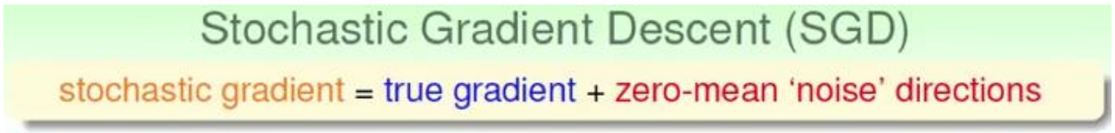
```
initialize $\mathbf{w}_0$
For $t = 0, 1, \cdots$
    1 compute
        $\nabla E_{\text{in}}(\mathbf{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$
    2 update by
        $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$
...until $\nabla E_{\text{in}}(\mathbf{w}_{t+1}) \approx 0$ or enough iterations
return last $\mathbf{w}_{t+1}$ as $g$
```

7. Stochastic gradient decent regression:
之前每次 iteration 都是 O(n)，因为需要求所有的样本之和，现在用一个随机值来代替求和平均的过程，把 iteration 消耗降低为 O(1)。和 PLA 很接近 (随机的 n 是哪儿来的？可能就是随机的)。很难得到精确解。所以只选择固定循环次数。不选择当梯度为 0.
The time complexity of normal lr is O(n), since we need to calculate all the samples in the training set. Now we want to reduce the time complexity to O(1). We only need to randomly calculate the gradient of one sample instead of all samples. However, it is hard to get an exact solution using stochastic lr, so we stop iterations by setting a fixed iteration times.



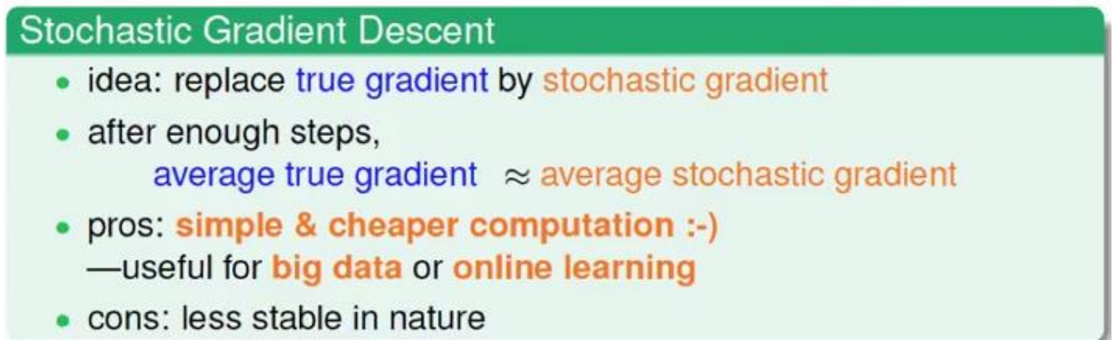
SGD logistic regression:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)$$



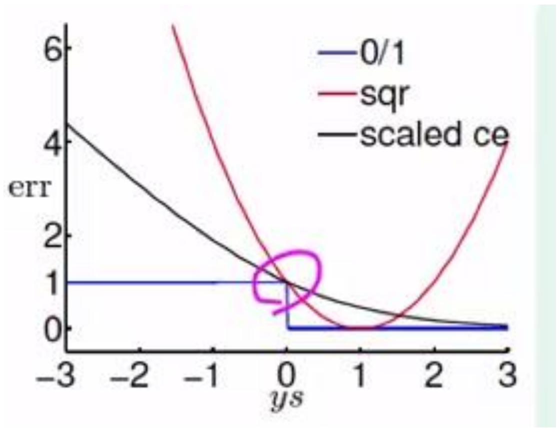
Stochastic Gradient Descent (SGD)

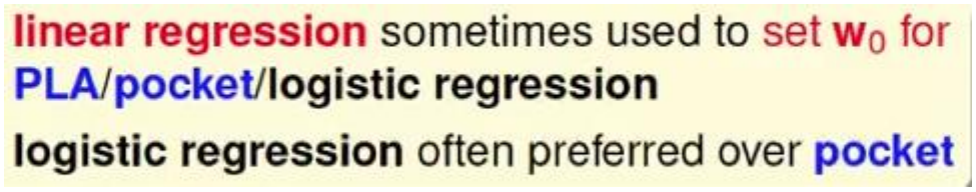
stochastic gradient = true gradient + zero-mean 'noise' directions



