Lecture 12: Linear regression

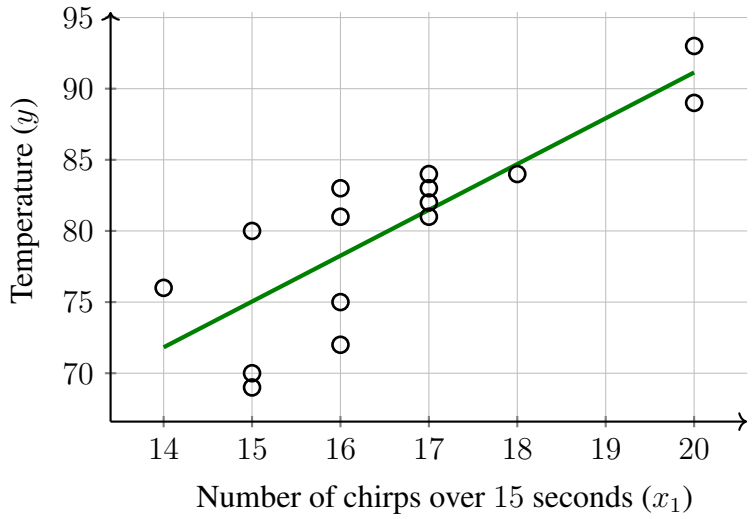
Tushaar Gangavarapu (TG352@cornell.edu)

Up to this point in the course, we have explored various interconnected themes. From a learning perspective, we examined supervised learning (data with labels) and unsupervised learning (data without labels). We then introduced maximum likelihood estimation (MLE) and maximum a posteriori estimation (MAP) as methods for defining appropriate cost functions. Finally, we discussed *iterative* optimization techniques, focusing on gradient descent and its variants.

In this lecture, we will focus on regression—a task distinct from classification—where the goal is to predict a *continuous* variable, $y^{(j)} \in \mathbb{R}$. We will also connect this task to previously covered concepts, using MLE and MAP to derive the cost function and applying gradient descent to minimize the observed cost function.

1 Chirping crickets and temperature

Let us consider a motivating example.[1] Did you know that you can estimate the temperature by measuring the rate of cricket chirps? The graph below marks measurements of the number of chirps (over 15 seconds) of a striped ground cricket at different temperatures (measured in degrees Fahrenheit).[2]



One can easily realize that these two are correlated: the higher the temperature, the faster the crickets chirp. We can quantify this by fitting a linear model as:

$$\text{temperature} = \theta_1 \cdot \text{chirp rate} + \theta_0 + \varepsilon,$$

for some θ_1 and θ_2, and ε is the error term that captures the variability in temperature that is not explained by the chirp rate alone (e.g., humidity, wind speed, measurement noise, etc.).

Now, our goal is recover θ_0 and θ_1 (essentially the green line above) to best the explain the observed relationship. More formally, we aim to learn a hypothesis function, $h(\text{chirp rate}; \theta_0, \theta_1)$,

[1] Example inspired from: https://jmahaffy.sdsu.edu/courses/s00/math121/lectures/linear/linear00.html.

[2] Chirping synchronously, as if led by the wand of a conductor: https://jmahaffy.sdsu.edu/courses/s00/math121/lectures/linear/cricket.wav.

such that

$$h(\text{chirp rate}; \theta_0, \theta_1) = \theta_0 + \theta_1 \cdot \text{chirp rate} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}^T \begin{bmatrix} 1 \\ \text{chirp rate} \end{bmatrix}.$$

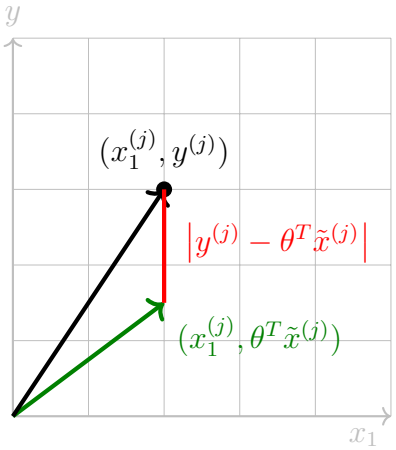
For a more general case of $x^{(j)} \in \mathbb{R}^d$, we write our hypothesis as

$$h(x^{(j)}; \theta) = \theta_0 + \sum_{i=1}^{d} \theta_i x_i^{(j)} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}^T \begin{bmatrix} 1 \\ x_1^{(j)} \\ \vdots \\ x_d^{(j)} \end{bmatrix} = \theta^T \tilde{x}^{(j)},$$

where $\theta \in \mathbb{R}^{d+1}$ is the vector of all parameters, including the bias term θ_0, and $\tilde{x}^{(j)} \in \mathbb{R}^{d+1}$ is the augmented input vector that includes the original features along with an additional intercept term, $x_0^{(j)} = 1$.