Stochastic Gradient Descent

- idea: replace true gradient by stochastic gradient
- after enough steps,
  average true gradient $\approx$ average stochastic gradient
- pros: **simple & cheaper computation :-)**
  —useful for **big data** or **online learning**
- cons: less stable in nature




linear regression sometimes used to set $\mathbf{w}_0$ for **PLA/pocket/logistic regression**

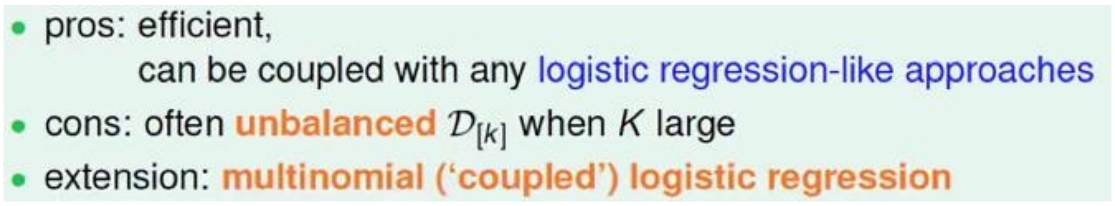
logistic regression often preferred over **pocket**

8. Multiclass problem:

用 logistic regression 进行 soft classification 比较分数，可以不用跑 logistic function 因为单调性。 Compared two logistic score, we do not need to calculate logistic function because of monotonicity.

One-Versus-All (OVA) Decomposition

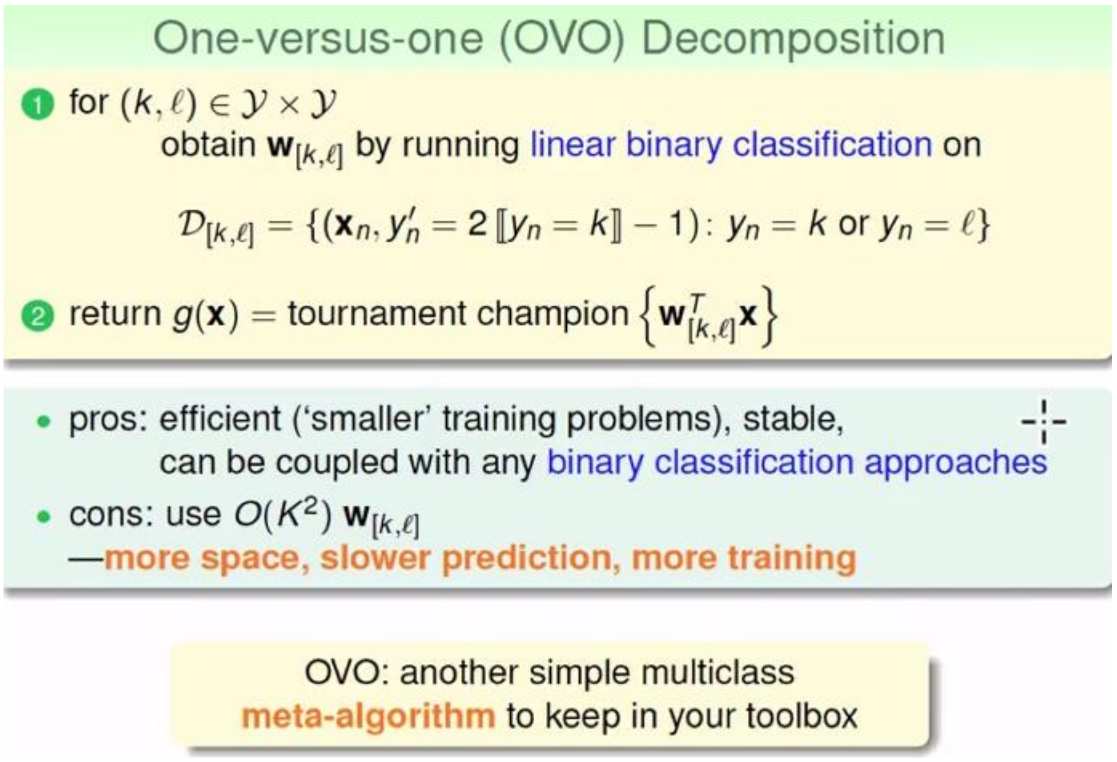
Cons: If the number of classes is large, the score for each class would be so small. We have to choose the biggest score among many small candidates. The result of classification is not very good. 如果有 100 类，每个 classifier 很容易猜非这一类。影响结果。


- pros: efficient,
  can be coupled with any logistic regression-like approaches
- cons: often **unbalanced** $\mathcal{D}_{[k]}$ when K large
- extension: **multinomial ('coupled') logistic regression**

To solve unbalance problem: One versus one

有效率因为训练所需 training set 很小。

Efficient! Because of the small size of training set.