Now, given a training dataset, $\mathcal{D} = \{(x^{(j)}, y^{(j)}) | 1 \leq j \leq n\}$, how do we choose (or, learn) θ? A reasonable choice is to pick θ to make $h(x; \theta)$ close to y for all training samples. We formalize this using the cost function[3]

$$\begin{aligned} J(\theta) &= \frac{1}{2n} \sum_{j=1}^{n} (y^{(j)} - h(x^{(j)}; \theta))^2 \\ &= \frac{1}{2n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2. \end{aligned}$$



If you have seen linear regression before, you would recognize the above least squares cost function as giving rise to the familiar "ordinary" least squares regression model.

2 Finding optimal parameters

2.1 Gradient descent

So we wish to choose θ that minimizes the above least squares $J(\theta)$. Lucky for us, we know how to iteratively find the minimum of a given cost function using gradient descent.[4] Recall that in gradient descent, we start with some initial guess, $\theta^{(0)}$, and iterative refine our "guess" for θ values by taking sufficiently small steps in the steepest descent direction:

$$\theta^{(k+1)} = \theta^{(k)} + \alpha(-\nabla J(\theta^{(k)})),$$

where α is the step size and $\nabla J(\theta^{(k)})$ is the vector of partial derivatives, evaluated at $\theta^{(k)}$, also known as the *gradient* vector.

To implement gradient descent, we need to evaluate $\nabla J(\theta^{(k)})$. Let us start by estimating

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{\partial}{\partial \theta_i} \frac{1}{2n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2.$$

[3] It is reasonable to ask: why not simply minimize the absolute value; why least "squares" instead of least "absolute values"? As it turns out, the absolute value is not differentiable everywhere, which is needed for gradient-based optimization. Even beyond that, as we will later show (in §3), least *squares* falls out naturally as the MLE estimate from assuming IID Gaussian noise.

[4] Other variants of gradient descent can also be used here, including momentum, adaptive gradients, etc.

We can distribute partials over addition to get

$$\begin{aligned}
\frac{\partial}{\partial \theta_i} J(\theta) &= \frac{1}{2n} \sum_{j=1}^{n} \frac{\partial}{\partial \theta_i} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2 \\
&= \frac{1}{2n} \sum_{j=1}^{n} 2(y^{(j)} - \theta^T \tilde{x}^{(j)}) \frac{\partial}{\partial \theta_i} (y^{(j)} - \theta^T \tilde{x}^{(j)}) \\
&= -\frac{1}{n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)}) \frac{\partial}{\partial \theta_i} \theta^T \tilde{x}^{(j)} \\
&= -\frac{1}{n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)}) \boxed{\frac{\partial}{\partial \theta_i}} (\theta_0 + \theta_1 x_1^{(j)} + \cdots + \boxed{\theta_i x_i^{(j)}} + \cdots + \theta_d x_d^{(j)}) \\
&= -\frac{1}{n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)}) x_i^{(j)}.
\end{aligned}$$

Now, we have the gradient vector as:

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \\ \frac{\partial J}{\partial \theta_d} \end{bmatrix} = -\frac{1}{n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)}) \begin{bmatrix} x_0^{(j)} \\ \vdots \\ x_d^{(j)} \end{bmatrix} = -\frac{1}{n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)}) \tilde{x}^{(j)}.$$

Thus, our update rule is

$$\theta^{(k+1)} = \theta^{(k)} + \frac{\alpha}{n} \sum_{j=1}^{n} (y^{(j)} - (\theta^{(k)})^T \tilde{x}^{(j)}) \tilde{x}^{(j)}.$$

In discussing gradient descent, one of the things we noted was that gradient descent is susceptible to local optima; however, the cost function, $J(\theta)$ is convex and has only one global minima (see Appendix B for specifics). Thus, gradient descent always converges irrespective of the initial guess, so long as α is sufficiently small.

Computational complexity. The "summation over the dataset" term in the computation of the gradient vector is bothersome. To quantify this amount of "bother," we note the computational complexity for one iteration of gradient descent as $\mathcal{O}(nd)$, meaning gradient descent needs to perform $\mathcal{O}(nd)$ work before it can make a single update towards optimal parameters. The larger the dataset, the slower this is!