One-versus-one (OVO) Decomposition

1. for $(k, \ell) \in \mathcal{Y} \times \mathcal{Y}$
   obtain $\mathbf{w}_{[k,\ell]}$ by running linear binary classification on

$$\mathcal{D}_{[k,\ell]} = \{(\mathbf{x}_n, y_n' = 2 [\![y_n = k]\!] - 1) \colon y_n = k \text{ or } y_n = \ell\}$$

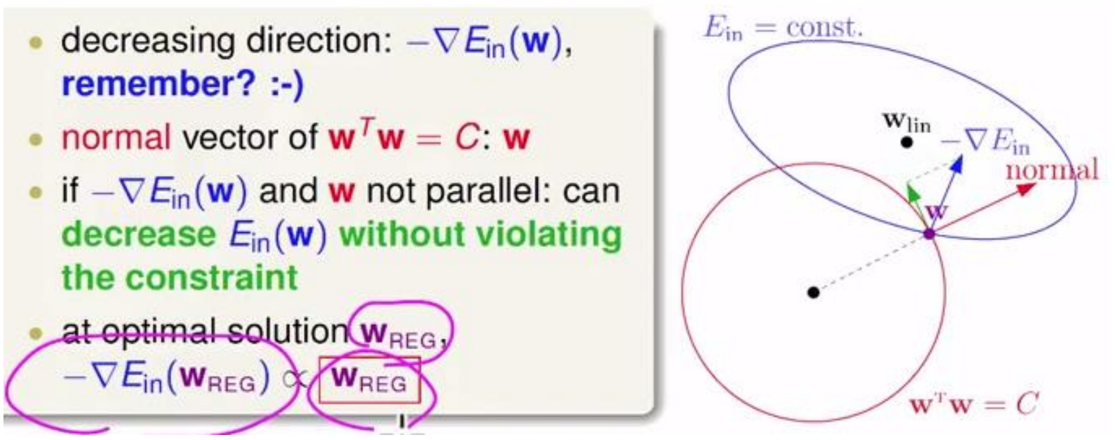
2. return $g(\mathbf{x}) = \text{tournament champion} \left\{ \mathbf{w}_{[k,\ell]}^T \mathbf{x} \right\}$

- pros: efficient ('smaller' training problems), stable,
  can be coupled with any binary classification approaches
- cons: use $O(K^2)$ $\mathbf{w}_{[k,\ell]}$
  —**more space, slower prediction, more training**

OVO: another simple multiclass
meta-algorithm to keep in your toolbox

Regularization:

9. L1 and L2



- decreasing direction: $-\nabla E_{\text{in}}(\mathbf{w})$, **remember? :-)**
- normal vector of $\mathbf{w}^T\mathbf{w} = C$: $\mathbf{w}$
- if $-\nabla E_{\text{in}}(\mathbf{w})$ and $\mathbf{w}$ not parallel: can **decrease $E_{\text{in}}(\mathbf{w})$ without violating the constraint**
- at optimal solution $\mathbf{w}_{\text{REG}}$, $-\nabla E_{\text{in}}(\mathbf{w}_{\text{REG}}) \propto \boxed{\mathbf{w}_{\text{REG}}}$

$E_{\text{in}} = \text{const.}$
$\mathbf{w}_{\text{lin}}$
$-\nabla E_{\text{in}}$
normal
$\mathbf{w}$
$\mathbf{w}^T\mathbf{w} = C$

Find the minimum value with constrain. Usually the solution is in the verge of red ball. If the inverse of gradient and normal are not parallel, this solution can decrease in the direction of green vector.

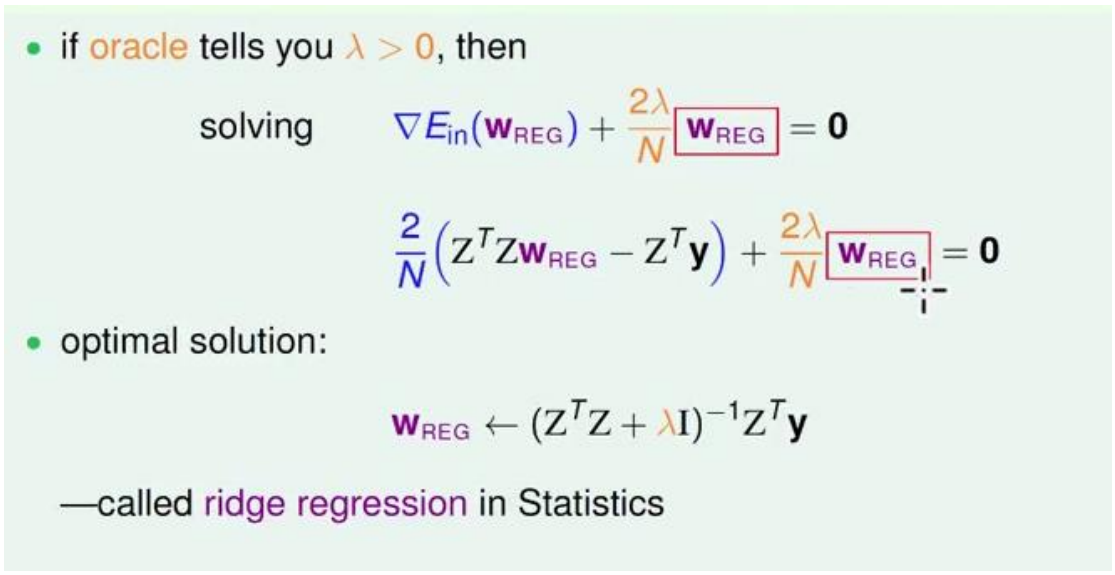
How to choose lambda?

用的 lagrange multiplier 把有 constrain 的 optimization 问题转换。

10. Ridge regression:

L2 solve the problem of invertible matrix of X^TX.

L2 解决了(X^TX)不可逆的问题。因为他又添加了一项。



- if oracle tells you $\lambda > 0$, then

solving $$\nabla E_{\text{in}}(\mathbf{w}_{\text{REG}}) + \frac{2\lambda}{N}\boxed{\mathbf{w}_{\text{REG}}} = \mathbf{0}$$

$$\frac{2}{N}\left(\mathrm{Z}^T\mathrm{Z}\mathbf{w}_{\text{REG}} - \mathrm{Z}^T\mathbf{y}\right) + \frac{2\lambda}{N}\boxed{\mathbf{w}_{\text{REG}}} = \mathbf{0}$$

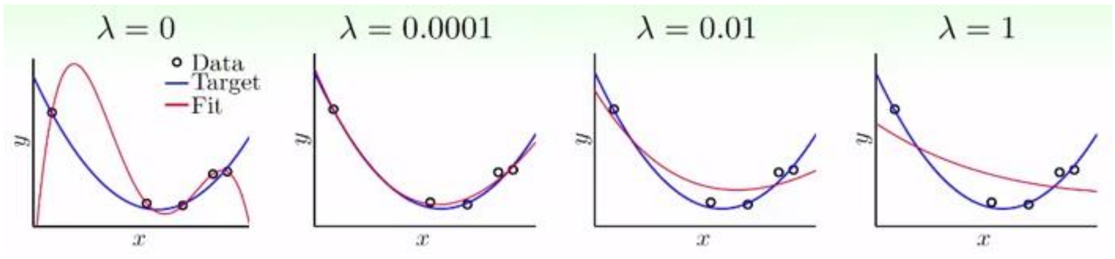
- optimal solution:

$$\mathbf{w}_{\text{REG}} \leftarrow (\mathrm{Z}^T\mathrm{Z} + \lambda \mathrm{I})^{-1}\mathrm{Z}^T\mathbf{y}$$