A rather simple idea to overcome this complexity is to approximate $J(\theta)$ at a given step by consider a single random sample. Simply put, instead of computing $\nabla J(\theta^{(k)})$, we compute

$$\nabla_{\tilde{j}} J(\theta^{(k)}) = -(y^{(\tilde{j})} - (\theta^{(k)})^T \tilde{x}^{(\tilde{j})}) \tilde{x}^{(\tilde{j})},$$

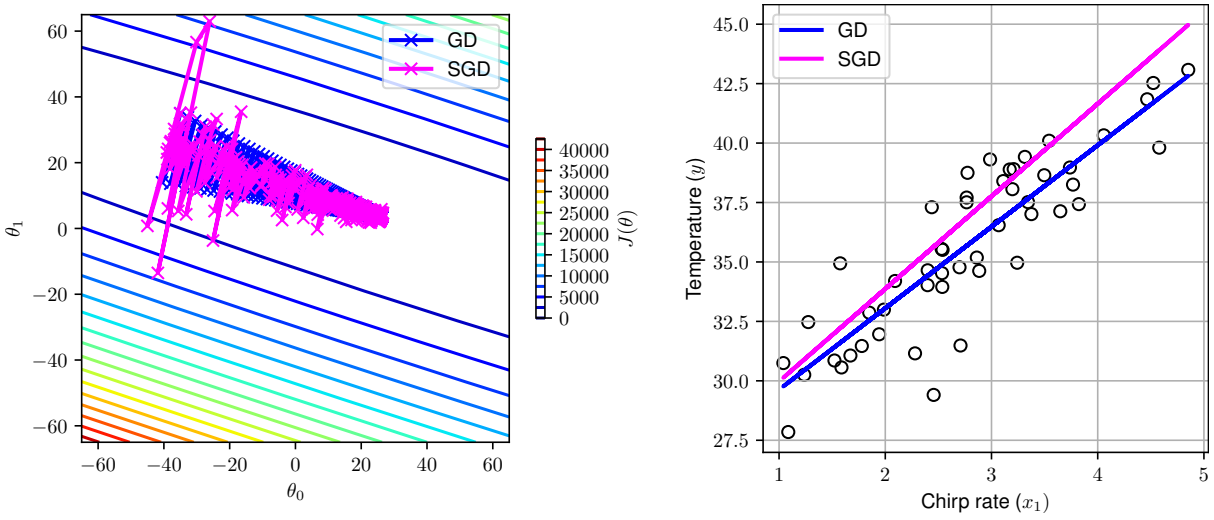
for some $\tilde{j}$ sampled uniformly at random from the training set at each update step. Now, we write our update step as

$$\theta^{(k+1)} = \theta^{(k)} + \alpha(-\nabla_{\tilde{j}} J(\theta^{(k)})) = \theta^{(k)} + \alpha(y^{(\tilde{j})} - (\theta^{(k)})^T \tilde{x}^{(\tilde{j})}) \tilde{x}^{(\tilde{j})}.$$

For obvious reasons, this iteration is called the *stochastic* gradient descent. Now, the cost-per-iteration is $\mathcal{O}(d)$ and no longer depends on the dataset size, n. We defer the reader to §3 of "Lecture 11: Of balls and bowls"[5] for more specifics on the iteration.

[5] https://cs.cornell.edu/courses/cs3780/2025sp/notes/lec11-gd-contd.pdf.

One final thing to note with stochastic gradient descent is that it often gets θ "close" to the optimum faster than gradient descent. However, it may never converge to the optimum and instead oscillate around the minimum of $J(\theta)$; in practice, this is often reasonably good enough approximation to the true minimum.[6] An example run of gradient descent and stochastic gradient descent for the chirps data is shown below.[7]



2.2 Closed-form solution

While gradient descent gives us one way of minimizing $J(\theta)$, the optimal parameters for the least squares objective can also be derived in a closed form. Here, we take the derivatives of $J(\theta)$, set them to zero, and solve for θ.