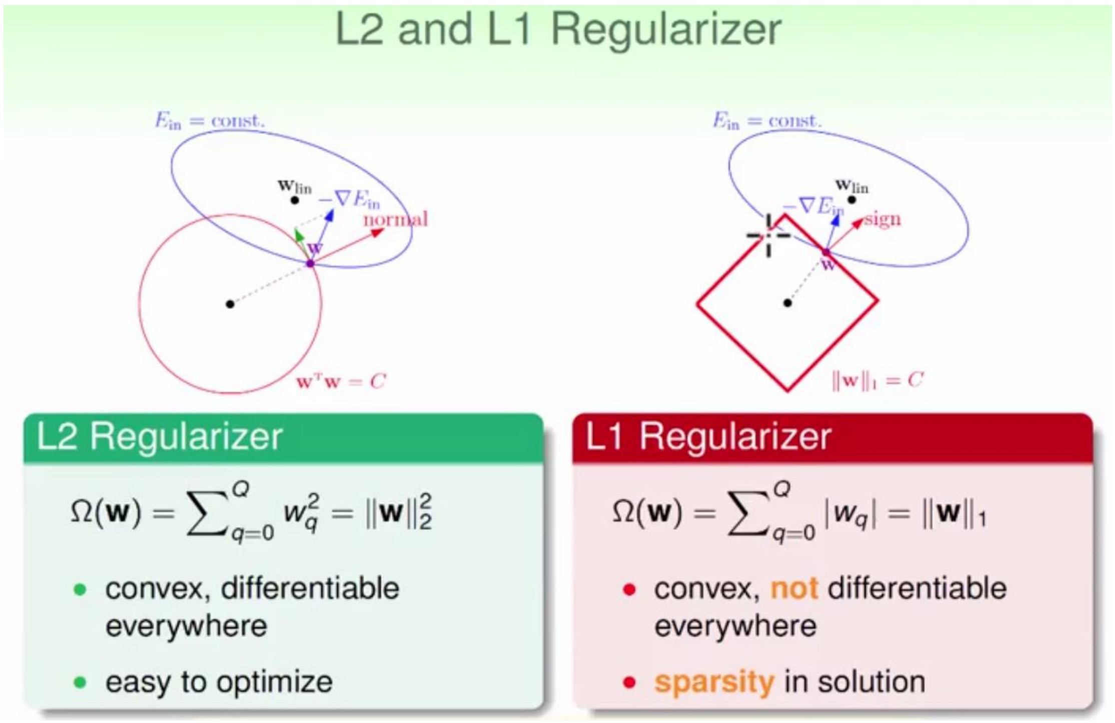
—called ridge regression in Statistics

Also, notice that the summation after λ does not include θ_0^2. intercept value.

Lambda:



Lasso get sparse since the solution always at vertex of red square. 因为参考左边图，沿着绿线走到头就是顶点


$E_{\text{in}} = \text{const.}$
$\mathbf{w}_{\text{lin}}$
$-\nabla E_{\text{in}}$
normal
$\mathbf{w}$
$\mathbf{w}^T\mathbf{w} = C$
$E_{\text{in}} = \text{const.}$
$\mathbf{w}_{\text{lin}}$
$-\nabla E_{\text{in}}$
sign
$\mathbf{w}$
$\|\mathbf{w}\|_1 = C$

L2 and L1 Regularizer

L2 Regularizer

$$\Omega(\mathbf{w}) = \sum_{q=0}^{Q} w_q^2 = \|\mathbf{w}\|_2^2$$

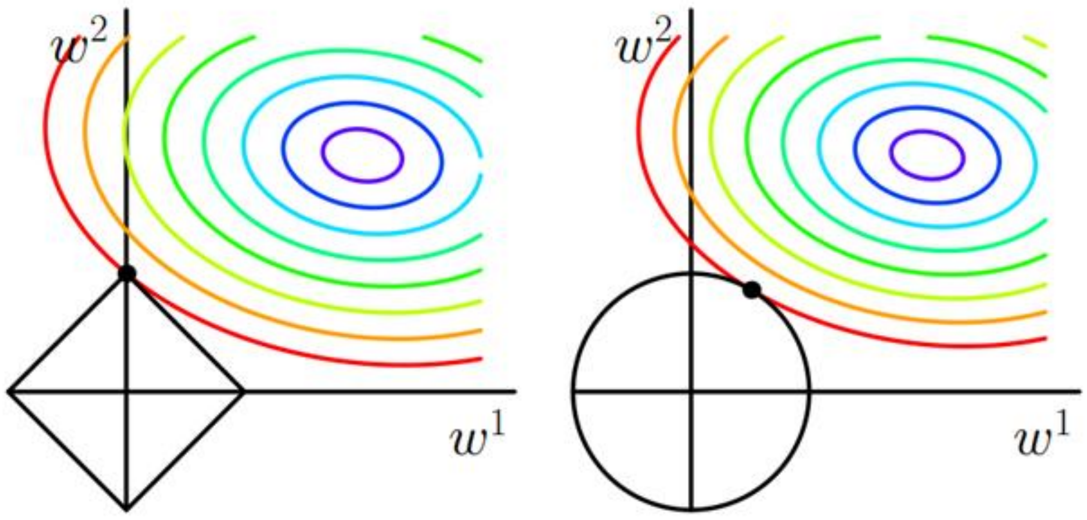
- convex, differentiable everywhere
- easy to optimize

L1 Regularizer

$$\Omega(\mathbf{w}) = \sum_{q=0}^{Q} |w_q| = \|\mathbf{w}\|_1$$

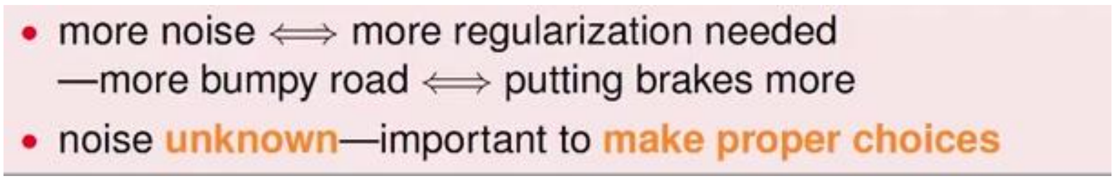
- convex, **not** differentiable everywhere
- **sparsity** in solution

L2 is differentiable everywhere，smooth，easily optimize. On the contrary L1 is not differentiable at vertex.



(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes."

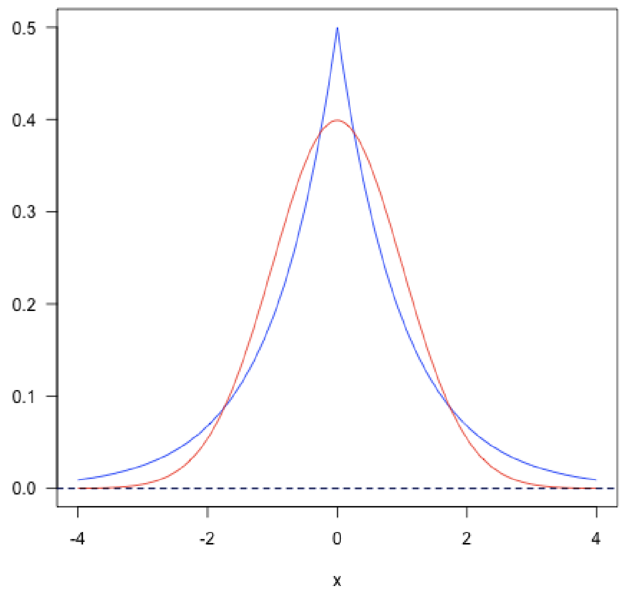


- more noise $\Longleftrightarrow$ more regularization needed —more bumpy road $\Longleftrightarrow$ putting brakes more
- noise unknown—important to make proper choices

11. In math, regularization means maximum a posterior. L1 and L2 use different prior probability.

As you can see, L1/Laplace tends to tolerate both large values as well as very small values of coefficients more than L2/Gaussian (tails).

Blue is Laplace density, red is Gaussian density



- MAP

$$
\begin{aligned}
w &= \arg\max_{w} p(w \mid D) \\
&= \arg\max_{w} \frac{p(D \mid w)p(w)}{p(D)} \\
&= \arg\max_{w} p(D \mid w)p(w) \\
&= \arg\max_{w} [\log p(D \mid w) + \log p(w)]
\end{aligned}
$$

L2 regularization : Gaussian Prior

- Gaussian prior

$$p(w) = \prod_{k=1}^{K} Norm(0,\delta_k^2)(w_k)$$

$$st,\ Norm(0,\delta_k^2) = \frac{1}{\delta_k\sqrt{2\pi}}\exp(-\frac{w_k^2}{2\delta_k^2})$$

- MAP

$$w_{MAP} = \arg\max_{w}[\log p(D\,|\,w) + \log p(w)]$$
$$= \arg\max_{w}[\sum_{i=1}^{N}\log(y^i\,|\,x^i,w) - c\sum_{k=1}^{K}w_k^2]$$

- Gradient descent step

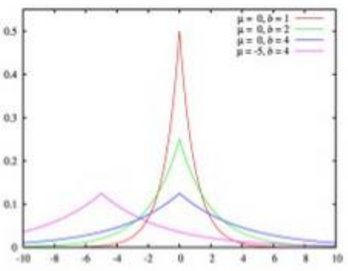
$$w_k \leftarrow w_k + \eta[(y^i - p(y^i = 1\,|\,x^i,w))x_k^i - \lambda w_k]$$

L1 regularization : Laplace Prior

- Laplace prior

$$p(w) = \prod_{k=1}^{K} Laplace(0,\delta_k^2)(w_k)$$

$$Laplace(0,\delta_k^2) = \frac{1}{\delta_k\sqrt{2}}\exp(-\sqrt{2}\frac{|w_k|}{\delta_k})$$



- MAP

$$w_{MAP} = \arg\min_{w} -[\sum_{i=1}^{N}\log(y^i\,|\,x^i,w) - c\sum_{k=1}^{K}|w_k|]$$

- Gradient descent step

$$w_k \leftarrow w_k + \eta[(y^i - p(y^i = 1\,|\,x^i,w))x_k^i - I(w_k > 0)\lambda]$$