Given a training set of n samples, we can write the design matrix, $X \in \mathbb{R}^{n \times d+1}$ and vector $y \in \mathbb{R}^n$ as

$$X = \begin{bmatrix} — & (\tilde{x}^{(1)})^T & — \\ & \vdots & \\ — & (\tilde{x}^{(n)})^T & — \end{bmatrix}; \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}.$$

Now, recall that for a given vector, $v \in \mathbb{R}^n$, we compute $\|v\|_2^2 = v^T v = v_1^2 + \ldots + v_n^2$. From this, we have

$$(y - X\theta)^T (y - X\theta) = \begin{bmatrix} y^{(1)} - (\tilde{x}^{(1)})^T\theta \\ \vdots \\ y^{(n)} - (\tilde{x}^{(n)})^T\theta \end{bmatrix}^T \begin{bmatrix} y^{(1)} - (\tilde{x}^{(1)})^T\theta \\ \vdots \\ y^{(n)} - (\tilde{x}^{(n)})^T\theta \end{bmatrix} = \sum_{j=1}^{n} (y^{(j)} - (\tilde{x}^{(j)})^T\theta)^2.$$

This follows that,

$$\frac{1}{2n}\|y - X\theta\|^2 = \frac{1}{2n}(y - X\theta)^T(y - X\theta) = \frac{1}{2n}\sum_{j=1}^{n}(y^{(j)} - (\tilde{x}^{(j)})^T\theta)^2 = J(\theta).$$

Now, using our compact matrix-vectorial notation of $J(\theta)$, we can compute $\nabla_\theta J$ (gradient of

[6] It is also possible to decrease α values to zero as we approach the minimum to ensure convergence rather than mere oscillation about the minimum.

[7] It is clear that the problem is quite ill-conditioned; precisely, $\lambda_{\max}/\lambda_{\min} \approx 105$ (estimated from the data), making convergence from vanilla gradient descent approaches extremely slow (rate is 0.98, really close to one (no reduction in iterate error from step k to $k+1$)). It is advised to use AdaGrad and/or momentum here.

J with respect to θ) as

$$\begin{aligned}
\nabla_\theta J &= \nabla_\theta \frac{1}{2n}(y - X\theta)^T(y - X\theta) \\
&= \frac{1}{2n}\nabla_\theta(y^T - \theta^T X^T)(y - X\theta) \\
&= \frac{1}{2n}\nabla_\theta(\underbrace{y^T y}_{\text{no ``}\theta\text{'' term}} - \theta^T X^T y - \underbrace{y^T X\theta}_{=\theta^T X^T y} + \theta^T X^T X\theta) \\
&= \frac{1}{2n}\nabla_\theta(-2\theta^T X^T y + +\theta^T X^T X\theta) \\
&= \frac{1}{2n}(-2X^T y + 2X^T X\theta) \overset{\text{set}}{=} 0.
\end{aligned}$$

In the fourth step above, we used the fact that $y^T(X\theta) = (X\theta)^T y = \theta^T X^T y$. In the fifth step, we realize that $\nabla_\theta \theta^T x = x$, and $\nabla_\theta \theta^T A\theta = 2A\theta$ for a symmetric A ($X^T X$ is symmetric).[8] Now, solving for θ, we get:

$$X^T X\theta = X^T y, \;\; \text{OR} \;\; \theta^\star = (X^T X)^{-1} X^T y.$$

Hence, for the least squares objective, we can derive optimal $\theta^\star$ in closed form. We note here that the above closed form can also be realized by running the Newton iteration for one step; we leave proof of this as a self-exercise.

It is important to state that we are implicitly assuming $X^T X$ to be invertible. It is easy to check beforehand if $X^T X$ is invertible. If the features are not linearly independent (i.e., $X^T X$ does not have full column rank), or if the number of linearly independent examples is fewer than the number of features, i.e., $n \leq d$, the null space of $X^T X$ is non-trivial, making $X^T X$ not invertible. Even in such cases, we can fix the situation using additional techniques such as regularization, which we will discuss later (in §4).

Gradient descent vs. closed form. It is easy to see that closed form solution requires forming and inverting $X^T X$, which takes $\mathcal{O}(nd^2 + n^3)$ time, which makes it expensive for large-scale problems. Additionally $X^T X$ being near singular adds to further numerical complications. In contrast, gradient descent, or better yet, stochastic gradient descent offer $\mathcal{O}(dT)$ for T total iterations, which is significantly cheaper, and they do not require forming or inverting $X^T X$.

3 Probabilistic interpretation

In designing $J(\theta)$ as the least squares objective, we noted how it intuitively fits the problem we are solving. It so happens that under a set of probabilistic assumptions, we can realize the least squares objective as naturally arising from maximizing likelihood of the observed data.

Recall our assumption of data having been generated from the linear model

$$y^{(j)} = \theta^T \tilde{x}^{(j)} + \epsilon^{(j)},$$

where $\epsilon^{(j)}$ is the error term that captures unmodeled effects (e.g., in our cricket chirps example, humidity, wind speed, measurement noise, etc. contribute to this). Additionally, let us assume that $\epsilon^{(j)}$ are drawn IID from a Gaussian distribution with zero mean and variance σ^2, i.e., $\epsilon^{(j)} \sim \mathcal{N}(0, \sigma^2)$.[9] This is not an unreasonable assumption since the central limit theorem tells us that small, independent random (unmodeled) effects tend to sum to a Gaussian distribution.

[8] For those less familiar with matrix derivatives, The Matrix Cookbook serves as a good reference: https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf.

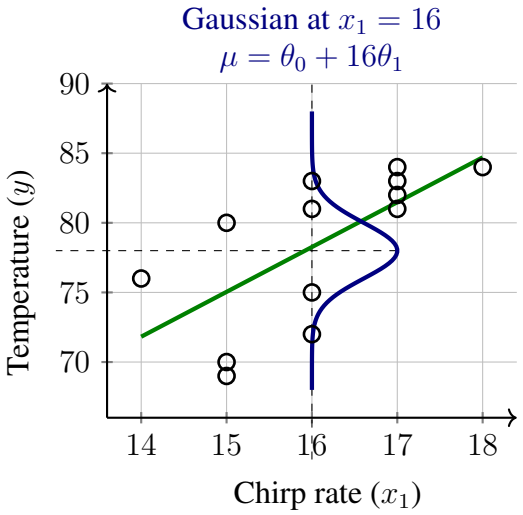
[9] If a non-zero mean is assumed, it can absorbed into the bias term, θ_0.

Now, we can write the density of $\epsilon^{(j)}$ as

$$P(\epsilon^{(j)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(j)})^2}{2\sigma^2}\right).$$

This implies that $y^{(j)}|x^{(j)} \sim \mathcal{N}(\theta^T \tilde{x}^{(j)}, \sigma^2)$ (illustration to the right), giving

$$P(y^{(j)}|x^{(j)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(j)} - \theta^T \tilde{x}^{(j)})^2}{2\sigma^2}\right).$$



By the independence of $\epsilon^{(j)}$,s we can realize the likelihood function as

$$\begin{aligned} L(\theta; X, y) = P(\mathcal{D};\theta) &= \prod_{j=1}^{n} P(x^{(j)}, y^{(j)};\theta) \\ &= \prod_{j=1}^{n} P(y^{(j)}|x^{(j)};\theta) P(x^{(j)}). \end{aligned}$$

As seen before, instead of maximizing $L(\theta)$, we will instead maximize the *log* likelihood:

$$\begin{aligned} \ell(\theta) = \log(L(\theta)) &= \log \prod_{j=1}^{n} P(y^{(j)}|x^{(j)};\theta) P(x^{(j)}) \\ &= \sum_{j=1}^{n} \log(P(y^{(j)}|x^{(j)};\theta)) + \sum_{j=1}^{n} \log(P(x^{(j)})) \end{aligned}$$

Since our objective is to find θ such that $\ell(\theta)$ is maximized, the "$\sum_j \log(P(x^{(j)}))$" term above doesn't affect the maximization. Hence,

$$\begin{aligned} \theta^\star = \arg\max_\theta \ell(\theta) &\equiv \arg\max_\theta \sum_{j=1}^{n} \log(P(y^{(j)}|x^{(j)};\theta)) \\ &= \arg\max_\theta \sum_{j=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(j)} - \theta^T \tilde{x}^{(j)})^2}{2\sigma^2}\right) \\ &= \arg\max_\theta \underbrace{\sum_{j=1}^{n} \log \frac{1}{\sqrt{2\pi}\sigma}}_{\text{same as before; a constant in our optimization}} + \sum_{j=1}^{n} -\frac{(y^{(j)} - \theta^T \tilde{x}^{(j)})^2}{2\sigma^2} \\ &= \arg\max_\theta -\frac{1}{2\sigma^2} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2 \\ &= \arg\min_\theta \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2 \\ &\equiv \arg\min_\theta \frac{1}{2n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2 \\ &= \arg\min_\theta J(\theta). \end{aligned}$$

This, we can observe how the least squares cost function, $J(\theta)$ falls out as a natural MLE

estimate under the assumption of an IID Gaussian noise. One final observation to make here is that our final estimate for optimal $\theta^\star$ does not depend on σ^2, i.e., irrespective of the variance of the noise term (width of the blue Gaussian in the illustration), we arrive at the same line.