L1 follows a Laplace distribution，

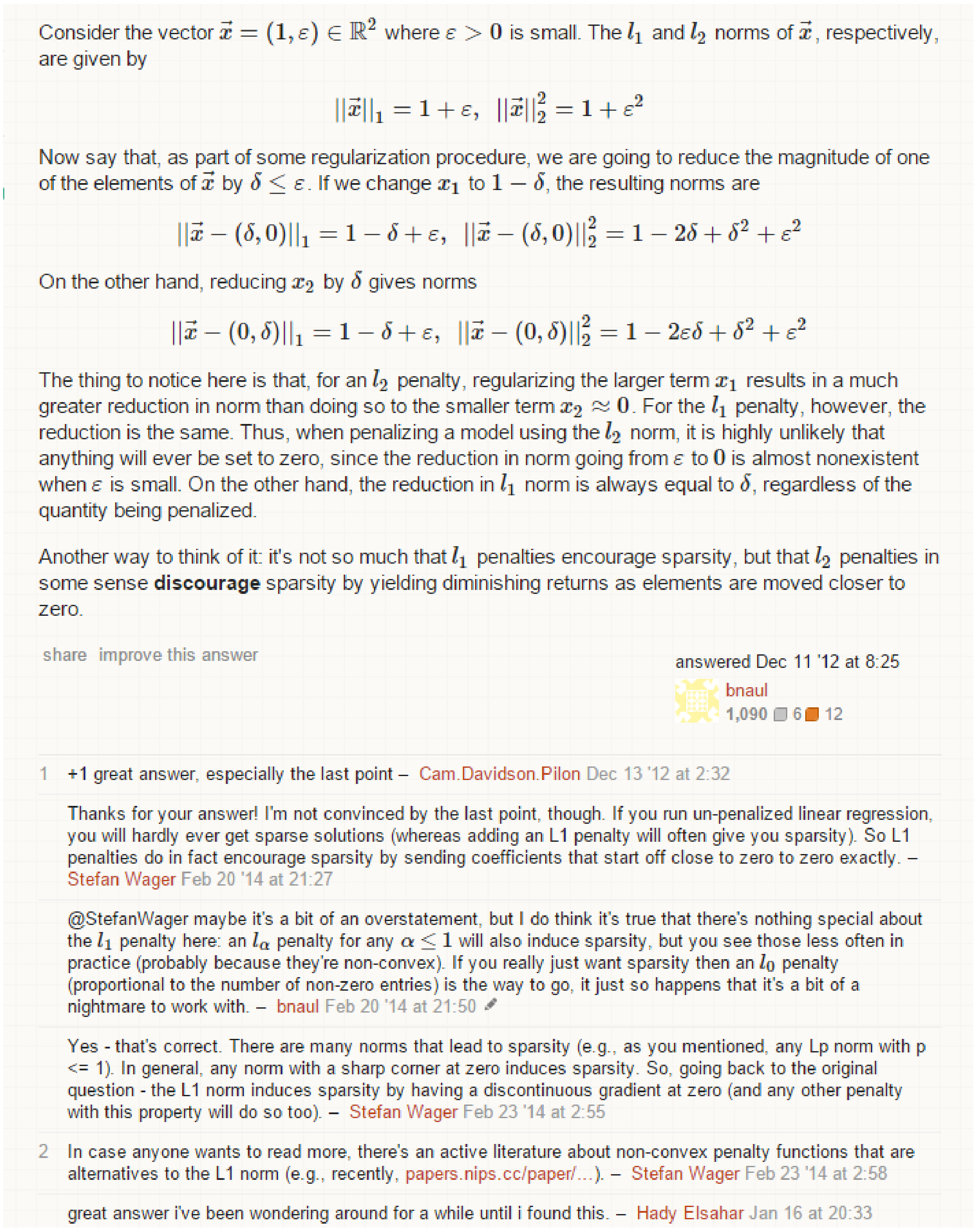
L1 usually corresponds to setting a Laplacean prior on the regression coefficients - and picking a maximum a posteriori hypothesis. L2 similarly corresponds to Gaussian prior. As one moves away from zero, the probability for such a coefficient grows progressively smaller

L1 can be used for feature selection because of sparsity. L1 is interpretability since it only highlights several important features instead of all the features. L2 is used for shrinking weights, in order to prevent overfitting.

_ℓ1_ vs _ℓ2_ for prediction:
Typically ridge or ***ℓ2*** penalties are much better for minimizing prediction error rather than ***ℓ1*** penalties. The reason for this is that when two predictors are highly correlated, ***ℓ1*** regularizer will simply pick one of the two predictors. In contrast, the ***ℓ2*** regularizer will keep both of them and jointly shrink the

corresponding coefficients a little bit. Thus, while the ***ℓ1*** penalty can certainly reduce overfitting, you may also experience a loss in predictive power.