4 Regularized least squares

Consider the case where we have more parameters than samples (classic case of an underdetermined system). More formally, if the null space of X is non-trivial, for any $\theta^\star$ that minimizes the least squares objective and θ^0 in the null space of X, we have $\theta^\star + \alpha\theta^0$ (for some constant, α) to also minimize the least squares objective:

$$X(\theta^\star + \alpha\theta^0) = X\theta^\star + \alpha \underbrace{X\theta^0}_{=0} = X\theta^\star.$$

In this case, it seems intuitive to choose $\theta^\star$ as the optimal solution (and not $\theta^\star + \alpha\theta^0$). To this end, we can rewrite the least squares objective with additional constraints as

$$J(\theta) = \frac{1}{2n}(\|y - X\theta\|^2 + \lambda^2\|\theta\|_2^2),$$

where λ is some hyperparameter that controls how much to penalize large weights. This is known as the *regularized* least squares. A simple observation we make is that the above regularized least squares cost function can be written as:

$$J(\theta) = \frac{1}{2n}\left\|\begin{bmatrix} X \\ \lambda \mathrm{I}_{d+1} \end{bmatrix}\theta - \begin{bmatrix} y \\ 0 \end{bmatrix}\right\|_2^2 = \frac{1}{2n}\|X_*\theta - y_*\|^2,$$

meaning, regularized least squares is ordinary least squares with $d+1$ additional rows. In this form, it is easy to realize that $X_* \in \mathbb{R}^{(n+d+1)\times(d+1)}$ will ensure that $X_*^T X_* = X^T X + \lambda^2 \mathrm{I}_{d+1}$ is invertible and has all linearly independent columns (full rank). Following §2.2, we can write the closed form solution to regularized least squares as

$$\theta^\star = (X_*^T X_*)^{-1} X_*^T y = (X^T X + \lambda^2 \mathrm{I}_{d+1})^{-1} X^T y.$$

One final thing we note is that one can easily realize the regularized version of least squares by assuming the prior that the parameters are normally distributed and using an MAP estimate; see Appendix C for specifics.

5 Conclusion

In this lecture, we discussed the linear *regression* setting, where our goal was to estimate a continuous variable using a linear model. We intuitively devised a cost function as the average squared deviation of the predicted values from the target values and argued this to be a natural choice under the assumption of an IID Gaussian noise. In minimizing our least squares objective, we realized the (stochastic) gradient descent and also derived a closed form solution.

Finally, we discussed how adding a penalty term, $\|\theta\|_2$, enabled additional, "nicer" properties in the choice of model parameters. It is worth noting that the penalty doesn't necessary have to use the ℓ^2 norm; another common choice is the ℓ^1 norm, which enables sparsity in the estimates.[10]

One last note: throughout this lecture, we modeled y as $\theta_0 + \theta_1 x_1$ (fit a line) but one could imagine $\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ (fit a quadratic), or an n-th order polynomial to $n+1$ data points (obviously bad). We will discuss tradeoffs between these when we talk about bias and variance.

[10] Nice visualization of ℓ^1 vs. ℓ^2 regularization: https://aunnnn.github.io/ml-tutorial/html/blog_content/linear_regression/linear_regression_regularized.html.

A Notation

$\mathcal{D}$	The training dataset of n samples
n	The number of training samples in the dataset $\mathcal{D}$
d	The number of parameters we wish to estimate. Note that we also use d to represent the number of features; these need not be the same, as we will see later in the course
$x^{(j)} \in \mathbb{R}^d$	The d-dimensional feature vector associated with the j-th training sample
$y^{(j)} \in \mathbb{R}$	The target value associated with the j-th training sample
$x_\ell^{(j)} \in \mathbb{R}$	The ℓ-th element of $x^{(j)}$
$\epsilon^{(j)}$	The error term capturing the unmodeled effects (e.g., measurement noise), associated with the j-th training sample.
$\tilde{x}^{(j)} \in \mathbb{R}^{d+1}$	Includes the intercept term, $x_0^{(j)} = 1$, along with the d feature values associated with the j-th training sample
$\theta \in \mathbb{R}^{d+1}$	A vector of $d+1$ model parameters, including the bias term, θ_0
$\theta_\ell \in \mathbb{R}$	The ℓ-th element of θ
$h(x^{(j)};\theta) \in \mathbb{R}$	The hypothesis function that estimates the target value for a given $x^{(j)}$
$J(\theta)$	The (regularized) least squares cost function we are trying to minimize
$\theta^{(k)} \in \mathbb{R}^{d+1}$	The k-th iterate of model parameters
$\theta^\star \in \mathbb{R}^{d+1}$	The optimal model parameters that minimize $J(\theta)$
$\nabla J(\theta^{(k)}) \in \mathbb{R}^{d+1}$	A vector of partial derivatives of $J(\theta)$, evaluated at $\theta^{(k)}$, known as the gradient vector. Also represented as $\nabla_\theta J(\theta^{(k)})$ to explicitly indicate partials with respect to θ (not to be confused with subscript $\tilde{j}$, i.e., $\nabla_{\tilde{j}} J(\theta^{(k)})$)
$\nabla_{\tilde{j}} J(\theta^{(k)}) \in \mathbb{R}^{d+1}$	The approximation of gradient vector $\nabla J(\theta^{(k)})$ obtained by using the cost function $J(\theta^{(k)})$ evaluated at $(x^{(\tilde{j})}, y^{(\tilde{j})})$, instead of for all training samples
α	The learning rate or step size used to move along the steepest descent direction in (stochastic) gradient descent
$\|u\|_2$	The ℓ^2 norm of a vector u, $\|u\|_2^2 = u^T u$
$X \in \mathbb{R}^{n \times d+1}$	The design matrix containing all d-dimensional data points, $x^{(j)}$s
$y \in \mathbb{R}^n$	The vector of all target values for all n samples in the training set
$P(a\|b;\theta)$	The conditional probability (density) of a given b, parameterized by θ
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution defined by mean, μ, and variance, σ^2
$\mathrm{I}_{d+1} \in \mathbb{R}^{(d+1)\times(d+1)}$	A $d+1 \times d+1$ identity matrix
λ (or, λ^2)	The regularization parameter in regularized least squares (not to be confused with eigenvalues)
$X_* \in \mathbb{R}^{(n+d+1)\times(d+1)}$	$X \in \mathbb{R}^{n\times d+1}$ with $d+1$ additional rows as $\lambda \mathrm{I}_{d+1}$
$y_* \in \mathbb{R}^{n+d+1}$	$y \in \mathbb{R}^n$ with $d+1$ additional zeroes appended at the bottom

B Optimal settings for gradient descent

Given a training dataset, $\mathcal{D} = \{(x^{(j)}, y^{(j)}) | 1 \le j \le n\}$, we can construct the design matrix, $X \in \mathbb{R}^{n\times d+1}$ as:

$$X = \begin{bmatrix} — & (\tilde{x}^{(1)})^T & — \\ & \vdots & \\ — & (\tilde{x}^{(n)})^T & — \end{bmatrix}.$$

Now, let $y \in \mathbb{R}^n$ be the vector containing all target values from the training data:

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}.$$

Now, we realize that

$$y - X\theta = \begin{bmatrix} y^{(1)} - (\tilde{x}^{(1)})^T\theta \\ \vdots \\ y^{(n)} - (\tilde{x}^{(n)})^T\theta \end{bmatrix} = \begin{bmatrix} y^{(1)} - h(x^{(1)};\theta) \\ \vdots \\ y^{(n)} - h(x^{(n)};\theta) \end{bmatrix}.$$

Next, recall that, for a given vector, $v \in \mathbb{R}^n$, we have $v^Tv = \|v\|^2 = v_1^2 + \ldots + v_n^2$. Hence,

$$\frac{1}{2n}(y - X\theta)^T(y - X\theta) = \frac{1}{2n}\sum_{j=1}^{n}(y^{(j)} - h(x^{(j)};\theta))^2 = J(\theta).$$

Hence, we have $J(\theta)$ in the matrix-vectorial form as:

$$\begin{aligned} J(\theta) &= \frac{1}{2n}(y - X\theta)^T(y - X\theta) \\ &= \frac{1}{2n}(y^T - \theta^T X^T)(y - X\theta) \\ &= \frac{1}{2n}(y^Ty - \theta^T X^T y - y^T X\theta + \theta^T X^T X\theta). \end{aligned}$$