过拟合是因为某一个 feature 过于被看重，w 值过于高，所以用 L2 来降低拟合，因为平方对大的数值惩罚度很高，所以 L2 结果倾向于比较小的数。对于 L1，大家都说的是那个画图的意思，就是凡是在零点具有 discontinuous gradient 的都有 sparsity 的特性。
Overfitting means that we think highly of one feature. Square is a big penalty for large number. Thus L2 tends to choose small result.
One distribution which has discontinuous gradient at zero is usually sparse? Any norm with a sharp corner at zero induces sparsity!
ordinary least squares (OLS)
Generalized least squares (GLS)



Consider the vector $\vec{x} = (1, \varepsilon) \in \mathbb{R}^2$ where $\varepsilon > 0$ is small. The l_1 and l_2 norms of $\vec{x}$, respectively, are given by

$$||\vec{x}||_1 = 1 + \varepsilon, \quad ||\vec{x}||_2^2 = 1 + \varepsilon^2$$

Now say that, as part of some regularization procedure, we are going to reduce the magnitude of one of the elements of $\vec{x}$ by $\delta \leq \varepsilon$. If we change x_1 to $1 - \delta$, the resulting norms are

$$||\vec{x} - (\delta, 0)||_1 = 1 - \delta + \varepsilon, \quad ||\vec{x} - (\delta, 0)||_2^2 = 1 - 2\delta + \delta^2 + \varepsilon^2$$

On the other hand, reducing x_2 by δ gives norms

$$||\vec{x} - (0, \delta)||_1 = 1 - \delta + \varepsilon, \quad ||\vec{x} - (0, \delta)||_2^2 = 1 - 2\varepsilon\delta + \delta^2 + \varepsilon^2$$

The thing to notice here is that, for an l_2 penalty, regularizing the larger term x_1 results in a much greater reduction in norm than doing so to the smaller term $x_2 \approx 0$. For the l_1 penalty, however, the reduction is the same. Thus, when penalizing a model using the l_2 norm, it is highly unlikely that anything will ever be set to zero, since the reduction in norm going from ε to 0 is almost nonexistent when ε is small. On the other hand, the reduction in l_1 norm is always equal to δ, regardless of the quantity being penalized.

Another way to think of it: it's not so much that l_1 penalties encourage sparsity, but that l_2 penalties in some sense **discourage** sparsity by yielding diminishing returns as elements are moved closer to zero.

share improve this answer

answered Dec 11 '12 at 8:25
bnaul
1,090 ▪6 ▪12

1 +1 great answer, especially the last point – Cam.Davidson.Pilon Dec 13 '12 at 2:32

Thanks for your answer! I'm not convinced by the last point, though. If you run un-penalized linear regression, you will hardly ever get sparse solutions (whereas adding an L1 penalty will often give you sparsity). So L1 penalties do in fact encourage sparsity by sending coefficients that start off close to zero to zero exactly. – Stefan Wager Feb 20 '14 at 21:27

@StefanWager maybe it's a bit of an overstatement, but I do think it's true that there's nothing special about the l_1 penalty here: an l_α penalty for any $\alpha \leq 1$ will also induce sparsity, but you see those less often in practice (probably because they're non-convex). If you really just want sparsity then an l_0 penalty (proportional to the number of non-zero entries) is the way to go, it just so happens that it's a bit of a nightmare to work with. – bnaul Feb 20 '14 at 21:50

Yes - that's correct. There are many norms that lead to sparsity (e.g., as you mentioned, any Lp norm with p <= 1). In general, any norm with a sharp corner at zero induces sparsity. So, going back to the original question - the L1 norm induces sparsity by having a discontinuous gradient at zero (and any other penalty with this property will do so too). – Stefan Wager Feb 23 '14 at 2:55

2 In case anyone wants to read more, there's an active literature about non-convex penalty functions that are alternatives to the L1 norm (e.g., recently, papers.nips.cc/paper/...). – Stefan Wager Feb 23 '14 at 2:58

great answer i've been wondering around for a while until i found this. – Hady Elsahar Jan 16 at 20:33

Reference:

MAP:
http://www.slideshare.net/guo_dong/logistic-regressionpptx
Difference between L1 and L2:
http://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization
L0, L1, L2:
http://blog.csdn.net/zouxy09/article/details/24971995/
http://www.stat.columbia.edu/~gelman/research/published/priors11.pdf
feature selection of regression and regularization
http://blog.datadive.net/selecting-good-features-part-ii-linear-models-and-regularization/