With some rearranging, we have

$$J(\theta) = \frac{1}{2}\theta^T(X^TX/n)\theta - \frac{1}{2n}(\theta^T X^T y + y^T X\theta) + \frac{1}{2n}y^Ty,$$

which we realize as the previously-presented (in the gradient descent lecture) as the general form of a quadratic $(1/2)\theta^T A\theta + b^T\theta + c$. Additionally, it is easy to realize X^TX/n is symmetric and $X^TX/n \succeq 0$ (positive semi-definite); for any $v \neq 0$, we have

$$\langle\frac{1}{n}(X^TX)v, v\rangle = \frac{1}{n}v^TX^TXv = \frac{1}{n}\langle Xv, Xv\rangle = \frac{1}{n}\|Xv\|_2^2 \geq 0.$$

We also note that when X has full column rank (of d for $n \geq d$), i.e., X has a trivial null space, $v = 0$ is the only solution to $Xv = 0$, making X^TX positive definite. Hence, when X has full column rank, $J(\theta)$ is strictly convex and has a single global minima.

If $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of X^TX/n,[11] from our analysis on gradient descent, we realize any $\alpha < 2/\lambda_{\max}$ guarantees convergence, with the optimal α as

$$\alpha_{\text{opt}} = \frac{2}{\lambda_{\min} + \lambda_{\max}}.$$

C MAP estimate for least squares

Recall that in estimating parameters using MAP, we assume θ to be a random variable and estimate θ as

$$\theta^\star = \arg\max_\theta P(\theta|\,\mathcal{D}) \equiv \arg\max_\theta \log(P(\mathcal{D}\,|\theta)) + \log(P(\theta)).$$

[11] For those familiar with singular value decomposition, it is fairly straightforward to show that for a given $X = U\Sigma V^T$, $\lambda_{\max} = \sigma_{\max}^2/n$ and $\lambda_{\min} = \sigma_{\min}^2/n$, where $\sigma_{\min}$ and $\sigma_{\max}$ are the smallest and largest singular values of X.

Now, if we assume an IID Gaussian prior for the parameters, i.e., $\theta_i \sim \mathcal{N}(0, \tau^2)$, we have

$$P(\theta_i) = \frac{1}{\sqrt{2\pi}\tau} \exp\left(-\frac{\theta_i^2}{2\tau^2}\right).$$

Following this, we can write

$$\begin{aligned}
\log(P(\theta)) &= \log \prod_{i=1}^{d} P(\theta_i) \\
&= \sum_{i=1}^{d} \log \frac{1}{\sqrt{2\pi}\tau} + \sum_{i=1}^{d} -\frac{\theta_i^2}{2\tau^2} \\
&= n \log \frac{1}{\sqrt{2\pi}\tau} - \frac{1}{2\tau^2} \sum_{i=1}^{d} \theta_i^2 \\
&= n \log \frac{1}{\sqrt{2\pi}\tau} - \frac{1}{2\tau^2} \|\theta\|_2^2.
\end{aligned}$$

Hence,

$$\arg\max_{\theta} \log(P(\theta)) \equiv \arg\min_{\theta} \frac{1}{2\tau^2} \|\theta\|_2^2.$$

Now, following our derivation in §3 for $P(\mathcal{D}\,|\theta)$, we realize that the MAP estimate is

$$\begin{aligned}
\theta^\star &= \arg\min_{\theta} \frac{1}{2\sigma^2} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2 + \frac{1}{2\tau^2} \|\theta\|_2^2 \\
&= \arg\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2 + \frac{\sigma^2}{2\tau^2} \|\theta\|_2^2 \\
&\equiv \arg\min_{\theta} \frac{1}{2n} \sum_{j=1}^{n} (y^{(j)} - \theta^T \tilde{x}^{(j)})^2 + \frac{\sigma^2}{2n\tau^2} \|\theta\|_2^2.
\end{aligned}$$

By letting $\lambda^2 = \sigma^2/\tau^2$, we realize the MAP estimate to be the regularized least squares objective.

□

References

D. Bindel. CS4220 Lecture notes on Intro to Least Squares. *CS4220 Lecture notes*, 1(1): 1–4, 2023a. URL https://www.cs.cornell.edu/courses/cs4220/2023sp/lec/2023-02-20.pdf.

D. Bindel. CS4220 Lecture notes on Ill-posedness and Regularization. *CS4220 Lecture notes*, 1(1):4–6, 2023b. URL https://www.cs.cornell.edu/courses/cs4220/2023sp/lec/2023-02-24.pdf.

A. Ng. CS229 Lecture notes. *CS229 Lecture notes*, 1(1):8–17, 2000. URL https://cs229.stanford.edu/main_notes.pdf. Version: June 11, 2023.

(Last compiled: 3/23/2025, 10.21am ET.)